UNIVERSIDAD DE OVIEDO

DOCTORADO EN INGENIERÍA INFORMÁTICA

TESIS DOCTORAL

“Diseño de objetos virtuales colaborativos orientados a servicios en el marco de Internet de las cosas”

Presentado por

Jordán Pascual Espada

TESIS DOCTORAL

“Diseño de objetos virtuales colaborativos orientados a servicios en el marco de Internet de las cosas”

Presentado por

Jordán Pascual Espada

Para la obtención del título de Doctor en Informática

Dirigido por

Doctor D. Juan Manuel Cueva Lovelle

Doctor D. Oscar Sanjuán Martínez

Oviedo, 2012

UNIVERSIDAD DE OVIEDO
Vicerrectorado de Ordenación Académica
y Nuevas Titulaciones

CENTRO INTERNACIONAL DE POSTGRADO
CAMPUS DE EXCELENCIA INTERNACIONAL

RESUMEN DEL CONTENIDO DE TESIS DOCTORAL

1.- Título de la Tesis	
Español/Otro Idioma: Diseño de objetos virtuales colaborativos orientados a servicios en el marco de internet de las cosas.	Inglés: Design service-oriented collaborative virtual objects within Internet of things.

2.- Autor	
Nombre: Jordán Pascual Espada	
Programa de Doctorado: Doctorado en ingeniería informática	
Órgano responsable: Centro internacional de postgrado	

RESUMEN (en español)

La idea general que hay detrás de "Internet de las cosas" es sencilla: cualquier "cosa", es decir, cualquier objeto convenientemente etiquetado, podrá ser capaz de interactuar o comunicarse con otros objetos y sistemas, ya sea utilizando Internet, redes privadas u otros mecanismos de comunicación.

Una pieza clave en el desarrollo del Internet de las cosas son los teléfonos móviles inteligentes, conocidos como Smartphones. A día de hoy millones de usuarios utilizan asiduamente sus Smartphones para interactuar con diversos tipos de objetos físicos o dispositivos electrónicos de su entorno. En la mayoría de los casos, el proceso de interacción entre el Smartphone y los objetos físicos o dispositivos está gestionado por una aplicación software específica. Debido a las características de este tipo de aplicaciones, en la mayoría de casos deben de ser desarrolladas como aplicaciones nativas específicamente para la plataforma móvil destino.

La utilización de aplicaciones software nativas en los procesos de interacción entre Smartphones y objetos físicos produce varias consecuencias no deseadas, entre las que se encuentran los elevados costes y dificultades de desarrollo derivadas de la replicación de desarrollos en distintas plataformas móviles. Por otra parte, la gestión de las actuales aplicaciones nativas móviles implica una serie de procesos secundarios, como la descarga, instalación y configuración. Estos procesos no son adecuados para los sistemas que se basan en la interacción ocasional o puntual con objetos físicos o dispositivos, puesto que las acciones secundarias podrían llegar a consumir más tiempo que el propio proceso de interacción.

Motivados por los problemas expuestos se ha procedido al desarrollo de esta tesis doctoral. En ésta se define un modelo aplicable al desarrollo de aplicaciones móviles que basan una parte importante de su funcionalidad en la interacción con objetos físicos o dispositivos electrónicos cercanos. El modelo propuesto no está ligado a ninguna plataforma móvil ni tecnología de desarrollo específica, por ello permite el desarrollo de aplicaciones móviles validas para múltiples plataformas. Además la propuesta incluye optimizaciones relativas al desarrollo de este tipo de aplicaciones, como la abstracción en la gestión de los elementos hardware del dispositivo que habilitan las comunicaciones y la captura de información de contexto. La propuesta reduce considerablemente los tiempos empleados por los procesos secundarios, permitiendo que las aplicaciones desarrolladas sean óptimas para su uso en sistemas que se basan en la interacción ocasional con objetos o dispositivos electrónicos. El modelo propuesto es el resultado de tres procesos de interacción, cada uno de ellos ha tenido como resultado modificaciones significativas en la

arquitectura y características de la propuesta, con el fin de que ésta se adecue de la mejor forma posible a la consecución de los objetivos. La versión final del modelo se basa en una especificación que se combina con las tecnologías web, las aplicaciones web generadas se ejecutan parcialmente en el propio dispositivo cliente, además de en el servidor remoto. Este modo de ejecución mixto permite a las aplicaciones web gestionar los elementos hardware del dispositivo cliente, con el fin de realizar comunicaciones o capturar información de contexto.

Por último, las herramientas y aplicaciones desarrolladas siguiendo las especificaciones presentadas han servido para evaluar y verificar diversos aspectos de la propuesta.

RESUMEN (en Inglés)

The main idea behind the "Internet of the things" is quite simple: any properly tagged object will be able to interact or communicate with other objects and systems via Internet, private networks or any other form of communication. Intelligent cell phones, known as Smartphones, are a key element in the development of the Internet of the things. Nowadays, millions of users use their Smartphones to interact with all kinds of physical objects or electronic devices in their environment. In most cases, the interaction process between the Smartphone and the physical objects or devices is managed by a specific software application. Due to these application's features, they often have to be developed specifically as native applications for the target mobile platform. The use of native software applications in the interaction process between Smartphones and physical objects has several unwelcomed consequences, like high costs and development hardships caused by the replication of developments in different mobile platforms. On the other hand, the actual native application's management implies a series of secondary processes, like the download, installation and configuration. These processes are inappropriate for systems that are based on occasional or precise interaction with physical objects or devices, as secondary actions could take more time than the interaction process itself.
The exposed problems have been the motivation for the development of this doctoral thesis, which defines a model applied to the development of mobile applications that base a strong part of their functionality in the interaction with physical objects or nearby electronic devices. The proposed model is not attached to any mobile platform or specific development technology, so it allows the development of valid mobile applications for multiple platforms. Also, the proposal includes optimizations related to the development of this type of applications, such as the abstraction in the management of the hardware elements of the device that allows communication and the capture of context information. The proposed model dramatically reduces the time used in the secondary processes, making the developed applications optimal for their use in occasional interaction based systems with objects or electric devices. The proposed model is the result of three interaction processes, and each of one has resulted in significant modifications in the architecture and characteristics of the proposal, looking for the best way of reaching this goals. The final version of the model is based on a specification that combines with web technologies. The resulting web applications are partially executed in the client device itself, and also in the remote server. This mixed execution method allows web applications to manage the client device's hardware elements, aiming to perform communication or capturing context information.

Finally, the tools and applications developed by following the proposed specifications have helped to speed up the experiments performed to evaluate and verify several aspects of the proposal.

RESUMEN

La idea general que hay detrás de "Internet de las cosas" es sencilla: cualquier "cosa", es decir, cualquier objeto convenientemente etiquetado, podrá ser capaz de interactuar o comunicarse con otros objetos y sistemas, ya sea utilizando Internet, redes privadas u otros mecanismos de comunicación.

Una pieza clave en el desarrollo del Internet de las cosas son los teléfonos móviles inteligentes, conocidos como Smartphones. A día de hoy millones de usuarios utilizan asiduamente sus Smartphones para interactuar con diversos tipos de objetos físicos o dispositivos electrónicos de su entorno. En la mayoría de los casos, el proceso de interacción entre el Smartphone y los objetos físicos o dispositivos está gestionado por una aplicación software específica. Debido a las características de este tipo de aplicaciones, en la mayoría de casos deben de ser desarrolladas como aplicaciones nativas específicamente para la plataforma móvil destino.

La utilización de aplicaciones software nativas en los procesos de interacción entre Smartphones y objetos físicos produce varias consecuencias no deseadas, entre las que se encuentran los elevados costes y dificultades de desarrollo derivadas de la replicación de desarrollos en distintas plataformas móviles. Por otra parte, la gestión de las actuales aplicaciones nativas móviles implica una serie de procesos secundarios, como la descarga, instalación y configuración. Estos procesos no son adecuados para los sistemas que se basan en la interacción ocasional o puntual con objetos físicos o dispositivos, puesto que las acciones secundarias podrían llegar a consumir más tiempo que el propio proceso de interacción.

Motivados por los problemas expuestos se ha procedido al desarrollo de esta tesis doctoral. En ésta se define un modelo aplicable al desarrollo de aplicaciones móviles que basan una parte importante de su funcionalidad en la interacción con objetos físicos o dispositivos electrónicos cercanos. El modelo propuesto no está ligado a ninguna plataforma móvil ni tecnología de desarrollo específica, por ello permite el desarrollo de aplicaciones móviles validas para múltiples plataformas. Además la propuesta incluye optimizaciones relativas al desarrollo de este tipo de aplicaciones, como la abstracción en la gestión de los elementos hardware del dispositivo que habilitan las comunicaciones y la captura de información de contexto. La propuesta reduce considerablemente los tiempos empleados por los procesos secundarios, permitiendo que las aplicaciones desarrolladas sean óptimas para su uso en sistemas que se basan en la interacción ocasional con objetos o dispositivos electrónicos. El modelo propuesto es el resultado de tres procesos de interacción, cada uno de ellos ha tenido como resultado modificaciones significativas en la arquitectura y características de la propuesta, con el fin de que ésta se adecue de la mejor forma posible a la consecución de los objetivos. La versión final del modelo se basa en una especificación que se combina con las tecnologías web, las aplicaciones web generadas se ejecutan parcialmente en el propio dispositivo cliente, además de en el servidor remoto. Este modo de ejecución mixto permite a las aplicaciones web gestionar los elementos hardware del dispositivo cliente, con el fin de realizar comunicaciones o capturar información de contexto.

Por último, las herramientas y aplicaciones desarrolladas siguiendo las especificaciones presentadas han servido para evaluar y verificar diversos aspectos de la propuesta.

PALABRAS CLAVE

Internet de las cosas, Objeto virtual, Teléfonos móviles inteligentes, Navegador web móvil, Servicios web, Aplicaciones web, Información de contexto, Control remoto, Dispositivos móviles.

ABSTRACT

The main idea behind the "Internet of the things" is quite simple: any properly tagged object will be able to interact or communicate with other objects and systems via Internet, private networks or any other form of communication. Intelligent cell phones, known as Smartphones, are a key element in the development of the Internet of the things. Nowadays, millions of users use their Smartphones to interact with all kinds of physical objects or electronic devices in their environment. In most cases, the interaction process between the Smartphone and the physical objects or devices is managed by a specific software application. Due to these application's features, they often have to be developed specifically as native applications for the target mobile platform. The use of native software applications in the interaction process between Smartphones and physical objects has several unwelcomed consequences, like high costs and development hardships caused by the replication of developments in different mobile platforms. On the other hand, the actual native application's management implies a series of secondary processes, like the download, installation and configuration. These processes are inappropriate for systems that are based on occasional or precise interaction with physical objects or devices, as secondary actions could take more time than the interaction process itself.

The exposed problems have been the motivation for the development of this doctoral thesis, which defines a model applied to the development of mobile applications that base a strong part of their functionality in the interaction with physical objects or nearby electronic devices. The proposed model is not attached to any mobile platform or specific development technology, so it allows the development of valid mobile applications for multiple platforms. Also, the proposal includes optimizations related to the development of this type of applications, such as the abstraction in the management of the hardware elements of the device that allows communication and the capture of context information. The proposed model dramatically reduces the time used in the secondary processes, making the developed applications optimal for their use in occasional interaction based systems with objects or electric devices. The proposed model is the result of three interaction processes, and each of one has resulted in significant modifications in the architecture and characteristics of the proposal, looking for the best way of reaching this goals. The final version of the model is based on a specification that combines with web technologies. The resulting web applications are partially executed in the client device itself, and also in the remote server. This mixed execution method allows web applications to manage the client device's hardware elements, aiming to perform communication or capturing context information.

Finally, the tools and applications developed by following the proposed specifications have helped to speed up the experiments performed to evaluate and verify several aspects of the proposal.

KEYWORDS

Internet of things, Virtual object, Smartphones, Mobile web browser, Web services, Web applications, Context information, Remote control, Mobile devices.

AGRADECIMIENTOS

Esta Tesis ha sido un largo camino en que han participado de manera directa o indirecta muchas personas, me gustaría expresar mi agradecimiento a todas ellas.

En primer lugar quisiera agradecer todo el apoyo recibido por parte de mis directores Dr. D. Juan Manuel Cueva Lovelle y Dr. D. Oscar Sanjuán Martínez. Gracias a los dos por sus orientaciones, ánimos y por haber estado dispuestos a ayudarme siempre que he necesitado.

Me gustaría resaltar mi agradecimiento a Sofi, Noemí, David y Carlos por el esfuerzo que han realizado y la gran ayuda prestada en el desarrollo de este trabajo.

Agradezco a todos mis compañeros del OOTLAB su predisposición a colaborar siempre que lo he requerido. Espero no olvidar ningún nombre si es así pido que me perdonen. Edward, Manuel, Weena, Pablo, Ismael, Yuri, Julio, Alberto, Natalia, Jaime, Chema, Nacho, César y Guillermo.

Y por supuesto quisiera agradecer el apoyo constante de mi familia y en especial el de mis padres.

Gracias a todos.

ÍNDICE

PARTE I

PLANTEAMIENTO DEL PROBLEMA

CAPÍTULO 1

INTRODUCCIÓN

1.1 INTRODUCCIÓN

Durante este capítulo se presenta la justificación y los objetivos de esta tesis doctoral.

Se comienza exponiendo las motivaciones de la investigación, donde se presentan los problemas y deficiencias detectadas en la situación actual, posteriormente se describirán los requisitos generales y específicos establecidos y para finalizar se describirán las aportaciones y beneficios derivados de la presente investigación.

1.2 MOTIVACIONES

En la actualidad muchos sistemas de información basan una parte importante de su funcionalidad en el intercambio de datos y servicios, implicando de forma directa o indirecta a diferentes tipos de objetos físicos. Los objetos físicos que participan en este tipo de sistemas van desde equipamiento del hogar e industria, a electrodomésticos, vehículos, incluso objetos consumibles como los productos del supermercado.

A medida que la tecnología avanza, aumenta la variedad de objetos físicos que, de alguna manera, pasan a formar parte de algunos tipos de sistemas de información. Los nuevos avances tecnológicos no sólo influyen en los propios objetos físicos, también lo hacen en las posibilidades de interacción y colaboración de estos con otros objetos físicos y aplicaciones informáticas.

El término "Internet of things" el cual suele traducirse al español como "Internet de las cosas" se atribuye originalmente al Auto-ID Lab sucesor del MIT (Instituto tecnológico de Massachusetts) Auto-ID Center, con mención especial a Kevin Ashton en 1999[1] y a David L. Brock en 2001 [Brock, 2001]. La idea general que hay detrás de Internet de las cosas es sencilla: cualquier "cosa", es decir, cualquier objeto convenientemente etiquetado, podrá ser capaz de comunicarse con otros objetos y sistemas, ya sea utilizando Internet, redes privadas u otros mecanismos de comunicación [Kortuem et al., 2010 ; Yan et al., 2008].

El Smartphone o teléfono móvil inteligente se está perfilando como una de las piezas claves en el acercamiento de los usuarios a los sistemas que permiten la interacción con objetos físicos [Thompson, 2005]. La buena acogida de los Smartphones en la sociedad y sus destacadas características técnicas de comunicación y procesamiento han propiciado que cada vez se usen para realizar un mayor número de tareas. Muchas de estas nuevas tareas implican la interacción con objetos físicos de diversos tipos, los sistemas que permiten la realización de este tipo de tareas estarían enmarcados en el Internet de las cosas [Kanma et al., 2003 ; Nichols & Myers, 2006].

A día de hoy millones de usuarios utilizan asiduamente sus Smartphones para interactuar con diversos tipos de objetos físicos de su entorno, como por ejemplo: identificar productos en el supermercado, controlar remotamente los electrodomésticos de su hogar [Nichols & Myers, 2006] [Piyare & Tazil, 2011 ; Juan et al., 2011] o interactuar con quioscos proveedores de servicios en las llamadas "Smart Cities" o ciudades inteligentes [Ferreira & Afonso, 2011 ; Sanchez et al., 2011 ; Naphade et al., 2011].

Las tiendas de aplicaciones móviles Android Market[2] e iTunes[3] contienen cientos de aplicaciones que basan su funcionalidad en la interacción con diversos tipos de objetos electrónicos: LG TV Remote, Samsung Remote, DSLR Remote Controller, AV Controller Yamaha, Logitech Squeezebox Controller, etc. La aceptación de este tipo de aplicaciones se

[1] "I could be wrong, but I'm fairly sure the phrase 'Internet of Things' started life as the title of a presentation I made at Procter & Gamble (P&G) in 1999", Kevin Ashton, RFID Journal, 22 June 2009.
[2] http://www.androidmarket.com
[3] http://www.apple.com/itunes/

refleja en el número de usuarios, por ejemplo la aplicación Samsung Remote Android[4] ha conseguido más de 750.000 usuarios solamente en los últimos seis meses del año 2011. Casi con total seguridad en el futuro la cantidad de objetos físicos con los que los usuarios podrán interactuar utilizando su Smartphone aumentará drásticamente.

En la mayoría de casos la interacción entre el Smartphone y los objetos físicos está condicionada al uso de una aplicación software específica. Este tipo de aplicaciones suelen tener un requerimiento común, el acceso a algunos de los elementos hardware del teléfono como: bluetooth, Wi-Fi, NFC (Near Field communication) [Ok et al., 2010] [Wei-Dar et al., 2011], la cámara de fotos u otro tipo de sensores capaces de capturar información de contexto [Pantsar-Syvaniemi et al., 2010]**.** El uso de estos elementos hardware tiene como objetivo establecer algún tipo de vinculación con los objetos físicos, leer un código de barras, transmitir un comando, etc. Son las propias aplicaciones las que rigen el intercambio de datos, procesan la información recibida de manera adecuada y ejecutan la lógica de negocio correspondiente. Cuando un usuario utiliza su Smartphone para interactuar con un gran número de objetos físicos de diferentes tipos, a menudo se ve obligado a instalar y utilizar multitud de aplicaciones software [Espada et al., 2011a]. Teniendo en cuenta la heterogeneidad que existe incluso entre muchos objetos físicos del mismo tipo (diferentes características, protocolos de comunicación, especificaciones técnicas, etc) el número de aplicaciones software que podría llegar a tener que manejar un usuario sería significativamente elevado [Perumal et al., 2008 ; Hyunho et al., 2011].

La necesidad de gestionar un número elevado de aplicaciones afecta negativamente a la experiencia del usuario, que debe buscar, descargar e instalar aplicaciones antes de iniciar cualquier interacción [Espada et al., 2011a]. Además este enfoque no contribuye en absoluto a potenciar el aspecto dinámico requerido en muchos sistemas, sistemas que basan su funcionalidad en interacciones puntuales con objetos físicos, las cuales es posible que un usuario no vuelva a repetir. El elevado número de aplicaciones software también puede favorecer la aparición de otros problemas tales como las incompatibilidades técnicas, consumo excesivo de los recursos del dispositivo, etc.

A la necesidad de desarrollar aplicaciones software ligadas al tipo de interacción deseada y a las características de los objetos físicos implicados se le debe sumar la heterogeneidad existente entre las plataformas móviles. Actualmente el mercado de los Smartphone está dividido en varias plataformas móviles: iOS, Android, WebOS, Windows Phone, Symbian, etc. Normalmente el desarrollo de aplicaciones software está parcialmente condicionado a la plataforma destino, es decir, para que una aplicación pueda llegar a un gran número de usuarios es muy posible que haya que implementarla varias veces, ya que cada plataforma suele utilizar su propio lenguaje de programación, especificaciones, APIS de acceso al sistema, etc [Espada et al., 2012].

La utilización de aplicaciones software nativas en los procesos de interacción entre Smartphones y objetos físicos produce varias consecuencias no deseadas se requiere una gran cantidad de aplicaciones software que deben de ser desarrolladas para diferentes plataformas, con los correspondientes costes que implica, se empeora la experiencia de usuario al obligarles a gestionar multitud de aplicaciones, las tareas de búsqueda, descarga

[4] https://market.android.com/details?id=com.samsung.remoteTV

e instalación de aplicaciones ralentiza la interacción con los objetos físicos en sistemas que pretenden un comportamiento dinámico o variable.

Motivados en parte por los problemas expuestos se han realizado numerosas propuestas tanto de carácter científico como comercial que tratan de minimizar el impacto de algunos de los aspectos negativos derivados de utilización de aplicaciones software nativas para la interacción con objetos físicos. Entre ellas existen formas muy diferentes de atajar varios de los problemas detectados. Numerosos frameworks [De & Moessner, 2009 ; Dey et al., 2001 ; Johnson, 2007] y plataformas [PhoneGap, 2011] ofrecen soporte al desarrollo de aplicaciones móviles multiplataforma, de esta forma consiguen simplificar los desarrollos y reducir los costes, aunque muchas de estas propuestas no cubren la totalidad de la funcionalidad requerida para implementar los mecanismos de interacción con los objetos físicos [Espada et al., 2012 ; Schmidt, 1999]. Para dotar del dinamismo requerido por algunos sistemas varios autores han realizado propuestas basadas en diferentes fundamentos, como: aplicaciones móviles fundamentadas en modelos basados en agentes (Agent-based model, ABM, multiagent systems) [Karnouskos & Tariq, 2009] o distintas clases de sistemas distribuidos [Weerawarana et al., 2005 ; Meng et al., 2011]. Varias de las propuestas analizadas resultan muy útiles para reducir el efecto de alguno de los problemas detectados, pero ninguna de ellas ofrece una solución global que combata de manera eficaz el conjunto de los problemas identificados.

Así pues, se trata de desarrollar una propuesta que proporcione una solución global y flexible al conjunto de problemas que afectan a los desarrolladores y usuarios de aplicaciones móviles que interactúan con objetos físicos. Desde el punto de vista de los desarrolladores la solución debería tener muy en cuenta los problemas derivados de la heterogeneidad entre plataformas móviles, permitiendo que un único desarrollo pueda ser utilizado en múltiples plataformas. Desde la perspectiva de los usuarios la propuesta debería ofrecer soluciones a las deficiencias derivadas de la necesidad de gestionar un gran número de aplicaciones y también a la falta de dinamismo en los procesos de interacción, entendiendo por falta de dinamismo la necesidad de realizar tareas repetitivas como la instalación y configuración de aplicaciones cuando se interactúa con un objeto por primera vez o cuando se pretende compartir una aplicación con otros usuarios.

1.3 HIPÓTESIS

Es posible solucionar los problemas derivados de la heterogeneidad entre plataformas móviles y la falta de adaptación a entornos dinámicos de los sistemas y aplicaciones móviles que basan una parte fundamental de su funcionamiento en la gestión de los elementos hardware de los teléfonos móviles inteligentes.

1.4 OBJETIVOS DE LA TESIS

Ante la situación descrita en el apartado anterior, se plantea como trabajo de tesis la propuesta de una especificación que permita el rápido desarrollo de aplicaciones aptas para ser ejecutadas en múltiples plataformas móviles. Estas aplicaciones deben soportar el acceso completo y de la forma más simple posible a los elementos hardware de los dispositivos móviles, especialmente a aquellos relacionados con la comunicación como: Bluetooth, NFC y a los que habilitan la captura de diferentes tipos de información de contexto como: cámara de fotos, sistemas de localización, micrófonos, etc. Además la especificación propuesta ha de ser lo suficientemente estructurada y flexible para que, en un futuro, resulte sencillo agregar mecanismos que habiliten la gestión de nuevos elementos hardware. Es muy posible, que las evoluciones técnicas en los dispositivos móviles ofrezcan próximamente nuevos elementos de comunicación y captura de información de contexto.

Las aplicaciones generadas a partir de esta propuesta deben favorecer la experiencia del usuario, sin requerir para su uso pesados procesos de instalación o configuración. Tampoco deben hacer caer sobre el usuario la carga de trabajo derivada de la gestión de demasiadas aplicaciones. Las características de estas aplicaciones tienen que contribuir a potenciar los aspectos dinámicos requeridos en muchos de los sistemas que basan parte de su funcionalidad en la interacción ocasional con objetos físicos. Por lo tanto las aplicaciones desarrolladas a partir de esta propuesta deben ofrecer un rápido acceso a los usuarios.

Este planteamiento puede enunciarse en base a los siguientes objetivos generales y específicos.

1.4.1 Objetivos generales

Para el desarrollo de esta tesis, se plantean los siguientes objetivos de índole general:

- Presentar un modelo que permita la construcción de manera simple y rápida de aplicaciones ligeras, multiplataforma y capaces de gestionar la mayoría de los elementos hardware de los dispositivos móviles, entre los que se encuentran los elementos que habilitan las comunicaciones con dispositivos electrónicos cercanos y la captura de la información de contexto.

- Favorecer el desarrollo de aplicaciones móviles dotadas de características óptimas para ser utilizadas en entornos que se basan en la interacción ocasional o puntual con objetos físicos. Se deben reducir al máximo los procesos de instalación y configuración, puesto que en estos procesos secundarios podrían llegar a consumirse más tiempo que en la propia ejecución del proceso de interacción para el que la aplicación fue diseñada.

- Garantizar la capacidad de adaptación del modelo de desarrollo de aplicaciones ligeras a futuras ampliaciones y modificaciones. Es predecible que en el futuro

aparezcan nuevos elementos hardware en los dispositivos que habiliten tanto nuevas formas de comunicación como de captura de información de contexto.

- Ofrecer los mecanismos de soporte necesarios para favorecer la aplicación del modelo propuesto en aplicaciones ubicuas de diversos tipos.

1.4.2 Objetivos específicos

Los objetivos enumerados anteriormente se desarrollan en los siguientes objetivos específicos:

- Realizar un análisis global de la situación actual de Internet de las cosas, abarcando el estudio de los distintos tipos de interacción entre objetos, especialmente de aquellos aspectos relacionados con la utilización de los teléfonos móviles inteligentes.

- Realizar un análisis de los sistemas y propuestas que permitan desarrollar aplicaciones móviles ligeras, las cuales puedan acceder a los distintos elementos hardware del teléfono.

- Desarrollar una especificación que permita crear aplicaciones móviles aptas para ser usadas en distintas plataformas móviles.

- Dotar a la especificación de mecanismos de acceso a los principales elementos hardware del dispositivo, como sensores, módulos de comunicaciones, etc.

- Dotar a la especificación de una arquitectura bien estructurada y modular que permita modificar o añadir mecanismos que habiliten la gestión de nuevos elementos hardware.

- Diseñar e incluir en la propuesta mecanismos simplificados de acceso a los elementos hardware del dispositivo.

- Dotar a la especificación de las características necesarias para poder generar aplicaciones bajo una arquitectura que favorezca la reducción del tiempo empleado en procesos secundarios, como configuraciones e instalaciones. Estas aplicaciones mejoraran la experiencia del usuario en entornos que se basan en la interacción ocasional o puntual con objetos físicos.

- Utilizar la especificación para desarrollar un conjunto de aplicaciones móviles, las cuales deben servir para ilustrar las características más destacadas de la propuesta.

- Comparar las aplicaciones desarrolladas siguiendo la especificación propuesta con otras aplicaciones de similares funcionalidades desarrolladas con otras

tecnologías, con el fin de analizar varias de sus características como el rendimiento técnico y la usabilidad.

- Validar que la especificación se adapte al cumplimiento de los objetivos propuestos.

1.4.3 Aportaciones y Beneficios

La especificación propuesta permite desarrollar aplicaciones móviles que basan una parte importante de su funcionalidad en la utilización de los elementos hardware del dispositivo, tales como: sensores de aceleración, cámaras de fotos, bluetooth, etc. La especificación abstrae los factores dependientes de las plataformas móviles específicas, posibilitando que las aplicaciones generadas puedan ser directamente utilizadas en distintas plataformas móviles, como: iOS, Android, WebOS, Windows Phone, Symbian, etc.

Tradicionalmente las aplicaciones móviles diseñadas para interactuar con objetos o capturar información de contexto, deben ser implementadas de forma parcial o completa para cada una de las plataformas móviles destino. La propuesta presentada en esta tesis introduce dos características que permitirán que muchos sistemas reduzcan significativamente sus costes de desarrollo. En primer lugar, la propuesta plantea la gestión de los elementos hardware del dispositivo mediante una especificación genérica totalmente independiente de la plataforma, por ello no será necesario dotar a los desarrolladores de los conocimientos técnicos específicos requeridos para gestionar los elementos hardware del dispositivo en cada plataforma: clases, métodos, APIs, etc. La propuesta utiliza una sintaxis abstracta y reducida que permite definir con pocas palabras procesos equivalentes a decenas o cientos de líneas de código nativo. En segundo lugar, las aplicaciones desarrolladas siguiendo la especificación son aptas para su funcionamiento en distintas plataformas móviles. La posibilidad de manejar una única implementación de la aplicación permite reducir los costes de desarrollo y mantenimiento, a la vez que aumenta el número de usuarios potenciales.

El factor dinámico de las aplicaciones que siguen la especificación propuesta elimina los procesos de instalación y hace énfasis en simplificar y agilizar tanto el acceso a las aplicaciones como su difusión. Gracias a estas características se libera a los usuarios de unos de los problemas tradicionalmente asociados al uso de las aplicaciones móviles nativas, la necesidad de gestionar (descargas en servidores y tiendas de aplicaciones, instalaciones, actualizaciones, etc) un elevado número de aplicaciones ya que la mayoría de los sistemas que interactúan con objetos físicos requieren el uso de una aplicación móvil específica.

La posibilidad de acceder y compartir las aplicaciones de forma rápida y simple satisface los requerimientos de los sistemas concebidos para que los usuarios realicen interacciones puntuales con objetos, interacciones que quizá el usuario nunca vuelva a repetir, como por ejemplo: acceso a los servicios ofrecidos por un quiosco electrónico en una ciudad inteligente, control remoto de los electrodomésticos de una habitación de hotel, etc. Las aplicaciones desarrolladas bajo la especificación proveen unos tiempos de

acceso en su primera utilización significativamente menores a los requeridos por las aplicaciones nativas.

1.5 ORGANIZACIÓN DEL DOCUMENTO

La memoria de esta tesis está estructurada en secciones y capítulos. En este apartado se presenta de forma resumida la organización del documento con el objetivo de facilitar su lectura.

En la **Parte I "Planteamiento del problema"** incluye los capítulos introductorios donde se presentan los problemas, la hipótesis y su contexto. En el capítulo 1, "**Introducción**", se muestra la justificación, motivación y planteamiento del problema. A continuación se presenta la hipótesis y se establecen los objetivos de la investigación, y el capítulo finaliza con la organización del documento. En el capítulo 2 "**Metodología y desarrollo de la investigación**" se presenta la metodología seguida durante el desarrollo de esta tesis.

En la **Parte II "Conceptos, soluciones y tecnologías existentes"** se presentan las tecnologías y conceptos sobre los que se fundamenta la investigación realizada en esta tesis doctoral. Los dos primeros capítulos de esta sección se centran en el estudio de tecnologías y conceptos existentes incluyendo "**Internet de las cosas**" (capítulo 3) y "**El teléfono móvil inteligente**" (capítulo 4). Los dos capítulos siguientes "**Sistemas de control remoto**" (capítulo 5) y "**Información de contexto**" (capítulo 6) se centran en el estudio de tecnologías y soluciones existentes en la actualidad y abordan algunos de los problemas planteados al inicio de esta memoria.

En la **Parte III "Desarrollo de la propuesta"** se presenta el modelo propuesto en esta tesis como solución de los problemas planteados al inicio del capítulo 1. El desarrollo de la solución propuesta se presenta en tres capítulos, cada uno de ellos corresponde con una interacción del sistema propuesto. El capítulo 7 "**Objetos virtuales v1.0 STDD**" describe la versión inicial de la solución, el capítulo 8 "**Objetos virtuales v2.0 distribuidos**" presenta la primera evolución del modelo. Finalmente, el capítulo 9 "**Objetos virtuales v3.0 En la web**" presenta la arquitectura, principios y elementos de la solución final propuesta.

En la **Parte IV "Desarrollo de prototipos y evaluación"** se presentan varios de los prototipos que han sido construidos durante el desarrollo de esta tesis doctoral. En el capítulo 10 "**Desarrollo de prototipos**" se detalla el proceso de desarrollo y funcionalidad de tres prototipos, estos prototipos abordan diversos aspectos de la propuesta. En el capítulo 11 "**Evaluación**" se presentan los resultados de los cuatro tipos de evaluación a la que fue sometida la solución propuesta.

En la **Parte V "Conclusiones"** se presentan las conclusiones generales que pueden extraerse de esta investigación (capítulo 12). Estas conclusiones incluyen una síntesis del trabajo propuesto, la verificación del cumplimiento de los objetivos planteados, un listado de las publicaciones derivadas y las aportaciones más significativas de esta tesis. Posteriormente se presentan las líneas futuras de investigación derivadas del trabajo realizado.

CAPÍTULO 2

METODOLOGÍA Y DESARROLLO DE LA INVESTIGACIÓN

2.1 INTRODUCCIÓN

En este capítulo se presentará la metodología utilizada para el desarrollo de esta tesis doctoral. A continuación se muestra el desarrollo temporal de la investigación, exponiendo los principales hitos en orden cronológico.

2.2 METODOLOGÍA DE TRABAJO

El trabajo de investigación ha sido desarrollado en varias etapas y se ha utilizando un enfoque iterativo e incremental. En la (Fig.2.1) se muestra el marco metodológico de la investigación con referencias temporales.

El punto inicial de la investigación coincide con el inicio del trabajo fin de máster en la especialización de investigación de la titulación Máster y Doctorado en Ingeniería Web , realizado en la Escuela de Ingeniería Informática de Oviedo (Campus de los Catalanes), Universidad de Oviedo. A partir de los conocimientos adquiridos durante los cursos académicos 2008/2009 y 2009/2010 se presentó una primera propuesta de estructura para los ***objetos virtuales colaborativos basados en servicios***, la especificación propuesta se utilizo para implementar un sistema de gestión de tickets de cine. Las experiencias de dicho proyecto sirvieron para obtener diversos puntos de vista que permitieron expandir la investigación hacia otras áreas y mejorar la arquitectura inicial. Las carencias detectadas, tanto en la propuesta inicial, como en los sistemas relacionados propuestos por otros autores se tomaron como motivación para la presente tesis doctoral.

Con una línea de investigación marcada, se comenzó el trabajo de investigación, que consistió en profundizar en los problemas detectados mediante la realización de un profundo estudio del estado del arte y el planteamiento de una línea a seguir para cumplir con los objetivos marcados, siendo dicho trabajo una base sólida para la realización de la investigación aquí presentada. A partir del estado del arte realizado y de las conclusiones obtenidas, se inició el desarrollo de esta Tesis Doctoral, estableciendo de forma precisa el problema a resolver y los objetivos e hipótesis de partida.

Para llevar a cabo la investigación se ha optado por un enfoque iterativo e incremental. Así en cada interacción, se desarrollaron prototipos y se modificó el modelo propuesto hasta llegar al finalmente expuesto. Se han realizado varias publicaciones de reconocido prestigio con el objetivo de obtener una valiosa retroalimentación por parte de la comunidad científica, que han servido para ayudar a desarrollar este trabajo en la línea adecuada.

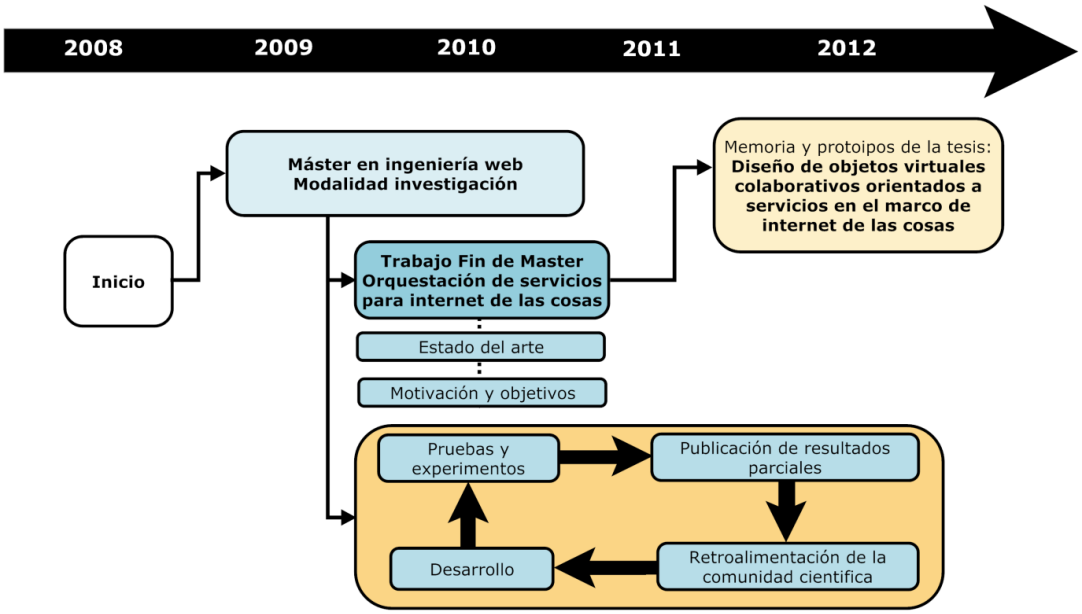

Figura 2.1 – Metodología utilizada en la investigación.

2.2.1 Ciclo de realimentación

El enfoque iterativo e incremental se basa en la iteración deductiva-inductiva, entendiéndose como un proceso de retroalimentación. Dicho proceso, conocido desde Aristóteles, es parte de nuestra experiencia diaria. Así, una idea inicial (o hipótesis) conduce, mediante un proceso de deducción, a ciertas consecuencias que se pueden comparar con los datos disponibles. Cuando las consecuencias y los datos no concuerdan, se puede llegar, por un proceso de inducción, a la modificación de la hipótesis, iniciándose un nuevo ciclo de iteración. En la (Fig.2.2) se reflejan estas ideas. Cabe destacar que la adquisición de datos durante esta tesis viene fundamentalmente reflejada en las referencias finales del documento y en los enlaces expuestos a lo largo del documento.

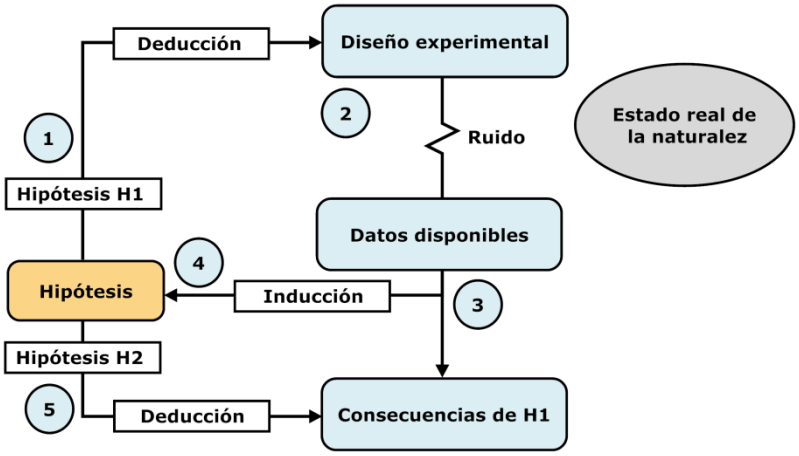

Figura 2.2 – Proceso iterativo de aprendizaje

2.3 DESARROLLO TEMPORAL DE LA INVESTIGACIÓN

A continuación, se presentan algunos de los hitos más significativos llevados a cabo durante la realización de esta investigación.

- **Septiembre 2008**. Inicio de máster en ingeniería web. Durante el primer año se cursan asignaturas de carácter técnico que confieren conocimientos sobre tecnologías que posteriormente serán utilizadas para el desarrollo y diseño de prototipos y sistemas contenidos en la presente tesis.
- **Septiembre 2009**. Durante el segundo año del máster en ingeniería web se cursan varias asignaturas pertenecientes a la rama de investigación: *Diseño y Construcción de Model Driven Architecture*, *Maquinas virtuales y reflectividad y Nuevos estándares en la web*. En estas materias se adquieren los conocimientos necesarios para comenzar a realizar una investigación científica de calidad.
- **Septiembre 2009**. Comienzo del trabajo fin de máster. El objetivo es analizar sistemas relacionados y sentar las bases y objetivos de la investigación para desarrollar una primera versión del modelo de los objetos virtuales colaborativos.
- **Mayo 2010**. Enviado el capítulo Standarization of virtual objects para su publicación en el libro Semantic Web Personalization And Context Awarenes editado por IGI-Global. En este capítulo se expone la situación actual de la investigación y los primeros resultados obtenidos.
- **Julio 2010**. Lectura del trabajo fin de máster en la modalidad de investigación Orquestación de servicios para Internet de las cosas con calificación de Matrícula de Honor. Este trabajo de investigación contiene el estado del arte, la definición de los objetivos y la especificación de una primera versión de la arquitectura propuesta, el trabajo se apoya en un prototipo funcional.
- **Mayo 2011**. El artículo Modelling architecture for collaborative virtual objects based on services publicado en la revista Journal of Network and Computer Applications, **con un último factor de impacto en el índice Journal Citation Reports de 0.660**. Este trabajo presenta una nueva versión del modelo con ciertas modificaciones en la arquitectura de los objetos virtuales colaborativos,-surgidas a raíz de las observaciones y análisis realizados por otros investigadores ajenos al proyecto. En este artículo se presentan varias evaluaciones de un prototipo que es comparado con varias aplicaciones móviles de carácter general.
- **Junio 2011**. Presentación del artículo Internet de las Cosas Cuando los objetos hablen entre sí al Certamen FECYT de Comunicación Científica. FECYT. Fundación Española para la Ciencia y la Tecnología, Ministerio de economía y competitividad. El artículo recoge varias ideas y perspectivas de futuro relacionadas el con Internet de las cosas.
- **Junio 2011**. El libro Internet de los objetos, libro de divulgación científica publicado a nivel nacional por la editorial Netbiblo donde se recogen y explican varias tendencias y conceptos ligados a Internet de las cosas.
- **Septiembre 2011** El artículo Using collaborative virtual objects based on services to communicate smart objects es enviado para su posible publicación a la revista **IET Software, con un último factor de impacto en el índice Journal Citation Reports de 0.671**. El trabajo analiza los beneficios obtenidos frente a otras alternativas del uso de objetos virtuales colaborativos basados en servicios como

componentes software que actúan como controladores remotos de objetos inteligentes físicos.

- **Octubre 2011.** El artículo Virtual Objects on the Internet of things es aceptado para su publicación en la revista International Journal of Interactive Multimedia and Artificial Intelligence newsletter. En el artículo se hace una revisión del estado del arte de los objetos virtuales en el marco de Internet de las cosas, se señalan las deficiencias de los sistemas actuales y se presentan e ilustran cómo los objetos virtuales normalizados pueden resultar más eficientes en determinados escenarios.
- **Diciembre 2011** El artículo A lightweight approach to managing real world information in mobile web applications es enviado para su posible publicación a la revista IEEE Pervasive Computing, **con un último factor de impacto en el índice Journal Citation Reports de 2.189**. En este trabajo se indaga en los beneficios y nuevas posibilidades derivadas del uso de la información de contexto en las aplicaciones distribuidas. En este artículo se presenta un lenguaje útil para expresar requerimientos de información de contexto aplicable tanto en la definición de interfaces web como en la de interfaces de los objetos virtuales colaborativos distribuidos. En él se presentan aplicaciones distribuidas que poseen funcionalidades que hasta ahora eran exclusivas de las aplicaciones nativas.
- **Diciembre 2011**. Se presenta el poster Colaborative virtual objects based on services within the Internet of things, en las primeras jornadas de investigación de la Universidad de Oviedo. En él se expone el contenido más significativo de la investigación que ha dado lugar a esta tesis doctoral.
- **Enero 2012** El artículo A simple model based on web services to exchange context information between web browsers and web applications es enviado para su posible publicación a la revista Journal of Universal Computer Science, **con un último factor de impacto en el índice Journal Citation Reports de 0,669**. En este artículo se presenta un nuevo enfoque para el intercambio de información de contexto entre navegadores y aplicaciones distribuidas, el sistema está basado en servicios web y obtiene una mejor eficiencia que las otras alternativas analizadas.
- **Febrero 2012** Aceptado el capítulo Using mobile web applications as a multiplatform mechanism to communicate users with embedded devices para su publicación en el libro Embedded Systems: Theory, Applications and Role in Quantum Mechanics editado por Nova publishers. En este capítulo se realizan varios análisis sobre las distintas formas de interacción con sistemas embebidos a través de los teléfonos móviles. En este trabajo se identifican varios de los problemas y debilidades más recurrentes en los distintos sistemas, los datos obtenidos se utiliza para adaptar en consecuencia la investigación de la presente tesis.
- **Febrero 2012** Aceptado el artículo Extensible architecture for context-aware mobile web applications para su publicación a la revista Expert System with Applications, **con un último factor de impacto en el índice Journal Citation Reports de 1.924**. En este artículo se propone una primera versión de un navegador web móvil sensible al contexto. El navegador integra parte de la lógica de los objetos virtuales colaborativos para permitir a las aplicaciones web acceder a algunos tipos de información de contexto.

- **Abril 2012** El artículo Mobile Web-based system for remote control electronic devices and smart objects es enviado para su posible publicación a la revista IEEE Internet Computing, **con un último factor de impacto en el índice Journal Citation Reports de 2.514**. En este artículo se presenta una arquitectura que permite a las aplicaciones distribuidas especificar procesos de comunicación básicos de manera sencilla, estos procesos se apoyan en los elementos hardware de comunicación de los dispositivos móviles como Wi-Fi o Bluetooth. En el trabajo se muestra un prototipo web capaz de controlar un centro multimedia vía Wi-Fi

PARTE II

CONCEPTOS, SOLUCIONES Y TECNOLOGÍAS EXISTENTES

CAPÍTULO 3

INTERNET DE LAS COSAS

3.1 INTRODUCCIÓN

Internet nació conectando personas a través de máquinas. Ahora una parte importante de su red, conecta máquinas que hablan entre ellas para cumplir una tarea sin necesitar al hombre. El siguiente paso es la llamada Internet de las cosas.

El concepto de Internet de las cosas (Internet of things) se atribuye originalmente al Auto-ID Center, fundado en 1999 y basado en el Instituto Tecnológico de Massachusetts. La idea es muy sencilla, parte de que cualquier "cosa", es decir objeto convenientemente etiquetado pueda ser capaz de comunicarse con otros objetos y personas a través de Internet, redes privadas u otros protocolos [Kortuem et al., 2010; Thompson, 2005]. Los objetos que formarían parte de la red pueden ser de lo más variado, desde maquinaria industrial, electrodomésticos, vehículos, a productos del supermercado. Dependiendo del tipo de objeto contendrán pequeños chips, sensores o mini computadores [Kranz et al., 2010].

Las posibilidades que ofrece Internet de las cosas para facilitar la vida de las personas y automatizar muchas de las tareas que realizamos actualmente son enormes, por ejemplo, es posible que el frigorífico envié un mensaje al teléfono si se acaba la leche, monitorizar a pacientes clínicos vía Internet y una multitud de aplicaciones prácticas que en la actualidad empiezan a ver la luz y que en breve impactarán de lleno en nuestra vida cotidiana.

Aunque la idea fundamental es simple, su aplicación resulta difícil. Con el paso de los años, la comunicación entre objetos ha evolucionado, se han sumando nuevas técnicas y mejorando las ya existentes que comprenden desde etiquetar objetos a través de tags RFID a añadir microprocesadores para dotarlos de más inteligencia y capacidad de reacción. No se trata sólo de conseguir que los objetos hablen, sino de que interactúen entre si y dotarlos de inteligencia.

Investigadores y grandes compañías coinciden en destacar la importancia de Internet de las cosas. Esta web al cuadrado aumentará exponencialmente los nuevos datos y mediciones de la realidad, ofreciendo una enorme cantidad de nuevas funcionalidades que ayudaran a crear una sociedad más eficiente e innovadora.

3.2 PRECURSORES

El concepto de Internet de las cosas proviene principalmente de dos fuentes: el MIT Auto-ID Center y la International Telecommunications Union.

3.2.1 MIT Auto-ID Center

El término "Internet de las cosas" se acuñó hace unos 10 años por los fundadores del MIT Auto-ID Center, con mención especial a Kevin Ashton en 1999[5] y David L. Brock en 2001[6]. El aumento de la reputación del Auto-ID Center se produjo en septiembre de 2003, cuando se realizó el lanzamiento oficial de la Red EPC (Electronic Product Code) - una infraestructura de tecnología abierta que permite identificar automáticamente objetos y rastrearlos. El simposio contó en aquel entonces con el apoyo de más de 90 grandes empresas de todo el mundo que abarcaban un gran número de sectores: alimentación, bienes de consumo, transporte y laboratorios farmacéuticos, entre otros. A partir de ese momento comenzó a considerarse al RFID (Radio-frequency identification) como una tecnología clave para el crecimiento económico en los próximos años.

El término "Auto-ID" hace referencia a cualquier clase de tecnología de identificación utilizada en la industria para automatizar sistemas, reducir errores y aumentar la eficiencia. Las tecnologías agrupadas en el Auto-ID incluyen entre otras: códigos de barras, tarjetas inteligentes, sensores, reconocimiento de voz y datos biométricos. El objetivo de los laboratorios Auto-ID es desarrollar una red que conecte equipos y objetos. El desarrollo de esta red incluye: software, hardware, protocolos y lenguajes para la descripción de objetos. El Auto-ID Labs no pretende crear otra red global similar a Internet sino desarrollar elementos que trabajen sobre Internet [Sundmaeker et al., 2010].

3.2.2 International Telecommunications Union

El concepto de "Internet de las cosas" entró en foco de atención mundial en 2005 cuando la International Telecommunications Union publicó el primer informe sobre el tema[7] . El informe adopta un enfoque global y sugería que, en un futuro, Internet de las cosas conectaría objetos repartidos por todo mundo, tanto de una forma sensorial como inteligente a través de la combinación de los avances tecnológicos en la identificación de objetos, redes de sensores, sistemas integrados y la nanotecnología [ITU, 2005].

[5] I could be wrong, but I'm fairly sure the phrase 'Internet of Things' started life as the title of a presentation Imade at Procter & Gamble (P&G) in 1999, Kevin Ashton, RFID Journal, 22 June 2009
[6] David L. Brock, MIT Auto-ID Center, MIT-AUTOID-WH-002, "The Electronic Product Code", January 2001
[7] The Internet of Things", ITU, November 2005

3.3 INTERNET / INTERNET DE LAS COSAS

Internet de las cosas tiene como ideal que prácticamente todas las "cosas" que hay en este mundo físico pueda convertirse en objetos conectados a Internet u otras redes [ITU, 2005]. Uno podría preguntarse si realmente todas las cosas necesitan disponer de un mini computador que las mantenga conectadas al actual Internet, la respuesta es no. Por ejemplo, un bien de consumo como un yogurt se podría considerar integrado en Internet de las cosas cuando se encuentra etiquetado con una etiqueta RFID que permita a otros dispositivos obtener información acerca del estado producto, composición, ciclo de vida, etc. [Meyer et al., 2009].

Existen otro tipo de objetos que poseen una elevada capacidad de comunicación con su entorno o con otras aplicaciones. Este tipo de objetos sí suelen contar con un mini computador y una conexión a la red. Ejemplos de estos objetos son las redes de sensores que pueden ser accedidas desde cualquier lugar del mundo.

3.2.1. Tipos de comunicación

Los analistas suelen describir dos modos básicos de comunicación en Internet de las cosas: cosa-persona y cosa-cosa [Global, 2008].

- **Cosa – persona:** Las comunicaciones de este tipo abarcan una serie tecnologías y aplicaciones en las cuales las personas interactúan con cosas y viceversa. Las más comunes son el acceso a distancia, control remoto y monitorización. También existen cosas que informan a las personas de cambios en su estado, datos recogidos, etc. Los objetos con ese comportamiento suelen denominarse "Blogjects", esta palabra es un neologismo creado por Julian Bleecker para referirse a objetos que comunican sus experiencias.

- **Cosa – Cosa:** Abarca tecnologías y aplicaciones en donde objetos interactúan sin que ningún humano haya iniciado la interacción ni sea receptor o intermediario. Los objetos pueden controlar otros objetos, tomar medidas correctivas y realizar notificaciones a las personas según sea necesario.

3.4 CAMPOS DE APLICACIÓN

Éstas son algunas de las tendencias más populares dentro de Internet de las cosas [Global, 2005]:

- Los teléfonos móviles como "**ventanas de las cosas cotidianas**". Los dispositivos móviles como teléfonos y PDAs pueden mostrar información acerca de los objetos etiquetados con códigos de barras y etiquetas RFID, gracias a la incorporación de lectores y cámaras. Estos dispositivos pueden conectarse con los servidores vía Internet u otros protocolos para identificar: personas, lugares y objetos. Así, un teléfono podría mostrar detalles sobre un producto identificado: atributos, origen, precio, garantía, opiniones, su manual de usuario, dónde comprarlo, cómo reciclar, etc.

- Los teléfonos móviles como "**los mandos a distancia del entorno**". Entre los usos comunes de los teléfonos móviles en países como Japón ya se encuentran tareas como la realización de pagos y el uso como mandos a distancia para equipos multimedia. Los teléfonos pueden convertirse también en un medio para controlar cualquier dispositivo cercano o lejano como pueden ser cerraduras, sistemas de seguridad, luces, etc.

- **La monitorización continua y la medición**. A través de sensores resulta posible identificar, almacenar información y monitorizar características de personas u objetos. Esta monitorización hace posible automatizar labores de supervisión, gestión e inventariado, etc. Algunos productos alimenticios llevan chips que contienen información acerca de su identidad, elaboración, lugares en que ha sido almacenado incluso sensores de frescura. Estos chips pueden ser leídos automáticamente por dispositivos a los que les transmiten su información útil, lo que facilita enormemente las labores de transporte, gestión e inventariado.

- **Localización de las cosas**. La capacidad de localización en lugares cerrados y otros avances tecnológicos como la miniaturización prometen aumentar la variedad de objetos que pueden informar de su localización, incluyendo: llaves, billeteras, gafas, joyas, y herramientas. Esta evolución de la capacidad de localizar objetos podría conducir a cambios en los paradigmas de presentación y almacenamiento.

- **Pronósticos para el mantenimiento de vehículos y maquinaria**. El continuo seguimiento permite establecer un nuevo paradigma en el mantenimiento de vehículos y maquinaria industrial. En lugar de llevar a cabo el mantenimiento en los intervalos especificados, las organizaciones pueden realizar el mantenimiento cuando sea necesario. Los sensores puede advertir a los técnicos cuando hay que realizar alguna revisión y, a su vez, micrófonos incorporados cerca de los puntos clave de las máquinas pueden detectar sonidos que indican un desgaste excesivo. Los sensores de lecturas pueden combinarse con los registros de servicios, creando bases de datos de predicción de mantenimiento; y a partir de estos datos deducir algoritmos que podrían prever problemas de fiabilidad y reducir el costo de mantenimiento. Por lo tanto, vehículos, generadores eléctricos, equipo industrial, y demás tipos de maquinaria serán pronto candidatos a formar parte de Internet de las cosas.

- **Servicios de salud y asistencia**. Sensores que monitorizan la actividad de los pacientes, incluso cuando estos se encuentran fuera del hospital. Los sensores en camas, suelos y cañerías pueden ser de gran ayuda para cuidar a las personas. La universidad de Virginia desarrolla el proyecto AlarmNet, consistente en la interconexión de redes con algunos objetos cotidianos como camas y suelos. Un sensor de presión en una cama detecta la frecuencia cardíaca, la respiración y el movimiento de la persona. Sensores en el suelo pueden detectar cuando una persona se cae y comunicárselo al hospital.

- **Control de elementos**. En cualquier momento se podría acceder a los servicios ofrecidos por los objetos conectados a la red. Por ejemplo, desde el supermercado se podría preguntar al frigorífico inteligente que está conectado a la red si quedan existencias de un determinado producto.

- **Automatizar el control de entornos abiertos con una gran incertidumbre**. La comunicación directa entre objetos puede servir para reducir los efectos de problemas inesperados. Por ejemplo, si un coche detectase que ha sufrido un accidente podría comunicárselo a las señales de tráfico cercanas, haciendo que éstas se auto regulen, reduciendo la velocidad permitida y advirtiendo a los conductores cercanos del peligro [Carabelea & Boissier, 2003].

3.5 TECNOLOGÍAS

3.5.1 Introducción

Internet de las cosas es una revolución tecnológica que posiblemente marcará el futuro de la informática y las comunicaciones. El desarrollo técnico de Internet de las cosas depende de una combinación en el avance de varios desarrollos:

1. El primer tipo de tecnologías tiene el fin de **identificar** a los objetos y dispositivos. Un sistema discreto y rentable de identificación de objetos es crucial.

2. En segundo lugar se encuentra la **recopilación de datos,** encargada de desarrollar la capacidad que permita detectar cambios en el estado de las cosas mediante tecnologías como los sensores.

3. **Inteligencia integrada** en las cosas. Con el beneficio del tratamiento integrado de información, elementos industriales y objetos cotidianos adquieren características inteligentes y nuevas capacidades.

4. Por último, los avances en la **miniaturización y la nanotecnología**. Esto significa que cosas cada vez más pequeñas tendrán también la capacidad de comunicarse con otras cosas y conectarse a redes.

3.5.2 Identificación

Ya sea en el mundo físico, dentro de una red privada o en Internet es necesario establecer mecanismos para que los objetos puedan ser identificados. La identificación tiene una especial importancia ya que personas, aplicaciones y otros dispositivos han de poder establecer contacto con los objetos. Esta tarea no será posible si no se localiza un objeto concreto o no se puede hacer referencia a él.

Los objetos pueden ser localizados o identificados a través de varios mecanismos. Algunos de los más populares son:

1. Mediante los identificadores.

2. Por la ubicación.

3. Utilizando el perfil de un objeto o elemento en combinación con una ubicación.

Existen varias tendencias para establecer los identificadores de objetos. Estas tendencias dependen en gran medida del tipo de red en la que se trabaje. Una técnica extendida consiste en utilizar cadenas miembro de espacios de nombres globales o locales como identificadores de objetos. [Grønbæk, 2008].

Muy ligado a la identificación se encuentran los perfiles de objetos. Estos perfiles se utilizan para describir los objetos y sus capacidades, por ejemplo, características funcionales que incluyen la pila de protocolos, perfil requerido para acceso, etc.

3.5.2.1 Servicio de nombres de objetos

El Servicio de nombres de objetos ONS (Object Naming Service) es parte de la Red EPC Global. La oficina de estadísticas ofrece resolución de nombres para Electronic Product Codes (EPC) [Armenio et al., 2007] es decir, la dirección de la interfaz para el propietario del EPC [EPCglobal , 2008].

3.5.2.2 RFID: Radio Frequency IDentification

Dentro del área de la identificación de objetos que no se encuentran conectados a Internet, la identificación por radiofrecuencia RFID es la tecnología más difundida y aceptada. RFID es un sistema de almacenamiento y recuperación de datos remoto capaz de ser leído y escrito sin contacto físico, vía ondas de radio usando antenas. Los identificadores y/o información relativa al objeto se inscriben en el chip RFID, el cual podrá ser leído por otros objetos.

Un sistema RFID consta de los siguientes componentes:

- **Etiqueta RFID:** compuesto por una antena, un transductor radio y un material encapsulado o chip. El propósito de la antena es permitirle al chip transmitir la información de identificación de la etiqueta. Existen etiquetas de dos tipos:
    - De sólo lectura.
    - De lectura y escritura: la información de identificación puede ser modificada por el lector.
- **Lector de RFID:** capta la señal de una etiqueta RFID, extrae la información y se la suministra al Middleware.
- **Middleware RFID:** proporciona los medios de proceso y almacenamiento de datos.

Las razones del crecimiento del uso de esta tecnología han sido el abaratamiento de los costes y el establecimiento de estándares ampliamente aceptados. Entre los más influyentes se encuentran el "EPC Network", una colección de hardware y estándares software para el uso de RFID en diversas industrias. El EPC Electronic Product Code está considerado como la siguiente generación de códigos de barras.

El RFID cuenta en la actualidad con una amplia aceptación, cada vez es más común ver este sistema implantado en distintos objetos cotidianos. Los teléfonos móviles están comenzando a incluir lectores RFID debido al tremendo éxito de esta tecnología y sus potenciales usos en el día a día. Las primeras aplicaciones de RFID incluyen, entre otras, la gestión en las cadenas de suministros y la prevención de la falsificación en la industria farmacéutica. Las etiquetas RFID pueden incluso implantase debajo de la piel en humanos para uso médico o realizar identificaciones de personal[8].

[8] http://www.wired.com/techbiz/it/magazine/17-03/st_best

3.5.2.3 Recopilación de datos

La capacidad de detectar cambios en el estado físico de las cosas es también esencial para el registro de los cambios en el ambiente. En este sentido, los sensores desempeñan un papel fundamental en la reducción de la brecha entre lo físico y lo virtual ya que hacen posible que las cosas puedan percibir los cambios en su entorno físico. Los sensores recogen los datos de su entorno y toda esta información genera concienciación sobre el contexto y puede ser aplicada en la toma de decisiones.

En la actualidad el número de campos en los que se aplica el uso de sensores ha crecido rápidamente. Por ejemplo, la incorporación de sensores en la ropa conocida como ropa inteligente. Varias de estas prendas se utilizan ya en la actualidad, iGarment[9] desarrolla ropa inteligente para bomberos y personal de rescate, una red de sensores distribuidos por las prendas controlan diversas señales vitales y factores externos, la información recogida es enviada a un controlador para evitar situaciones que puedan poner en peligro a la persona.

3.5.3 Inteligencia integrada

La Inteligencia integrada en las propias cosas supone distribuir la capacidad de procesamiento por toda la red, este enfoque ofrece nuevas posibilidades para el procesamiento de datos. La inteligencia integrada puede capacitar a las cosas para tomar decisiones de manera autónoma o colaborativa con otras cosas o personas. El término “objeto inteligente" (Smart Thing) se utiliza a menudo cuando se está hablando de Internet de las cosas pero es difícil definirlo de manera concreta. Un objeto inteligente implica que una cosa posea, al menos, cierta capacidad de procesamiento y/o de reacción ante eventos o señales externas. Los avances en las casas inteligentes, vehículos y la robótica son algunos de los mejores ejemplos de elementos que encajarían dentro del concepto de objeto inteligente. La evolución de los sistemas embebidos es rápida, por ello los fabricantes están utilizando estos sistemas para dotar de una mayor inteligencia a toda una gama de nuevos productos y electrodomésticos.

3.5.3.1 Objetos inteligentes

Con estos sistemas se pretende dotar de nuevas capacidades de interacción a objetos cotidianos o de uso industrial. Se trata de crear un vínculo entre los objetos físicos mediante la incorporación de un computador y de mecanismos que capaciten la comunicación.

Entre las capacidades con las que suelen contar estos elementos se encuentran la recolección de datos (sensores), la toma de decisiones y la interacción con personas u objetos. En ocasiones, a estos sistemas también se les denomina Networked embedded Devices (NEDs) o Smart objects /devices [Carabelea & Boissier, 2003]. La característica principal de estos objetos es que tienen embebidos dispositivos electrónicos con capacidad de computación y comunicación. Dichas características pueden convertir objetos cotidianos en elementos inteligentes, automatizando la toma de decisiones y

[9] http://www.netgarment.com/content/

ampliando los sistemas de información con la re-localización de la lógica de negocio, moviendo la ejecución de ciertos procesos hasta los propios elementos. Como resultado, el elemento puede operar como mayor autonomía, incluso de forma colaborativa con otros elementos de la red [Karnouskos, 2007].

Los sistemas embebidos se utilizan cada vez de una manera más generalizada. Están siendo incluidos en una amplia variedad de objetos como: equipos electrónicos, vehículos, maquinaria industrial, dispositivos médicos, etc.

Los actuales avances en la miniaturización, micro-sistemas y nano electrónica junto al creciente suministro de herramientas de desarrollo middleware hacen que estos sistemas se integren cada vez más en muchos segmentos de la sociedad. Aunque algunos objetos cuenten ya con capacidad de computación y comunicación el siguiente paso consiste en hacer que se cooperen con otros objetos y aplicaciones [ITU, 2005].

3.5.3 Miniaturización y nanotecnología

Se llama nanotecnología al conjunto de tecnologías y aplicaciones que usan el comportamiento peculiar que tiene la materia cuando se encuentra en estructuras muy pequeñas, a una escala menor que un micrómetro, es decir, a nivel de átomos y moléculas. A estas escalas, las leyes clásicas de la física ceden el paso a las de la física cuántica y se pueden obtener nuevos materiales con propiedades muy diferentes de las que presentan en su forma "normal" macroscópica. Así, los cambios producidos afectan a características como por ejemplo: el color, la conductividad, la reactividad y la resistencia. Estos cambios, lejos de ser futurología, ya tienen aplicaciones generales en prácticamente todos los campos. Se prevé que en unos años, los avances en el campo de la nanotecnología harán que las computadoras y demás dispositivos electrónicos dejen de utilizar el silicio como sistema para integrar los transistores y empiecen a manejarse con lo que se llama mecánica cuántica, lo que hará que utilicen transistores a escala atómica. Algunos gigantes del mundo informático como IBM, NEC, Intel y Hewlett-Packard están invirtiendo millones de dólares al año en tecnologías relacionadas con la miniaturización y la nanotecnología.

A medida que los avances técnicos permitan que los sensores y computadores tengan tamaños más reducidos, menor consumo y mejor eficiencia resultara más sencillo embeberlos en un mayor abanico de objetos físicos.

3.6 INTELIGENCIA DE LAS COSAS

Los sistemas que forman parte de Internet de las cosas pueden ser clasificados de diversas maneras, una de ellas es analizando su inteligencia. Principalmente son dos cualidades las que definen la inteligencia de un objeto, la primera es el nivel de inteligencia y la segunda la localización de la misma.

3.6.1 Nivel de inteligencia

El nivel intelectual de un objeto puede variar en gran medida dependiendo de su objetivo o el grado de integración que tenga dentro del sistema. Podríamos establecer una clasificación del nivel de inteligencia de las cosas evaluando tres categorías [Meyer et al., 2009].

- **Manejo de la información.** Un producto inteligente debería, al menos, ser capaz de gestionar su propia información, que podría venir dada por sensores, etiquetas RFID u otras tecnologías. Sin esta capacidad, difícilmente un objeto puede ser llamado inteligente. Se dice que un producto sólo posee la capacidad de "manejo de información" cuando maneja información pero no tiene el control de su propia vida y, además, este control se realiza de forma externa. Un ejemplo de este tipo de inteligencia sería cualquier producto del supermercado etiquetado mediante RFID.

- **Notificación**. Un mayor nivel de inteligencia implica que una cosa puede notificar a otros dispositivos o personas que ha detectado determinados eventos y/o problemas, por ejemplo: que se ha caído, que la temperatura es demasiado alta, etc. El objeto no posee el control de su actividad pero sí es capaz de informar cuándo se producen ciertas situaciones.

- **Toma de decisiones**. Los objetos con mayor nivel de inteligencia son aquellos que pueden auto-administrarse, estos objetos son capaces de tomar decisiones en momentos concretos por sí mismos, con o sin intervención externa dependiendo del contexto. En este caso, el producto podría tener un control total sobre sí mismo sin requerir la colaboración de personas [Kärkkäinen et al, 2003].

3.6.1 Ubicación de la inteligencia

En el contexto de Internet de las cosas, cuando, refiriéndonos a un objeto, decimos que posee inteligencia no significa necesariamente que la inteligencia se encuentre en él. Existen dos tendencias claramente diferenciadas a la hora de ubicar la inteligencia [Meyer et al, 2009].

- **Inteligencia a través de la red**. La inteligencia se encuentra fuera del producto físico. Por ejemplo, es un caso muy común que el objeto esté vinculado a un servidor en el que un agente dedicado se encarga de gestionar su funcionamiento basándose en la información obtenida a través del objeto físico o algún patrón de comportamiento [Främling et al., 2003]. En este tipo de sistemas la inteligencia del objeto se ejecuta en hosts que suelen denominarse plataformas de portal [Ramparany & Boissier, 2002].

- **Inteligencia en el objeto**. En este caso el objeto no solo se encarga de recopilar y tratar la información sino que, además, es el encargado de realizar la toma de decisiones. Para poseer inteligencia un objeto debe contar necesariamente con cierta capacidad de computación y, muy probablemente, necesite también capacidad de almacenamiento. En este caso el objeto es el encargado ejecutar su inteligencia [Brock, 2001].

No significa que estos dos enfoques tengan que aplicarse siempre de manera excluyente sino que pueden aparecer soluciones intermedias. Por ejemplo, el proyecto Collaborative Business items (CoBis) [Collaborative, 2007 ; Decker et al., 2006] combina la inteligencia en los propios objetos con la inteligencia a través de la red. Las redes de CoBis están formadas por dispositivos inteligentes que intercambian información entre sí. Esta información es analizada por cada dispositivo para realizar la toma de decisiones. Una posible aplicación de los CoBis es la de servir como controladores de seguridad en el almacenaje de productos químicos. En este caso los "bidones inteligentes" intercambian información con otros bidones cercanos basándose en su contenido, temperatura, etc. Con los datos intercambiados y la inteligencia de cada bidón el sistema es capaz de establecer si hay una incompatibilidad de productos o si se está generando una situación potencialmente peligrosa en el almacén.

3.7 ESTÁNDARES

A diferencia de Internet, actualmente no existe ningún conjunto único de normas mundiales para Internet de las cosas, y según los expertos es probable que nunca lo haya. La razón más importante es que Internet de las cosas se sale de la limpieza del mundo digital cerrado, lógicamente consistente y autosuficiente. En muchos de los proyectos englobados en Internet de las cosas tienen gran peso propiedades físicas como la distancia y las características del contexto, materiales, etc. Estas propiedades dependen de factores casi tan variados como el mundo físico en sí [Fleisch, 2010].

Por otra parte, sí que existen varias normas y estándares con un gran peso y aceptación que regulan algunos sectores relacionados con Internet de las cosas. Por ejemplo, debido principalmente a los mandatos de Walmart Metro y otros grandes minoristas, el estándar Electronic Product Code global (EPCglobal) se convirtió en la pila estándar de facto en el comercio minorista y de la industrias de bienes de consumo [Thiesse et al., 2009] . Los minoristas no sólo venden bienes de consumo, por lo que el estándar EPC continúa expandiéndose a otros sectores industriales, como el textil o productos farmacéuticos, etc. La implantación de estas tecnologías a otro tipo de industrias se sigue impulsando, gracias a la disponibilidad de abrir implementaciones de la pila de EPC [Flörkemeier et al., 2006].

El Auto-ID center no es el único organismo que propone estándares que pretenden normalizar ciertos sectores englobados en Internet de las cosas, cada vez hay más propuestas por parte de empresas y asociaciones. Entre las especificaciones que más aceptación están obteniendo se encuentran las que provienen de ZigBee Alliance, esta entidad es una asociación de empresas que trabajan conjuntamente para avanzar en materias de interconexión inalámbrica, monitoreo y control de productos de una manera fiable, rentable y de bajo consumo. Para lograr su objetivo se basan entre otras cosas en la creación de estándares globales abiertos. El objetivo de la Alianza ZigBee es ofrecer al consumidor la máxima flexibilidad, movilidad y facilidad de uso mediante la construcción de la inteligencia y las capacidades inalámbricas en dispositivos de uso común. La tecnología ZigBee se está incorporando en una amplia gama de productos y aplicaciones de consumo, comerciales, industriales y gubernamentales.

Las normas que regirán el desarrollo de Internet de las cosas deberían estar diseñadas para soportar una amplia gama de aplicaciones. El número de requisitos es muy grande debido a la diversificación de la industria y de los sectores que aplicarían estas tecnologías, así como a las necesidades del medio ambiente, de la sociedad y de las personas. A través de múltiples procesos y consensos se trata de desarrollar modelos de datos normalizados, interfaces y protocolos comunes, haciendo la definición inicial en un nivel abstracto, y posteriormente realizando especificaciones para distintas plataformas o tecnologías. También el uso de ontologías y la semántica juegan un importante papel dentro de la evolución de Internet de las cosas, ayudando a eliminar las ambigüedades resultantes de errores humanos y las diferencias entre interpretaciones debido a los diferentes lenguajes. Las normas son especialmente necesarias en temas de comunicación e intercambio de información entre las cosas, entornos, y sus homólogos de la nube virtual [Sundmaeker et al., 2010]. Además, el diseño de normas para Internet de las cosas debe tener en cuenta medidas eficientes con respecto al uso de energía y capacidades de las

redes, así como respetar las limitaciones de otros reglamentos existentes como el que restringe el uso de las bandas de frecuencia.

CAPÍTULO 4

EL TELÉFONO INTELIGENTE

4.1 INTRODUCCIÓN

El avance de tecnología dentro del sector de los teléfonos móviles ha hecho que estos dispositivos cada vez posean características más avanzadas. Características que los capacitan para desarrollar un mayor número de tareas: alta capacidad de computación, sensores de luz, sonido y movimiento, receptores GPS, cámaras, lectores RFID, conexión a Internet, Bluetooth, Near Feald Communication, etc.

El teléfono inteligente comúnmente denominado "Smartphone" se ha convertido en un fenómeno emergente tanto para uso personal como empresarial. El ahorro de energía en sus procesadores, los sistemas operativos móviles de nueva generación, y la conexión a Internet de banda ancha han mejorado la productividad e impulsando la popularidad de estos dispositivos. El Smartphone resulta de la convergencia entre los teléfonos móviles y las PDA [Chang et al., 2009]. Hay una serie de características concretas que deben estar asociadas a un teléfono móvil para que sea considerado un Smartphone: alta capacidad de procesamiento, un sistema operativo potente, (los más extendidos: Windows Mobile, Android, iPhone OS, Symbian, BlackBerry OS), gestión de ficheros y ofrecer varios mecanismos de comunicación como Bluetooth, Wi-Fi, etc. Cada generación de nuevos dispositivos añade características adicionales

Tanto por la buena acogida de los Smartphones en la sociedad como por el aumento de sus características técnicas de este tipo de dispositivos, se ha propiciado que cada vez se usen más y para realizar un mayor número de tareas. Muchas de estas nuevas tareas asociadas a los Smartphones están directamente relacionadas con Internet de las cosas.

En este capítulo se realizará una revisión sobre los aspectos clave que potencian el uso del Smartphone, se revisaran los siguientes puntos:

- Servicios. Se trata de servicios electrónicos que residen en ubicaciones específicas y han sido especialmente diseñados para ser consumidos por dispositivos móviles.
- Aplicaciones que se les están asignando actualmente a este tipo de dispositivos y que encajan dentro del marco de Internet de las cosas.
- Comercio móvil.
- Mecanismos de comunicación para la interacción con otros objetos o servicios.

4.2 SERVICIOS

Los dispositivos móviles deben ser considerados como un participante más dentro de la arquitectura orientada a servicios. Los teléfonos inteligentes suelen tomar el papel de consumidores de servicios en la mayoría de modelos de negocio que observamos en la actualidad, ya sea accediendo a servicios específicos vía Bluetooth, o a servicios web convencionales. Que los teléfonos móviles sean consumidores de servicios es lo más común pero no es la única posibilidad [Rohs & Gfeller, 2004]. Gracias a la capacidad de computación avanzada y la disposición de varios protocolos de comunicación (Internet, Wi-Fi, Bluetooth, etc.) estos dispositivos poseen un gran potencial y versatilidad para ser utilizados como proveedores de servicios. Por ejemplo, el teléfono podría actuar como proveedor de servicios para otros dispositivos que se encontrasen cerca de él o variar los servicios que ofrece dependiendo del contexto en el que se encuentre [Espada et al., 2011c].

4.2.1 Acceso a servicios en contextos específicos

Los teléfonos móviles inteligentes tienen un gran potencial para servir como herramienta de acceso a servicios electrónicos o a aplicaciones que residen en una ubicación específica. Los servicios específicos ya existen en numerosos puntos como: quioscos de información electrónica, catálogos de productos, etc. La integración de los teléfonos móviles puede mejorar aun más la interacción con esta clase de servicios. Mediante el uso de información almacenada en los teléfonos, los servicios específicos podrían adaptarse de manera automática a un usuario en particular. Además, los usuarios pueden almacenar la información resultante de la interacción con un servicio en su teléfono inteligente para realizar futuras consultas (por ejemplo, la descarga de un mapa). Se puede aumentar la eficiencia para la realización de ciertas tareas ya que varios usuarios podrían acceder al mismo tiempo a un servicio específico [Toye et al., 2005] por ejemplo: consultar la información sobre un objeto en un museo.

4.3 APLICACIONES

Entre las capacidades más difundidas de los teléfonos móviles dentro del contexto de Internet de las cosas se encuentran el enlace con objetos o servicios y el control o monitorización de objetos.

4.3.1 Enlace con objetos y servicios

Enlace con objetos

Los dispositivos móviles pueden reconocer objetos etiquetados con códigos de barras y etiquetas RFID, el teléfono y la cámara pueden colaborar para identificar elementos sin etiquetar como a las personas, lugares y demás objetos. Normalmente la información del objeto reconocido se coteja con un servidor, de esta forma un teléfono puede dar detalles sobre el producto incluyendo sus atributos, origen, precio, garantía, opiniones, manual del usuario, compra, reciclaje, etc.

Enlace con servicios

En ocasiones se utiliza el código de barras o el reconocimiento de objetos para realizar un acceso rápido a un servicio, por ejemplo: es común que los recursos descargables para teléfonos móviles se identifiquen con un código de barras, escaneando estos códigos con la cámara del teléfono se puede identificar un producto no físico y establecer una conexión con el servidor para iniciar su descarga. Otro tipo de ejemplo de enlace con servicios se puede encontrar en algunos restaurantes japoneses, en los cuales es posible realizar pedidos seleccionando los objetos mostrados en un poster mediante un teléfono móvil.

4.3.2 Reconocimiento de objetos

La extensión de Internet de las cosas promete un mundo más inteligente, donde hay un alto grado de interacción entre personas y objetos. Entre las formas de interacción más básicas se encuentran: la solicitud de información sobre los objetos y la ejecución de acciones asociadas a los mismos. Para realizar estas acciones existen varios métodos, actualmente la gran mayoría de ellos se basan en una especie de código integrado en el objeto. Algunos de estos marcadores se pueden analizar mediante diferentes sistemas de comunicación inalámbrica (por ejemplo las etiquetas RFID o Bluetooth beacons [Fuhrmann & Harbaum, 2003]), otros sistemas se basan en marcadores visuales que suelen ser analizados utilizando cámaras o escáneres, como pueden ser los códigos de barras [Adelmann et al., 2006] o los códigos 2D [Rohs & Gfeller, 2004].

Los teléfonos móviles integran además de cámaras una amplia gama de canales de comunicación adicionales: Bluetooth, WLAN, Wi-Fi. Otros sistemas no requieren marcadores para reconocer e identificar un objeto, sino que lo hacen a través de su apariencia, es decir, utilizando el reconocimiento de objetos visuales de una imagen tomada con la cámara. Con estos sistemas, tomando una foto de un objeto sería suficiente para identificarlo o solicitar su información. Si bien esta visión está lejos de convertirse en realidad, los recientes avances en el campo de la visión por ordenador han dado lugar a

métodos que permiten reconocer ciertos tipos de objetos de manera muy fiable y utilizarlos como "hipervínculo" hacia la información digital.

Cierto tipo de objetos no son apropiados para colocar marcadores, esto incluye lugares de interés turístico y grandes edificios. Usar estos métodos de reconocimiento aporta varias ventajas a la hora de reconocer un objeto, como sería el caso de un monumento que está a cientos de metros de distancia; pero incluso si el objeto está cerca, los marcadores pueden ser más prácticos que un código de barras o una etiqueta RFID fijado sobre la etiqueta de un objeto, en un museo sería difícil tener acceso a un código de barras si la habitación estuviese llena de gente. Sin embargo si se pudiese identificar el objeto mediante una foto del mismo, la identificación podría realizarse desde cualquier punto de la sala. Además el etiquetado habitual de los objetos es a menudo difícil de lograr. Un ejemplo de ello son los carteles de publicidad en exteriores. Si una empresa anunciante diseña un cartel-hipervínculo debería instalar una etiqueta RFID o Bluetooth en cada panel de publicidad o adjuntar un código de barras a cada uno de ellos, lo que requiere un sistema normalizado y posiblemente un aumento en los costos de instalación y mantenimiento. Dicho esto, el reconocimiento de objetos tiene algunas restricciones. En la actualidad resulta imposible distinguir dos objetos muy similares, por ejemplo, dos versiones ligeramente diferentes del mismo producto. Por otra parte, la indexación y búsqueda visual eficiente entre millones de características constituye todavía un problema sin resolver [Quack et al., 2008].

4.3.3 Control y monitorización de objetos

Los Smartphones pueden ser utilizados como mandos a distancia de multitud de “cosas”. En la actualidad muchos de estos dispositivos permiten su utilización como mandos a distancia en equipos audiovisuales o computadores. Con la evolución de Internet de las cosas el número de objetos con dispositivos electrónicos embebidos y conectados a la red aumentará enormemente; Por lo que los teléfonos podrían convertirse en un medio para controlar la mayoría de los objetos cercanos o distantes que se encuentren conectados a la red: puertas, cerraduras, sistemas de seguridad, luces, aparatos y equipos de oficina.

Cada vez se están elaborando más propuestas de sistemas que permiten controlar objetos remotamente desde dispositivos móviles; una de las ventajas de este control remoto es que puede realizarse tanto de manera “cercana” como a través de Internet. Gran parte de estas propuestas se centran en entornos residenciales, permitiendo establecer conexión con la red domestica y controlar los dispositivos que la integran desde cualquier lugar [Fasbender et al., 2008].

4.3.3.1 Control y monitorización de objetos

Con el creciente números de objetos conectados aumentan también las posibilidades de monitorización de los mismos. En la actualidad es común que se realicen labores de monitorización desde páginas web o aplicaciones, pero con la introducción de los dispositivos móviles en estas actividades la monitorización puede realizarse desde cualquier lugar, el usuario puede recibir las notificaciones rápidamente mediante una aplicación específica o por SMS.

Tanto desde el ámbito empresarial como desde la investigación se están utilizando cada vez más los teléfonos móviles para tareas de monitorización de objetos y redes. Por ejemplo dentro del ámbito de la salud MobiCare es una arquitectura que da soporte eficientemente a un gran rango de servicios médicos [Chakravorty , 2006]. Los sensores toman datos de multitud de parámetros médicos, los cuales se conectan a través de una interfaz Wireless para comunicar sus mediciones al teléfono móvil del paciente y de este se envía al servidor.

La incorporación de los teléfonos móviles para labores de monitorización en ambientes residenciales resulta muy atractiva y practica para los usuarios, existen multitud de sistemas domésticos de monitorización y control remoto en temas de seguridad y domótica (monitorización, cámaras, cerraduras, etc.). En la actualidad, gran parte de los objetos se controlan directamente a través de infrarrojos, Bluetooth u otras tecnologías basadas en ondas de radio; pero están surgiendo varias propuestas que se apoyan en la publicación en servidores de los controles pertenecientes a los dispositivos, es decir conectarlos a Internet de las cosas. Lo que permitiría el descubrimiento y la monitorización de un número ilimitado de dispositivos a través de un único dispositivo móvil, de manera sencilla y desde cualquier lugar [Atukorala et al., 2009].

4.4 COMERCIO MÓVIL

El comercio móvil incluye una gran variedad de actividades relacionadas con transacciones monetarias que se inician o llevan a cabo a través de un teléfono móvil. Estas transacciones incluyen bienes intangibles y tangibles. Ejemplos de bienes intangibles son el software o la información entregada al propio dispositivo móvil en formato digital. Aunque aún no está totalmente extendido hay sistemas para que cierta clase de bienes tangibles puedan ser adquiridos también pagando a través del teléfono móvil como es el caso de bebidas, tickets de metro, etc. a través de determinadas máquinas expendedoras. Los teléfonos móviles se pueden utilizar para una amplia gama de transacciones, ya sea mediante la compra a distancia a través de la red o de manera local en un punto de venta específico.

Los mecanismos de pago utilizados por el comercio móvil son prácticamente iguales a los utilizados en un pago electrónico tradicional [Chang et al., 2009].

1. Dinero electrónico: Digicash, Cybercoin y Netcach [Karnouskos, 2007].

2. Tarjeta de crédito / débito – El pago con tarjeta de crédito es la forma más popular de pago electrónico.

3. Pequeños pagos - el precio de la carga es directamente facturado al número de teléfono, normalmente estos pagos suelen hacerse para artículos de pequeño coste.

4. E-Check a través de Automated Clearing House (ACH) [Automated Clearing House] – este mecanismo realiza una verificación similar a un cheque de papel ordinario, la verificación activará una transferencia electrónica de fondos. Los E-Check pueden tramitarse más rápido que el papel cheque; por lo general implican el uso de ACH.

Los factores de motivación para el uso de Smartphones en el comercio móvil son [Chang et al., 2009]:

1. Conveniencia, los consumidores pueden iniciar la operación en cualquier momento y lugar.

2. Notificaciones vía SMS, EMS.

3. Facilidad de uso, el porcentaje de gente que sabe manejar un ordenador es más reducido que el que sabe manejar un teléfono móvil.

4. Ahorro de costes y acceso a mayor número de clientes potenciales. Hoy en día la mayoría de la gente dispone de un teléfono móvil que llevan siempre consigo.

4.5 COMUNICACIONES

Los mecanismos de comunicación inalámbrica más extendidos con los que cuentan los Smartphones son Wi-Fi [Wi-Fi, 2010] y Bluetooth [Bluetooth, 2009], aunque no son los únicos. Los dos son estándares de redes inalámbricas y ofrecen conectividad a través de ondas de radio. La principal diferencia entre ellos es el uso "ideal"; Bluetooth surge para sustituir los cables, mientras que la utilización más demandada del Wi-Fi, es la de proporcionar acceso inalámbrico a Internet o redes de área local. Las características técnicas de ambos estándares los capacitan para realizar descubrimientos e interacción con objetos y servicios [Fajardo et al., 2008] pero dependiendo de los objetos o servicios a los que se quiera acceder y del contexto, una tecnología se podría perfilar más eficiente y adecuada que la otra.

4.5.1 Bluetooth

Esta tecnología posee un bajo consumo, un alcance corto y velocidades de transmisión moderadamente rápidas. Bluetooth suele utilizarse para proporcionar una conexión inalámbrica punto a punto. Esta tecnología inalámbrica puede utilizarse en cualquier lugar que tenga dos o más dispositivos compatibles.

Bluetooth resulta muy útil cuando se pretende realizar una interconexión entre dos o más objetos que están cercanos y no hay necesidad de transferir datos muy pesados, dentro del mundo de Internet de las cosas hay muchos objetos que incorporan la tecnología Bluetooth. Esta tecnología resulta adecuada para varios tipos de comunicaciones:

- Comunicarse con objetos cercanos que no son de nuestra propiedad, comunicaciones en las que se intercambie información poco pesada, y sea lógico estar cerca del punto de suministro para consumir sus servicios, estas comunicaciones suelen tener un carácter esporádico. Por ejemplo un servidor de información de productos en un centro comercial.

- Conexiones ininterrumpidas con objetos cercanos que sean de nuestra propiedad: un brazalete que mide las pulsaciones podría estar sincronizado vía Bluetooth con el teléfono móvil.

4.5.2 Wi-Fi

Dependiendo del caso las conexiones Wi-Fi pueden establecerse a más de 100 metros de distancia de un nodo. Actualmente Wi-Fi es también la comunicación base de muchos productos electrónicos y electrodomésticos, lo que permite que se incorporen a la red domestica o proporcionarles acceso a Internet. Dentro del mundo de Internet de las cosas hay muchos objetos que incorporan esta tecnología y resulta adecuada tanto para la interacción puntual con un objeto, como para la creación de redes en las que varios objetos necesitan comunicarse entre sí de manera continuada. Prácticamente todos los objetos que se conectan a Internet cuentan con esta tecnología, la conexión a la red de estos objetos aporta la ventaja de que puedan ser controlados o monitorizados desde fuera del hogar a través de servidores remotos.

4.5.3 NFC: Near Field Communication y RFID

Las tecnologías de identificación por radiofrecuencia disfrutan de una enorme aceptación, no sólo desde el punto de vista de la investigación, sino también de la práctica empresarial. Las empresas de diversos sectores están generando soluciones para una amplia gama de problemas a través de RFID y NFC. Los objetivos van desde el mero incremento de la eficiencia de procesamiento para la recepción y envío de mercancías, hasta la lucha contra la falsificación de productos. Tal vez la iniciativa de normalización más influyente hasta la fecha haya sido la del Auto-ID Center, que se transformó en la organización sin ánimo de lucro EPC global en el 2003. Los resultados de las actividades de investigación del centro Auto-ID se conocen como la red "EPC", una colección de hardware y software estándares para el uso de RFID en varias industrias.

A medida que el RFID ganó popularidad en la industria, surgió un segundo estándar que está estrechamente relacionado con él. Near Field Communication (NFC) [Ecma, 2005] denota una tecnología que permite la integración de la funcionalidad RFID en los dispositivos personales, tales como teléfonos móviles. Por el momento el número de teléfonos con tecnología NFC disponibles en el mercado es todavía escaso, algo que puede cambiar en un futuro muy cercano. Por otro lado, la presencia de etiquetas EPC está aumentado rápidamente. Sin embargo, las dos normas no han sido desarrolladas para ser compatibles entre sí [Wiechert et al., 2009].

4.5.3.1 Aportaciones de NFC

Near Field Communication fue aprobado como un estándar ISO / IEC el 8 de diciembre de 2003 y más tarde como un estándar ECMA. Es un protocolo basado en una interfaz inalámbrica. La comunicación se realiza entre dos entidades. El protocolo establece conexión wireless entre las aplicaciones de la red y los dispositivos electrónicos a distancias cortas. Combina la funcionalidad de un dispositivo lector RFID y etiqueta RFID en un solo circuito integrado.

La tecnología NFC no está pensada para ser utilizada en una transmisión masiva de datos, como puede darse con las tecnologías Wi-Fi o Bluetooth, sino para el intercambio rápido de unos pocos bits de información, lo estrictamente necesario para que identifique y valide al usuario. La comunicación entre dispositivos NFC se realiza a través de un diálogo: a una petición del dispositivo iniciador responde él dispositivo destino, debiendo responder antes de recibir otra petición.

Los dispositivos NFC pueden realizar:

1. Smart Card emulación. Este modo permite el uso del dispositivo NFC como tarjeta de crédito, de contacto o de billete electrónico entre otros.

2. Peer-to-Peer. Trasferencia de datos entre dos dispositivos.

3. Escritura / Lectura de tags RFID.

Los beneficios de la NFC más mencionados en las publicaciones e industria son: pago seguro, compra de entradas, rápida transferencia de datos entre dispositivos, vinculación

entre dispositivos tales como otros teléfonos o accesorios y la descarga de información por ejemplo mediante carteles inteligentes [NFC, 2012 ; GSM, 2007]. Otro punto importante a señalar es que existe la implementación de servicios de pago sin contacto por medio de: Visa VisaWave, MasterCard PayPass, y American Express express pay [Wiechert et al., 2009]. Por lo que se está impulsando el uso comercial de esta tecnología.

Varios sistemas basados en NFC ya se han desplegado en el marco de proyectos piloto, como algunos sistemas de venta de entradas en China, Japón y Alemania. También hay dos proyectos previstos para la adquisición de tickets de metro en Nueva York y Londres [Hardgrave et al., 2005]. Sin embargo, la barrera para la adopción generalizada sigue siendo la falta de dispositivos NFC en los usuarios consumidores.

CAPÍTULO 5

INTERACCIÓN REMOTA CON ELEMENTOS FÍSICOS

5.1 INTRODUCCIÓN

La forma más extendida de comunicación entre personas y dispositivos electrónicos se basa en la utilización de las interfaces de usuario contenidas en los propios dispositivos. Interfaces tales como: botones, pantallas táctiles, etc. Sin embargo existen otras formas de interacción que se han vuelto muy populares en los últimos tiempos, como por ejemplo las que se basan en la utilización de Smartphones. La aplicación de los dispositivos móviles en las tareas de control remoto no es algo novedoso, en 1997 el Huma-Computer Interaction Institute (HCII) de la universidad Carnegie Mellow presento el proyecto Pebbles. Este proyecto es uno de los pioneros en presentar la utilización de dispositivos móviles (PDAs) como herramientas de control remoto efectivas [Myers et al., 1998].

En la actualidad los Smartphones pueden aportar muchos beneficios en los procesos de comunicación que involucran a dispositivos electrónicos y usuarios [Nichols & Myers, 2006]. Los Smartphones permiten intercambiar información con otros dispositivos electrónicos, ya que están equipados con mecanismos de comunicación tales como Bluetooth y Wi-Fi. De esta manera una persona puede ser capaz de interactuar con diversos dispositivos electrónicos utilizando su Smartphone. La utilización de los teléfonos móviles con el fin de interactuar con otros dispositivos electrónicos es una práctica cada vez más extendida. Esta tendencia se debe a varios factores: la mejora sustancial de las características técnicas de los Smartphones, el aumento de su popularidad entre los usuarios y también a los beneficios derivados de su uso para facilitar la interacción entre usuarios y otros dispositivos electrónicos. La utilización de los Smartphones como vínculo de comunicación entre usuarios y dispositivos electrónicos puede ofrecer numerosas ventajas, algunas de las más destacadas son:

1. Comunicación múltiple, permite a varios usuarios comunicarse de manera simultánea con un mismo dispositivo electrónico.

2. Permiten adaptar y mejorar el diseño de las interfaces de usuario. En muchos escenarios los teléfonos móviles permiten mejorar la experiencia de usuario a través de interfaces de usuario más completas y personalizables. Las aplicaciones móviles cuentan con numerosos elementos útiles para esta labor: opciones de configuración, preferencias de usuario, patrones de uso, etc.

3. Separación de aspectos. La separación entre el propio dispositivo electrónico y las interfaces de usuario, permite reducir costes de fabricación al suprimir la necesidad de incluir interfaces tradicionales como pantallas y botones. Por otro lado esta separación también favorece la posibilidad de realizar actualizaciones o mejoras en las interfaces sin modificar el dispositivo electrónico. Si los dispositivos electrónicos son más simples al implicar el uso de menos componentes, se pueden reducir los costes de mantenimiento, sobre todo en caso de objetos electrónicos de uso público.

En los últimos años el número de dispositivos electrónicos candidatos a ser remotamente controlados desde teléfonos móviles ha aumentado considerablemente, sobre todo en el entorno domestico, la oficina y la industria [Myers et al., 2004]. La evolución tecnológica de los dispositivos implicados en los sistemas de control remoto ha derivado en diferentes arquitecturas. Hay varios factores que pueden resultar determinantes a la hora de definir la arquitectura de un sistema de control remoto, los componentes hardware y software implicados, los objetivos del sistema, las especificaciones de los dispositivos, los tipos de comunicación, factores ambientales, etc.

A continuación se presentan varias de las alternativas más relevantes empleadas en la construcción de sistemas de control remoto que implican el uso de dispositivos móviles.

5.2 APLICACIONES NATIVAS

La forma más habitual de habilitar el uso de los Smartphones como controles remotos de otros dispositivos electrónicos es mediante la utilización de aplicaciones nativas, las cuales han sido especialmente diseñadas para ese propósito. Estas aplicaciones se ejecutan en los Smartphones con el fin de habilitar la comunicación entre usuarios y otros dispositivos electrónicos. Las tiendas de aplicaciones móviles más populares contienen muchas aplicaciones de este tipo, como por ejemplo: LG TV Remote, Samsung Remote, DSLR Remote Controller, AV Controller Yamaha, Logitech Squeezebox Controller, entre otras. La aceptación de este tipo de aplicaciones se refleja en el número de usuarios y en la cantidad de valoraciones positivas recibidas. La utilización de la aplicación Samsung Remote para la plataforma Android se incremento en más de 750.000 usuarios en los últimos seis meses del año 2011 [Samsung, 2011].

Las aplicaciones móviles nativas están diseñadas para un dispositivo específico o un grupo de dispositivos de la misma plataforma. En la actualidad hay una gran variedad de modelos de Smartphones, con diferentes especificaciones y características: tamaño de pantalla, memoria, capacidad de procesamiento y sistema operativo (iOS, Android, WebOs, Windows Phone, Symbian, etc). Esta heterogeneidad entre dispositivos móviles hace que en muchos casos los fabricantes se vean obligados a desarrollar una versión de la misma aplicación para cada plataforma. Por otro lado, la primera vez que un usuario utiliza una aplicación nativa para interactuar con otro dispositivo electrónico, emplea gran parte del tiempo total en la descarga e instalación de la aplicación. La utilización de las aplicaciones nativas puede implicar una gran pérdida de tiempo en escenarios donde los usuarios interactúan de manera ocasional con otros dispositivos, como por ejemplo: dispositivos proveedores de servicios en las ciudades o establecimientos, los electrodomésticos de una habitación de hotel, etc.

5.3 APLICACIONES WEB

Para reducir el aumento de costes y esfuerzo derivados de la necesidad de desarrollar y mantener distintas versiones de la misma aplicación en diferentes plataformas móviles, muchos desarrolladores optan por desarrollar sus aplicaciones móviles utilizando tecnologías web. La utilización de las tecnologías web permite desarrollar aplicaciones que pueden ser utilizadas en múltiples plataformas, al contrario de lo que ocurre con las aplicaciones nativas. Estas medidas no solo permiten ahorrar costes de desarrollo, sino que también aseguran que las aplicaciones desarrolladas pueden ser utilizadas por una gran cantidad de usuarios, ya que la mayoría de los Smartphones de hoy en día incluyen algún tipo de navegador web [Hernandez, 2009].

Las aplicaciones web y las aplicaciones nativas son muy diferentes técnicamente. Las aplicaciones nativas corren sobre el propio sistema operativo del dispositivo, en cambio las aplicaciones web se ejecutan en un servidor externo que envía varios tipos de archivos (HTML, CSS, Scripts) a un navegador web móvil. Dependiendo de los objetos y de la funcionalidad de la aplicación, esta puede ser desarrollada como una aplicación web o no. Las aplicaciones nativas tienen varias características que resultan difíciles de imitar por parte de las aplicaciones web [Espada et al., 2012]. Una de estas características es la gestión de los componentes hardware del dispositivo cliente, como por ejemplo los sensores, GPS y los elementos de comunicación Wi-Fi y Bluetooth [Gossweiler et al., 2010].

Precisamente los elementos de comunicación Wi-Fi y Bluetooth son piezas clave en la mayoría de aplicaciones control remoto, ya que las aplicaciones requieren la gestión de estos elementos para poder comunicarse de manera directa con otros dispositivos electrónicos. Algunos componentes ActiveX [ActiveX, 2011] como la Wireless Communication Library [WCL, 2012] habilitan la gestión de elementos de comunicación a partir de aplicaciones que se ejecutan dentro del navegador. Las posibilidades de crear aplicaciones web multiplataforma utilizando esta alternativa son limitadas, ya que normalmente los componentes ActiveX solamente son compatibles 100% con entornos Windows y el navegador Internet Explorer. Los Java Applets [Applets, 2011] y las aplicaciones Java Web Sart [Java, 2011] pueden ser ejecutados dentro de un navegador web consiguiendo una funcionalidad similar a los ActiveX. Estas tecnologías Java también son técnicamente capaces de gestionar los elementos de comunicación del dispositivo cliente. Aunque estas tecnologías Java se utilizan de manera frecuente en las aplicaciones web, no están soportadas en la mayoría de los móviles con sistema operativo Android, BlackBerry OS, iOS y Symbian. Para poder utilizar estas tecnologías Java se necesita cumplir una serie de requerimientos que no están disponibles para la gran mayoría de las plataformas móviles, como los plugins específicos del navegador y la maquina virtual de Java.

5.4 AMBIENTES CENTRALIZADOS

Las redes domésticas son el objetivo de muchos proyectos comerciales y de investigación que utilizan los dispositivos móviles como ejes centrales, por lo general estos sistemas presentan limitaciones, como la heterogeneidad de los dispositivos que son capaces de controlar ya que existe un gran número de protocolos diferentes: X-10, Bluetooth, Universal Plug and Play (UPnP) [Rose, 2001].

Entre los diferentes tipos de sistemas de control remoto, podemos identificar aquellas propuestas que se basan en la utilización de servidores y servicios para construir un ambiente centralizado capaz de gestionar dispositivos heterogéneos. En la mayoría de los casos los dispositivos pertenecientes al entorno centralizado están conectados a algún componente o servidor que actúa como controlador [Xu et al., 2009]. En estos sistemas el elemento controlador que gestiona el entorno centralizado suele proporcionar interfaces de usuario basadas en tecnologías web. Los interfaces web facilitan que cualquier dispositivo con un navegador pueda controlar remotamente cualquiera de los dispositivos que forman el entorno centralizado [Filibeli et al., 2007]. Esta clase de sistemas requieren de un entorno centralizado en el que los dispositivos hayan sido asociados a un elemento hardware específico como por ejemplo un servidor, en el caso de Ethernut [Fibelli et al., 2007] este componente se denomina "Ethernut Home Server" (EHS). El servidor es el responsable de coordinar y controlar los dispositivos electrónicos que forman parte del ambiente centralizado. Existen varias alternativas comerciales basadas en arquitecturas de carácter similar. Los fabricantes RHUB [RHUB, 2012], Z-Wave [Z-Wave, 2012] y la ZigBee Alliance [ZigBee Alliance, 2012] entre otros, comercializan diferentes tipos de componentes electrónicos que pueden ser utilizados en la creación de entornos centralizados. Varios de estos componentes ofrecen la posibilidad de gestionar el entorno a través de interfaces web.

La utilización de componentes hardware adicionales como el EHS permite que el sistema realice labores de coordinación compleja de dispositivos, pero en muchos escenarios el uso de estos componentes hardware adicionales no reporta ningún beneficio. Los componentes adicionales aumentan la complejidad de los sistemas, la posibilidads de que se produzcan errores, los costes y los tiempos de comunicación. Los Smartphones actuales son técnicamente capaces de comunicarse directamente con otros dispositivos sin necesidad de hacer uso de componentes intermedios.

5.5 ADAPTADORES

Varias propuestas introducen el uso de componentes software multiplataforma los cuales pueden ser utilizados para construir sistemas de control remoto [Pascual et al., 2011a]. En este tipo de sistemas de control remoto el componente software funciona como un adaptador, que puede ser transferido y ejecutado en dispositivos móviles pertenecientes a diferentes plataformas. De esta forma un dispositivo móvil puede ejecutar múltiples adaptadores que le permitan controlar de forma remota multitud de dispositivos electrónicos, incluso incluyendo dispositivos electrónicos con diferentes protocolos de comunicación y especificaciones [Kanma et al., 2003].

Este enfoque resulta adecuado para varios escenarios. El adaptador es un componente software que se transfiere y ejecuta entre dispositivos, estos adaptadores contienen código que se ejecuta en el dispositivo, lo que resulta potencialmente peligroso. El proceso de descarga y configuración de los componentes adaptadores puede llegar a consumir un tiempo significativamente alto. La primera vez que el usuario utilice el adaptador, la mayor parte del tiempo consumido en el proceso se habrá empleado en la descarga y configuración del componente. Este enfoque no resulta adecuado en escenarios donde los usuarios interactúan de manera ocasional con otros dispositivos, como por ejemplo: dispositivos proveedores de servicios en las ciudades o establecimientos, en una habitación de hotel, etc.

CAPÍTULO 6

INFORMACIÓN DE CONTEXTO

6.1 INTRODUCCIÓN

La primera definición de aplicaciones sensibles al contexto fue presentada por Schilit y Theimer [Schilit & Theimer, 1994], en ella se limitaban las aplicaciones sensibles al contexto a aquellas aplicaciones que eran informadas del contexto para que se adaptaran en consecuencia. Pronto el término "sensible al contexto" (context-aware) comenzó a extenderse para designar a aquellas aplicaciones que hacían uso de cualquier tipo de información de contexto.

Hay muchas definiciones distintas de la "información de contexto" [Schimidt, 1999] ya que no parece haberse acordado una definición totalmente satisfactoria. Algunos investigadores han tratado de definir este concepto por medio de ejemplos [Chen & Kotz, 2000 ; Schilit et al., 1994] otros autores han buscado definiciones más formales [Schmidt et al., 1999 ; Abowd et al., 1999].

Aunque no se haya podido llegar a una definición totalmente aceptada, la mayoría de los autores tiene conceptos similares sobre a lo que nos estamos refiriendo al citar la información de contexto. Normalmente en el ámbito de los sistemas informáticos la información de contexto hace referencia a toda aquella información que proviene del mundo real. Los dispositivos electrónicos actuales permiten capturar varios tipos de información de contexto, como la localización, el nivel de luz, la orientación del dispositivo, la temperatura ambiente, etc. En la mayoría de los casos ésta se obtiene utilizando unos sensores específicos y otros elementos hardware.

Los Smartphones resultan muy apropiados para la captura y empleo de información de contexto. Estos dispositivos suelen incluir muchos mecanismos como sensores, micrófonos, cámaras, etc. El número de aplicaciones móviles que utilizan la información de contexto ha crecido considerablemente en los últimos años. Muchas aplicaciones, tanto comerciales como pertenecientes al ámbito de la investigación, utilizan información directamente capturada del mundo real. Esta información suele utilizarse como entrada para determinadas tareas [Kim et al., 2009 ; Lahti et al., 2006] como utilizar el GPS para obtener de manera automática la posición, la cámara de fotos para decodificar un código de barras, los altavoces para capturar comandos de voz, los sensores de aceleración para interactuar con los interfaces, entre otras. Otro uso común de la información de contexto es el de configurar las tareas de la operación en función de los diferentes contextos, como por ejemplo búsquedas sensibles a la localización [Bhatti et al., 2004 ; Sheng et al., 2009 ; Ahn & Nah, 2010]. La información de contexto es un aspecto clave en el proceso de interacción entre personas y dispositivos electrónicos, la

integración del mundo físico y el mundo digital permite desarrollar aplicaciones novedosas e innovadoras [Chua et al., 2011].

6.2 FRAMEWORKS PARA EL DESARROLLO DE APLICACIONES

Existen varios frameworks que provén soporte para simplificar tanto la implementación como otros aspectos relacionados con la integración de la información de contexto en las aplicaciones software.

Los sistemas sensibles al contexto han sido objeto de muchas investigaciones desde hace más de una década. Muchos de estos enfoques se han centrado en aplicaciones específicas sensibles al contexto, tales como guías de turismo [Cheverst et al., 1999] o asistentes de recomendación personal [Riva & Toivonen, 2006]. Como contraste a estos sistemas fuertemente acoplados se han derivado otra clase de propuestas que abordan la gestión de información de contexto de una forma más genérica e independiente de la lógica de negocio de los sistemas. La mayoría de estas propuestas podrían catalogarse como frameworks que proporcionan diferentes tipos de apoyos al desarrollo de sistemas que emplean información de contexto. Estos entornos de trabajo acostumbran a ofrecer soporte a la captura de información de contexto y en muchas ocasiones también a un conjunto de características adicionales como la interpretación o la difusión de la misma.

A continuación se realizará una revisión de varios de los frameworks más relevantes que ofrecen soporte al desarrollo de aplicaciones sensibles al contexto. Los frameworks citados presentan varias características específicas que resultan de especial interés para el desarrollo de este trabajo de investigación. Aunque ninguna de las propuestas analizadas comparte la totalidad de sus objetivos con esta tesis, si que permiten obtener una visión general sobre algunas características comunes como pueden ser distintos mecanismos empleados en la gestión de información de contexto y los componentes que ofrecen soporte a otras tareas o funcionalidades complementarias.

6.2.1 Context Toolkit

Context Toolkit [Salber et al., 1999 ; Dey, 2000] Es uno de los frameworks pioneros en ofrecer soporte al desarrollo de aplicaciones sensibles al contexto. Esta propuesta introduce el concepto de widgets en los sistemas sensibles al contexto. Un widget o un interactor es una abstracción que facilita la separación de la semántica de la aplicación de bajo nivel de las entradas que manejan los detalles de la aplicación. El método de interacción por widgets se basa en la realización de una consulta que obtiene una respuesta, de esta forma las aplicaciones obtienen la información sobre los cambios producidos. Los widgets tienen un interfaz externo común que permite a las aplicaciones utilizarlos de manera similar. Estos interfaces permiten a las aplicaciones subscribirse a determinados cambios o eventos de un sensor, así como recuperar datos capturados anteriormente.

El Context Toolkit ofrece distintos componentes que permiten la agregación, la mediación y la interpretación de los resultados provenientes de los widgets [Dey et al., 2001]**.** La utilización de los widgets facilita el desarrollo rápido de aplicaciones y prototipos, pero el entorno de trabajo requerido para la utilización de esta tecnología requiere una configuración bastante compleja además de unos requisitos concretos que incluyen componentes hardware específicos [De & Moesser, 2009].

6.2.2 Gaia

Gaia [Román et al., 2002] ofrece un enfoque experimental que desplaza la mayor parte de la computación sensible al contexto a redes accesibles desde infraestructuras middleware. Gaia se basa en un middleware distribuido que soporta el desarrollo y la ejecución de aplicaciones portables sensibles al contexto. El componente middleware es capaz de gestionar varios dispositivos electrónicos que forman parte de una misma red y están concentrados en un espacio físico. Los dispositivos que forman parte de la red podrán ser proveedores o consumidores de información de contexto. El componente middleware provee puntos de entrada uniformes y abstractos a los servicios básicos, estos puntos de entrada serán utilizados por los dispositivos consumidores.

La propuesta proporciona un framework de desarrollo centrado en el usuario, que permite construir aplicaciones móviles que consuman la información de contexto. Basándose en este framework, las aplicaciones pueden recibir eventos procedentes de distintos sensores de la red. Este enfoque simplifica parte del desarrollo de aplicaciones portables, ya que las aplicaciones generadas utilizando Gaia podrían ser dinámicamente particionadas y mapeadas en varios tipos de dispositivos.

En su primera versión la Gaia limita su campo de actuación a dos entornos concretos: la enseñanza y la oficina. Las aplicaciones generadas con este framework sufren de falta de flexibilidad y generan incertidumbre en su manejo. Además este modelo está enfocado a la creación de espacios inteligentes en oposición a los que se centran en el dispositivo móvil [De & Moesser, 2009].

6.2.3 CoBrA

Context Broker Architecture (CoBrA) [Chen at al., 2003 ; Chen et al., 2004]. Es una arquitectura basada en agentes. En este enfoque los mensajes se envían a un tablero de mensajes compartido "Backboard". Los procesos pueden suscribirse a este tablero para recibir todos los mensajes que han sido escritos o únicamente aquellos que coinciden con un patrón específico. Los patrones de coincidencias pueden variar en los diferentes sistemas.

Existen varios frameworks capaces de gestionar información de contexto que implementan diferentes versiones del enfoque "Backboard". Estos se caracterizan normalmente por un gestor de información de contexto que hace el papel de tablero y otros componentes de apoyo, donde se incluyen las aplicaciones que realizan el papel de clientes y gestores del tablero. El gestor de información de contexto contiene las funcionalidades de procesamiento y razonamiento y proporciona interfaces de consulta a las aplicaciones clientes.

En el caso de Context Broker Architecture (CoBrA) la propuesta se basa en una arquitectura que soporta la sensibilidad al contexto en espacios inteligentes. Esta arquitectura asume que cada espacio inteligente debe tener un gestor de contexto central para controlar el funcionamiento y la estructura de la información.

6.2.4 Sentient

Sentient [Biegel & Cahill, 2004] es un framework que ha sido diseñado principalmente para facilitar el desarrollo de aplicaciones móviles sensibles al contexto. Este framework permite gestionar datos procedentes de uno o varios sensores de manera eficiente y sencilla. Las aplicaciones basadas en Sentient liberan al usuario de escribir el código complejo que suele acompañar a la gestión de los sensores y demás elementos hardware. Una de las características más resaltables de esta propuesta es que provee mecanismos para fusionar la información procedente de varios sensores, permitiendo crear contextos de más alto nivel.

Sentient utiliza un modelo de comunicación basado en eventos que está específicamente diseñado para soportar las comunicaciones ad-hoc inalámbricas. Este framework permite desarrollar aplicaciones móviles capaces de integrarse de manera eficiente y simple con sensores externos. La simplificación de los desarrollos se logra mediante una abstracción de muy alto nivel de sensores y actuadores, de esta forma eliminar la interacción a bajo nivel con los elementos hardware.

La propuesta no contempla la integración de las aplicaciones con los sensores y demás mecanismos hardware del propio teléfono. El desarrollo de las aplicaciones se basa en la utilización de herramientas y entornos propios, por lo que las aplicaciones generadas son poco flexibles a la vez que se dificulta que puedan ser utilizadas en múltiples plataformas.

6.2.5 COMODE

COMODE [Vale & Hammoudi, 2009] es un framework que se basa en la utilización de MDE (Model Driven Engineering) con el objetivo de simplificar el desarrollo de aplicaciones sensibles al contexto. Muchas de las propuestas relacionadas con la captura y adaptación de la información de contexto no se basan en un modelo bien definido, el cual separe claramente las funcionalidades del sistema en elementos reutilizables. COMODE realiza una separación clara de funcionalidades en modelos individuales, a los que aplica diversas técnicas de transformación que permiten generar modelos adaptados de forma independiente a la lógica de negocio y los detalles específicos de la plataforma.

Esta propuesta posee varias características destacables, entre las que se encuentran la orientación a la reutilización de los componentes más básicos y la simplificación del proceso de desarrollo de aplicaciones sensibles al contexto. La generación de aplicaciones mediante la utilización de la propuesta requiere manejar un entorno de desarrollo propio que requiere un proceso de aprendizaje. Actualmente, la especificación no contempla un conjunto significativo de tipos de información de contexto y las aplicaciones generadas están destinadas a una plataforma móvil específica.

6.2.6 Conclusiones

Los objetivos con los que los frameworks analizados fueron desarrollados no coinciden por completo con los de esta tesis aunque si que comparten el eje central sobre el que se orientan el resto de características y funcionalidades: la gestión de la información de contexto.

Del análisis de los frameworks pueden extraerse conclusiones muy valiosas.

Aspectos positivos:

- Abstraen la utilización de los elementos que capturan la información de contexto, haciéndolos más simples y fáciles de utilizar, evitando las implementaciones complejas.

- Varios frameworks aplican una separación de elementos que favorece la reutilización de componentes, separando los métodos de captura de información de contexto de otras tareas como los procesos de razonamiento.

- Algunos de los frameworks abstraen la gestión de los elementos hardware para que estos puedan ser utilizados en distintas plataformas.

- Aunque la mayoría de los frameworks no lo contemplan, algunos de ellos están técnicamente preparados para manejar nuevos tipos de información de contexto. Ya que incluyen diferentes mecanismos para diseñar componentes adicionales que permitan la gestión de nuevos tipos de información de contexto.

Como contrapunto se ponen de manifiesto otros aspectos potencialmente negativos presentes en las propuestas analizadas. Estos aspectos pueden no tratarse de deficiencias en sí, dado que los objetivos de los frameworks analizados difieren de los de esta tesis. Los aspectos negativos señalados corresponden con los problemas que se encontrarían al tratar de adaptar parte del comportamiento de estos frameworks a la consecución de los objetivos de esta tesis.

Aspectos negativos:

- Muchos de los frameworks incluyen unos requerimientos hardware determinados, no aptos para sistemas ligeros. Entre estos requerimientos podemos encontrar: la necesidad de incluir redes de comunicación complejas, entornos específicos para el desarrollo y los componentes software propios sobre los que se sustentan las aplicaciones generadas.

- Complejos, poco eficientes y en ocasiones poco prácticos para ser ampliamente utilizados por los desarrolladores. Esto se debe principalmente a las tareas de configuración y al uso de demasiados componentes hardware y software propios.

- En general este tipo de frameworks no manejan una amplia variedad de tipos de información de contexto, sino que muchos de ellos se ciñen a un pequeño subconjunto o incluso a un solo tipo de información de contexto, como por ejemplo la ubicación que resulta ser el caso más recurrente.

- Los frameworks suelen hacer énfasis y consumir recursos en la consecución de objetivos que no resultan relevantes para el desarrollo de esta tesis, como por ejemplo los sistemas de razonamiento para filtrar la información de contexto capturada.

- La mayoría de estos frameworks se centran en una única plataforma destino. Esto provoca que el conjunto del sistema o las aplicaciones móviles clientes no puedan ser utilizadas en distintas plataformas móviles.

- Varios de los frameworks no soportan la gestión de la información de contexto de una forma genérica, sino que orientan sus procesos hacia la generación de aplicaciones sensibles al contexto enmarcadas dentro de aéreas específicas, como puede ser el turismo o el entorno domestico.

- En cuanto a la recolección de información de contexto, varios frameworks no contemplan ningún mecanismo de soporte a los elementos hardware de los propios dispositivos móviles clientes. Centran el soporte ofrecido en la gestión de elementos hardware externos, como redes de sensores wireless, etc.

6.3 LA INFORMACIÓN DE CONTEXTO EN LA WEB

Varios autores han confeccionado propuestas que trataban de incluir la información de contexto en diferentes tipos de sistemas o aplicaciones basados en la web. Los enfoques para conseguir este objetivo son muy diversos, algunas de estas propuestas modifican los navegadores web tradicionales para incluir el manejo de información de contexto [Challiol et al., 2007]. En cambio otras propuestas se basan en el uso de algún tipo de componente adicional (plugins, extensiones, software de apoyo) que complementa el funcionamiento del navegador web, encapsulando la funcionalidad relacionada con la gestión de la información de contexto [Noltes, 2008 ; Shirazi et al., 2010]. Las diferencias entre las propuestas no solo se sitúan en el enfoque adoptado para realizar la gestión de información de contexto, sino que éstas se ven influidas por los posibles objetivos y usos de la información de contexto capturada. Algunos sistemas han sido concebidos para que la información de contexto pueda ser utilizada de forma genérica, para cualquier objetivo. En cambio otras propuestas se centran en objetivos concretos, como la utilización de la información de contexto para adaptar el aspecto gráfico de la aplicación [Gasimov et al., 2010] o la mejora en los procesos de búsquedas [Hattori et al., 2007 ; Coppola et al., 2010]. La posibilidad de manejar información de contexto resulta muy útil no sólo en las aplicaciones web, sino también para otros tipos de tecnologías web, como los servicios web [Ennai & Bose, 2008]. En algunas propuestas los servicios web sensibles al contexto se utilizan como base para desarrollar aplicaciones web sensibles al contexto [Kapitsaki et al., 2008].

A continuación se realizará un análisis de varias alternativas relevantes, las cuales introducen el uso de ciertos tipos de información de contexto en la web.

6.3.1 CAMB: Context-aware mobile browser

CAMB [Gasimov et al., 2010] es un navegador web HTML móvil sensible al contexto complementado con un framework de desarrollo de aplicaciones web. El objetivo de CAMB es adaptar las interfaces de usuario de las páginas web HTML dependiendo de un conjunto predeterminado de situaciones o ambientes. Esta propuesta centra el empleo de la información de contexto en la presentación de las páginas web. CAMB no ha sido concebido para que los desarrolladores manejen la información de contexto de manera genérica, es decir no independiza la gestión de la información de contexto del objetivo de la plataforma. En esta plataforma un desarrollador no puede utilizar la información de contexto para cualquier finalidad, como por ejemplo en los procesos de la lógica de negocio de la aplicación.

Anteriormente varios autores ya habían realizado sistemas que utilizaban la información de contexto capturada por el teléfono móvil para modificar en consecuencia el aspecto gráfico o contenido de las aplicaciones [Nathanail et al., 2007], la mayoría de estas propuestas no se centraban en las aplicaciones web móviles sino en las aplicaciones nativas [Lemlouma & Layaida, 2004 ; Eissele et al., 2009].

6.3.2 CRITERIA

Noltes [Noltes, 2008] propone un sistema compuesto por dos plug-in, uno en el lado del cliente y otro en el lado del servidor web. El plug-in cliente se ejecuta en el teléfono móvil y se encarga de capturar diversos tipos de información de contexto utilizando los elementos hardware del dispositivo móvil, la información capturada se encapsula y se remite al plug-in instalado en el servidor. Esta información de contexto se envía utilizando un protocolo específico basado en HTTP. El plug-in del servidor recibe y procesa la información de contexto, posteriormente la información puede emplearse para diversas tareas.

CRITERIA posee algunos aspectos interesantes, como el de otorgar a los usuarios el control total sobre su información de contexto y la divulgación de la misma. Otros navegadores web sensibles al contexto envían esta información de manera transparente al usuario. Como contrapunto, CRITERIA no contempla un conjunto demasiado extenso de tipos de información de contexto. La necesidad de utilizar dos plugins dependientes de plataforma, uno en el lado del servidor y otro en el del cliente no convierte a esta propuesta en una solución demasiado portable, tampoco la hace fácil incluirla en aplicaciones web ya desarrolladas ya que la arquitectura de la aplicación web pasa a verse condicionada por la integración con el plug-in.

6.3.3 SENSE-SATION

La plataforma SENSE-SATION [Shirazi et al., 2010] facilita el desarrollo de aplicaciones móviles y aplicaciones web basadas en comunidades y redes sociales. La peculiaridad de esta plataforma es que ofrece mecanismos para que las aplicaciones desarrolladas puedan tener acceso a los sensores y funcionalidades propias de los teléfonos móviles, como sensores, cámara de fotos, listines telefónicos, enviar SMS, etc. La plataforma está compuesta por un entorno que se ejecuta en el propio teléfono móvil cliente y una aplicación web propia gestionada por los desarrolladores de SENSE-SATION.

El cliente móvil provee mecanismos para la recolección y el manejo de la información de contexto. Este cliente es una aplicación software que se pone en marcha de manera automática al encender el teléfono y que, principalmente, se ejecuta en segundo plano. El cliente es capaz de almacenar varios tipos de información de contexto y cuenta con una interfaz que permite al usuario configurar aspectos sobre la privacidad.

La plataforma web de SENSE-SATION presenta una API sencilla, que permite recuperar la información de contexto capturada desde teléfono utilizando pocas peticiones. Abstrae la gestión de los sensores del teléfono utilizando sensores virtuales, por lo que el desarrollo de las aplicaciones web que utilizan esta plataforma resulta relativamente simple.

Esta plataforma cuenta con numerosos aspectos destacables, entre los que cabe mencionar: el elevado número de tipos de información de contexto que maneja y la simplicidad del API que permite gestionar la información de contexto. El aspecto más cuestionable de la plataforma es la necesidad de utilizar la aplicación web propia de SENSE-SATION como vinculo de unión entre el dispositivo móvil y las aplicaciones desarrolladas. Este aspecto genera lentitud, pérdida de control y una fuerte dependencia

que puede no resultar adecuada para algunos tipos de aplicaciones web. Por otro lado, el acceso a los mecanismos hardware de los dispositivos suele acarrear un fuerte consumo de recursos de computación y batería. La utilización de una aplicación que se ejecuta continuamente, incluso momentos en los que no se requiere ningún intercambio de información de contexto, puede no resultar demasiado eficiente.

6.3.4 CAB: Context-aware web browser

CAB Context-aware web browser [Coppola et al., 2010] surge como evolución de la propuesta MoBe [Coppola et al., 2005], un framework para el desarrollo de aplicaciones móviles sensibles al contexto. El objetivo de este navegador web sensible al contexto es utilizar la información de contexto capturada por el dispositivo móvil cliente para permitir realizar búsquedas refinadas de contenidos web. Aunque en un futuro plantean ampliar los objetivos de la propuesta para adaptarla al intercambio de información de contexto en las redes sociales y otro tipo de aplicaciones Web 2.0. El objetivo de este navegador no es ofrecer un mecanismo genérico que permita a los desarrolladores de aplicaciones web manejar la información de contexto capturada por el dispositivo cliente.

CAB aplica a la información de contexto técnicas de inteligencia artificial para conseguir filtrados eficientes que reduzcan el ancho de banda que se dedica al intercambio de la información de contexto. La propuesta tiene varios aspectos destacables. El diseño genérico que se ha realizado del componente navegador web, permite que éste pueda ser portado de manera sencilla a otras plataformas móviles diferentes de la actual J2ME. Además el navegador contempla diversos aspectos de privacidad y seguridad.

6.3.5 Geolocation API Specification

El interés por la utilización de la información de contexto en la web no se limita estrictamente a las propuestas de carácter científico. Hace varios años, surgieron las primeras propuestas para dotar al lenguaje Javascript de mecanismos que le permitieran solicitar ciertos tipos de información de contexto al dispositivo cliente. En marzo del 2012 el W3C convirtió en una recomendación la propuesta "Geolocation API Specification" [Geolocation, 2012], esta API que permite a las aplicaciones web obtener vía script acceso a la ubicación del dispositivo cliente. Aunque el Geolocation API Specification puede utilizarse para obtener la ubicación de varios tipos de dispositivos, son los dispositivos móviles los que aportan una mayor potencia a esta funcionalidad ya que utilizan sus sistemas GPS para estimar una posición bastante exacta. El W3C también ha publicado la especificación "DeviceOrientation Event Specification" [DeviceOrientation, 2011] que propone un API Javascript para obtener la orientación del dispositivo cliente, así como obtener la medición de los sensores de movimiento, aceleración y rotación. Actualmente estas especificaciones cuentan con soporte en muchos de los navegadores web móviles más populares.

6.3.6 Protocolos

Un enfoque totalmente divergente utilizado en el intercambio de información de contexto con las aplicaciones web es el uso de protocolos específicos. Existen varios protocolos que permiten la gestión y el intercambio de un conjunto reducido de tipos de información de

contexto, sobre todo la ubicación. La mayor parte de estos protocolos han sido diseñados específicamente para promover el intercambio de información de contexto entre las aplicaciones web y los navegadores web. Algunos de los protocolos más relevantes son: Secure User Plane location [Goze et al., 2007] Mobile Location Protocol [Mobile, 2002] Presence Information Data Format [Sugano et al., 2004] y HTTP Enabled Location Delivery [Barnes, 2008]. Las características más destacables de estos protocolos son la seguridad y el rendimiento. El aspecto negativo más notorio de estos protocolos respecto a los objetivos de esta tesis, es que todos ellos manejan un conjunto de tipos de información de contexto muy reducido, principalmente la ubicación.

6.3.7 Plataformas de aplicaciones centradas en Web

Las "Web-Centric Application platforms" plataformas centradas en web comparten ciertos aspectos con las aplicaciones web sensible al contexto [Gossweiler et al, 2011]. El uso y funcionalidad de estas plataformas se ha ampliado considerablemente desde la implantación de HTML 5 [Hickson, 2011] y el WebKit [WebKit, 2011]. PhoneGap [PhoneGap, 2011] es una de las plataformas de aplicaciones centradas en la web más populares y maduras, permite desarrollar aplicaciones móviles nativas empleando las tecnologías web HTML5, CSS y Javascript. PhoneGap utiliza un conjunto de librerías propias que soportan de manera parcial o total el acceso a la mayoría de los componentes hardware del dispositivo móvil. El nivel de acceso a varios elementos hardware del dispositivo que se puede llegar a conseguir en las aplicaciones desarrolladas bajo la plataforma PhoneGap resulta similar a la que ofrecen las APIs nativas. PhoneGap permite controlar el acelerómetro, el micrófono, la cámara de fotos, la localización y la brújula. En menor medida, y haciendo uso de plugins específicos, permite gestionar de manera simple algunos de los elementos hardware de comunicación del dispositivo.

Concretamente las aplicaciones desarrolladas con PhoneGap pueden ser exportadas a diversas plataformas móviles como: iOS, Android, WebOs, Windows Phone y Symbian.

Uno de los aspectos positivos mas señalado de PhoneGap es el gran número de tipos de información de contexto que maneja, mucho mayor que el de la mayoría de los sistemas similares. La utilización de las tecnologías web en el desarrollo permite ahorrar costes. A pesar de ser una plataforma muy completa PhoneGap no ha sido diseñada en base a los mismos objetivos que esta tesis ya que no presenta un sistema que ofrezca un buen soporte a los elementos hardware de comunicación de los dispositivos. PhoneGap tampoco hace énfasis en el acceso ágil y rápido a las aplicaciones, el proceso de instalación y gestión es muy similar al de las aplicaciones nativas.

6.3.8 Conclusiones

Desde hace años las aplicaciones web móviles se han utilizado como alternativa para reducir las consecuencias negativas derivadas de la heterogeneidad entre plataformas móviles, ya que las aplicaciones web pueden ser utilizadas casi en la totalidad de las plataformas móviles. El desarrollo de las aplicaciones web móviles respondía a objetivos de reducción de coste y esfuerzo: una única aplicación podía ser ejecutada en todas las plataformas, era más fácil de mantener y en la mayoría de los casos el proceso de desarrollo seria más rápido que el de una aplicación nativa.

Varias alternativas web analizadas en este apartado cumplen con objetivos significativos de esta tesis. Estas aplicaciones web móviles se pueden construir de manera simple, son ligeras y multiplataforma. En general las tecnologías web disponen de muchas herramientas y componentes reutilizables, como gestores de contenidos, frameworks, etc. Estas posibilidades unidas a la simplicidad de los lenguajes web, favorecen que en gran parte de los casos el desarrollo de aplicaciones web sea más rápido que el de las aplicaciones nativas. Las aplicaciones web no están sujetas a los criterios y normas que cada fabricante impone en sus tiendas de aplicaciones (Google Play, App Store, BlackBerry App World, etc), por este motivo las aplicaciones web pueden pasar a producción o ser actualizadas de manera directa. Las aplicaciones distribuidas en general plantean un acceso ligero y dinámico, en condiciones normales la carga de un sitio web consume tiempos relativamente bajos. El enfoque de las aplicaciones distribuidas es óptimo para conseguir suprimir los procesos de instalación y configuración requeridos en las aplicaciones nativas. De esta forma se consigue aumentar el dinamismo en aplicaciones que se utilizan de manera puntual, a la vez que no se sobrecarga innecesariamente el dispositivo móvil con multitud de aplicaciones.

Gran parte de las alternativas analizadas no cumplen con uno de los objetivos fundamentales de esta tesis: no facilitan la gestión de la mayoría de los elementos hardware del dispositivo móvil. Casi todos los sistemas analizados solo consiguen manejar un pequeño grupo de tipos de información de contexto, principalmente la ubicación. La gestión de los elementos hardware del dispositivo resulta imprescindible para establecer comunicaciones con dispositivos electrónicos cercanos y captura de la información de contexto. Por otro lado, las alternativas web que sí que cumplen con este segundo objetivo no cumplen con los primeros, pues solamente pueden ser ejecutadas en plataformas y entornos con requerimientos concretos.

PARTE III

DESARROLLO DE LA PROPUESTA

CAPÍTULO 7

OBJETOS VIRTUALES V.1.0 STDD

7.1 INTRODUCCIÓN

La primera versión de los objetos virtuales comenzó a desarrollarse a mediados del 2009. Desde su inicio hasta principios del 2011 la propuesta ha estado en desarrollo y en constante evolución.

¿Qué es un objeto virtual?

Al término "objeto virtual" se le atribuyen diferentes definiciones dentro de distintos ámbitos relacionados con las ciencias de la computación (sistemas e-learning, realidad virtual, etc.).

Uno de los pilares de Internet de las cosas reside en la comunicación y sincronización entre objetos físicos, y de estos con los usuarios. Los objetos físicos pueden estar formados por una parte virtual, como por ejemplo la aplicación software que se ejecuta en el sistema embebido de un electrodoméstico. No obstante también pueden ser meramente físicos, como una botella de agua etiquetada con un código Q2. Los dos tipos de objetos pueden formar parte de un sistema clasificado como perteneciente a Internet de las cosas.

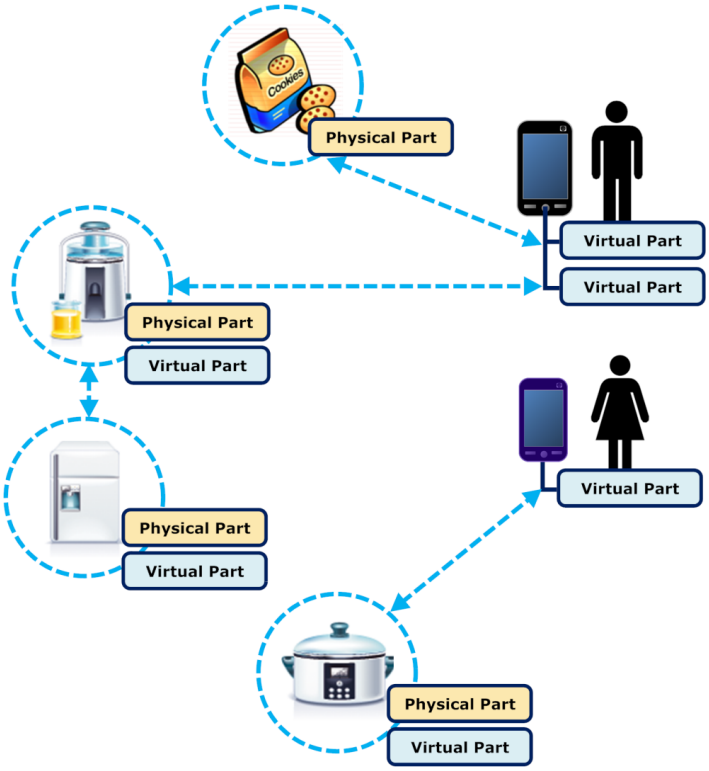

Figura 7.1 – Entorno de interacción entre objetos (Ilustraciones zcool.com.cn)

Sin embargo, muchos sistemas emplean dispositivos como "contenedores genéricos" que ejecutan aplicaciones, las cuales gestionan la interacción con otros objetos (Fig.7.1). Nos referimos a dispositivos como contenedores genéricos cuando la parte física del dispositivo no tiene una gran relevancia en el proceso de interacción, únicamente debe cumplir unos requisitos mínimos. Por ejemplo: un televisor inteligente ejecuta aplicaciones software que son dependientes de su parte física (pantalla, conexiones, altavoces, etc), sin embargo este televisor puede interactuar con otro dispositivo electrónico que ejecuta una aplicación de control remoto, esta aplicación tendrá un funcionamiento similar independientemente de que el dispositivo electrónico controlador sea un ordenador portátil, un teléfono móvil, una PDA, etc. En este caso basta con que el dispositivo contenedor cumpla con los requerimientos técnicos (capacidad de computación, elementos de comunicación, etc) y utilice una parte virtual (aplicación software) adecuada. A este componente software capaz de hacer que un dispositivo interactué con objetos o elementos físicos se la ha denominado **objeto virtual**. Un dispositivo no específico que contiene un objeto virtual se puede convertir en un integrante activo de un sistema que basa una parte de su funcionalidad en la interacción con objetos físicos [Espada et al., 2011b].

En esta investigación se ha definido el término objeto virtual dentro del ámbito de Internet las cosas como:

"Componente software que se combina con un dispositivo electrónico con el objetivo de desarrollar una determinada funcionalidad, la cual puede llegar a hacer las veces de un objeto físico, implicando la interacción con objetos físicos, dispositivos electrónicos u otros elementos del entorno".

La primera versión de la arquitectura diseñada que habilita la implementación de objetos virtuales se ha denominado: **Objetos Virtuales Servicios Temporales Dependientes de Dispositivo – Objetos Virtuales STDD.**

7.2 MODELO

7.2.3 Estructura de los objetos virtuales DDTS

De manera similar a una aplicación convencional la estructura propuesta para el desarrollo de los objetos virtuales se divide en tres capas:

- **Capa de aplicación**: en esta capa se incluyen los mecanismos necesarios para que el objeto virtual pueda interactuar con usuarios y aplicaciones. La forma clásica de interacción con los usuarios es mediante una interfaz gráfica. La interacción con otras aplicaciones u objetos suele hacerse a través de un catálogo de servicios.

- **Capa de lógica de negocio**: en esta capa se encuentra toda la lógica de negocio, el código ejecutable de las acciones o servicios que el objeto puede realizar.

- **Capa de acceso a datos**: almacenan los datos necesarios para operar con el objeto virtual.

Un objeto virtual STDD resulta conceptualmente similar a una aplicación nativa: código ejecutable, interfaces de usuario, datos, etc. La principal diferencia radica en que los objetos virtuales STDD no se instalan y ejecutan sobre el sistema operativo del dispositivo móvil, sino que son directamente interpretados por una aplicación especialmente diseñada para este propósito. Este enfoque permite orientar a los objetos virtuales hacia la consecución de los objetivos propuestos en esta tesis:

- Independiza el código fuente del objeto virtual de la plataforma móvil concreta, haciendo posible que un mismo objeto pueda ser ejecutado en distintas plataformas.

- Sustituye los procesos clásicos de instalación y configuración, consiguiendo una mejor adaptación de los objetos virtuales a sistemas que requieran la interacción ocasional con otros objetos físicos o dispositivos.

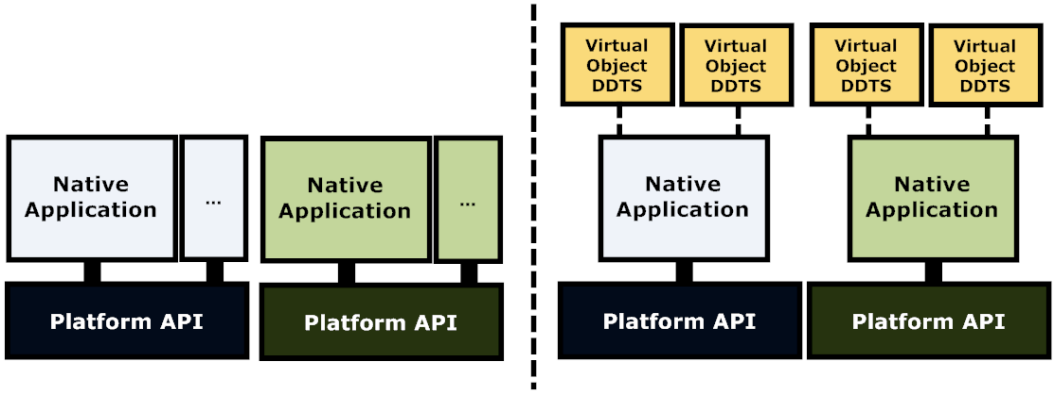

Figura 7.1 – (1) Las aplicaciones nativas se instalan en el sistema operativo del dispositivo móvil. (2) El modelo propuesto para los objetos virtuales no depende de la plataforma ni instalación sobre el sistema operativo.

7.2.4 Dispositivo contenedor

El objeto virtual no es una aplicación típica que se instala y ejecuta sobre el propio sistema operativo del dispositivo, sino que requiere de una aplicación específica que lo interprete Esta aplicación contenedora deberá estar instalada en un dispositivo móvil que cumpla con los requisitos de computación y capacidades de comunicación.

En los últimos años los Smartphones se han convertido en elementos de uso frecuente para la mayoría de personas. El avance tecnológico en estos dispositivos es muy rápido, cada vez cuentan con más mejoras técnicas que han permitido ampliar su funcionalidad hasta el punto de convertirlos en pequeños computadores. El consumo de servicios a través del teléfono móvil tiene una gran importancia en países como Japón y este tipo de dispositivos prometen ser una pieza clave en la evolución de Internet de las cosas. Aunque otros dispositivos electrónicos serían capaces de ejecutar y gestionar objetos virtuales, el teléfono móvil es el que tiene unas características más adecuadas para ser el destinatario principal.

7.3 ARQUITECTURA

En este apartado se hará una introducción sobre varios conceptos y elementos, con el objetivo de ofrecer una idea global sobre las partes del sistema propuesto y la función que cumple cada elemento.

7.3.1 Visión general

Para que los objetos virtuales puedan ser consumidos por cualquier tipo de dispositivo se necesita:

- Objetos virtuales en un formato conocido, preferiblemente normalizado.
- Una aplicación instalada o integrada en los dispositivos la cual haya sido construida teniendo en cuenta la especificación de los objetos virtuales y que permita:
    - Interpretar, cargar y ejecutar cualquier objeto virtual estándar.
    - Almacenar, ordenar y gestionar los objetos virtuales.
    - En caso de que haya sido programado en la lógica del objeto, permitir la ejecución de forma remota de los objetos virtuales alojados en el dispositivo. Es decir "publicar" el objeto para que pueda ser utilizado por otros usuarios o aplicaciones.
    - Descubrir y permitir la ejecución de manera remota de otros objetos virtuales, es decir conectarse a otros objetos y usarlos.

7.3.2 Objetos Virtuales STDD: Servicios Temporales Dependientes de Dispositivo

Al formato de objetos virtuales que se presenta en este capítulo lo hemos denominado de manera técnica como: **Objetos virtuales STDD** (Servicios Temporales Dependientes de Dispositivo). En el capítulo anterior se definió el término objeto virtual dentro del ámbito de Internet de las cosas. La dependencia de dispositivo se debe a que a diferencia del resto de objetos físicos y dispositivos electrónicos que forman parte de los sistemas de Internet de las cosas, el modelo propuesto para los objetos virtuales requiere de un dispositivo que actué como contenedor; en todo momento los objetos virtuales necesitan depender de un dispositivo para formar parte de un sistema.

7.3.3 Gestor STDD

El Gestor de STDD, es la aplicación encargada de administrar los objetos virtuales en el dispositivo contenedor. Debe estar diseñado específicamente para poder ser instalado y

ejecutado en el dispositivo en cuestión, teniendo en cuenta sus características, sistema operativo, lenguajes de programación compatibles, etc.

Un Gestor STDD debe poseer al menos las siguientes capacidades:

- **Cargar, Interpretar y ejecutar** cualquier objeto virtual normalizado que haya sido introducido en el dispositivo.

  El primer paso para comenzar a utilizar un objeto virtual una vez se dispone de su código es cargarlo en el Gestor STDD. Cuando el objeto virtual se encuentra cargado el usuario puede seleccionarlo y comenzar a interpretarlo, el resultado será una interfaz gráfica rica por medio de la cual el usuario realizará invocaciones a las acciones implementadas en la lógica del objeto virtual.

- **Almacenar, ordenar y gestionar**. A menudo se utilizarán varios objetos virtuales simultáneamente, el Gestor STDD debe ofrecer mecanismos para almacenar, organizar y gestionar los objetos virtuales cargados en el dispositivo.

  Los objetos virtuales que hayan sido cargados en el dispositivo deben ser almacenados y mostrarse de manera ordenada al usuario para que este pueda gestionarlos, es decir: eliminarlos, copiarlos y transferirlos siempre que la naturaleza del objeto lo permita.

- **Publicar objetos virtuales:** en caso de que haya sido especificado en las propiedades del objeto virtual, permitir la ejecución de forma remota. El Gestor STDD debe ofrecer mecanismos para que los objetos virtuales puedan ser consumidos por otros usuarios o aplicaciones.

  El Gestor STDD contendrá mecanismos específicos para permitir la ejecución remota del objeto virtual y publicar sus servicios, normalmente estas acciones se realizarán vía Bluetooth, Internet, Wi-Fi, etc.

- **Descubrimiento y ejecución remota:** el Gestor STDD debe ser capaz de descubrir los objetos virtuales que otros dispositivos publican. Una vez descubiertos los objetos virtuales, podrían ser ejecutados de manera remota (si se ha especificado). El código del objeto se ejecutará en el dispositivo servidor y este intercambiará un flujo de datos con el dispositivo cliente.

7.3.4 Consumo de objetos virtuales

Un objeto virtual puede ser consumido por una persona de dos maneras:

- **De manera local:** Una vez el objeto está cargado en el dispositivo, el usuario puede seleccionarlo e interpretarlo; el resultado será una interfaz gráfica por medio de la cual el usuario realizará invocaciones a las acciones implementadas en el código ejecutable del objeto.

- **De manera remota:** descubriendo objetos virtuales que se encuentren cargados y activos en otros dispositivos. Una vez descubiertos los objetos virtuales podrían ser ejecutados de manera remota, de forma que el código del objeto se ejecutaría en el dispositivo servidor y este intercambiaría un flujo de datos con el dispositivo cliente.

Existe otra forma de consumir objetos virtuales, consistente en invocar directamente los servicios que este ofrece, los cuales se especifican en su catálogo de servicios, esta sería la forma de interacción que utilizaría una aplicación para comunicarse con el objeto virtual.

7.3.4.1 Ejecución local

La ejecución local se produce cuando un usuario abre un objeto virtual que ha sido cargado en su propio dispositivo, consta de los siguientes pasos (Fig.7.2):

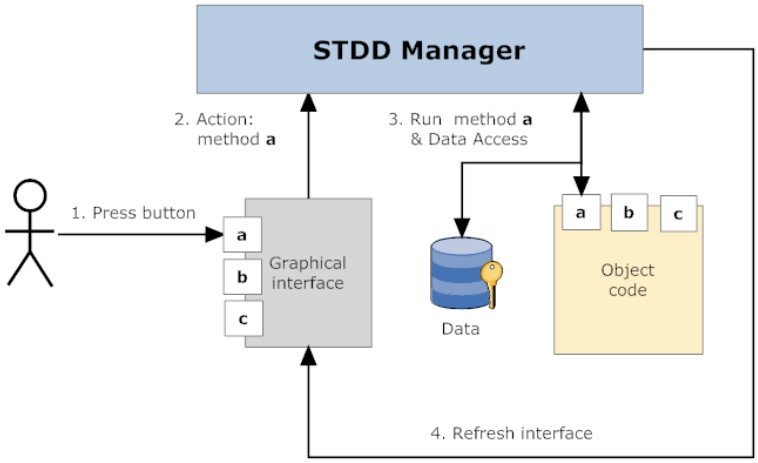

Figura 7.2 - Diagrama de ejecución de un objeto virtual STDD de manera Local.

1. Al abrir uno de los objetos virtuales cargados en el dispositivo, se carga en pantalla su interfaz gráfica principal. Existen dos tipos de interfaces gráficas: las privadas que se muestran cuando el objeto se ejecuta de manera local y las públicas que lo hacen cuando se ejecuta de manera remota. Si el usuario manipula alguno de los elementos de la interfaz, como por ejemplo un botón, pueden lanzarse eventos que requieran la ejecución de código.

2. Cada vez que se lanza un evento en la interfaz gráfica el Gestor STDD recoge la información correspondiente a la acción que debe ejecutarse, nombre del método, parámetros que recibe y fichero de código donde se encuentra.

3. El Gestor STDD se sincroniza con el almacén de datos del objeto virtual, busca y ejecuta el método en el archivo de código que le ha sido indicado. En caso de que el método tenga algún tipo de retorno este es almacenado.

4. El último paso de la ejecución consiste en refrescar la interfaz gráfica que se le muestra al usuario, los cambios en la interfaz gráfica tanto privada como pública se realizan desde el código ejecutable, por lo que cada vez que se ejecuta un método puede que se hayan realizado cambios; el Gestor STDD pregunta al código a través de unos métodos predefinidos cual es el estado actual de la interfaz gráfica privada. En el caso de que haya modificaciones, el Gestor STDD realiza la nueva carga de la interfaz, actualizando la pantalla que visualiza el usuario.

Durante la interacción de un usuario con un objeto virtual estos cuatro pasos se repiten en forma de bucle.

7.3.4.2 Descubrimiento

Una vez cargados en el Gestor STDD los objetos virtuales tienen dos estados:

- **Activo:** podrá ser consumido de manera local y de forma remota, por lo que resultará visible para cualquier dispositivo que se encuentre en el radio de cobertura.
- **Inactivo:** solamente podrá ser consumido de forma local.

El proceso de descubrimiento de servicios consta de las siguientes fases (Fig.7.3):

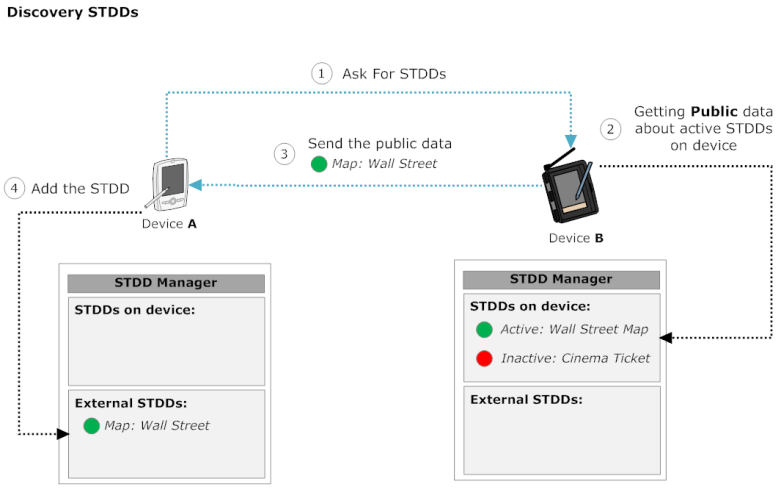

Figura 7.3 - El dispositivo A descubre un objeto virtual STDD activo que se encuentra alojado en el Dispositivo B.

1. El cliente, dispositivo **A**, envía una petición al servidor, dispositivo **B** solicitando una lista de los objetos virtuales STDD que se encuentran activos.
2. El servidor selecciona la parte pública de la información de aquellos objetos virtuales STDD que se encuentran activos. La parte pública consiste en:
    - Nombre.
    - Información sobre el propietario.
    - Icono del objeto virtual STDD.

El servidor envía la información y recursos seleccionados acerca de los objetos virtuales STDD activos.

3. El cliente procesa la información recibida desplegando en su lista de objetos virtuales STDD remotos los nuevos elementos que acaba de obtener.

7.3.4.1 Acceso remoto

Una vez descubiertos los objetos virtuales el dispositivo cliente los almacena en una lista, el proceso seguido para el consumo de objetos virtuales STDD de manera remota es el siguiente (Fig.7.4):

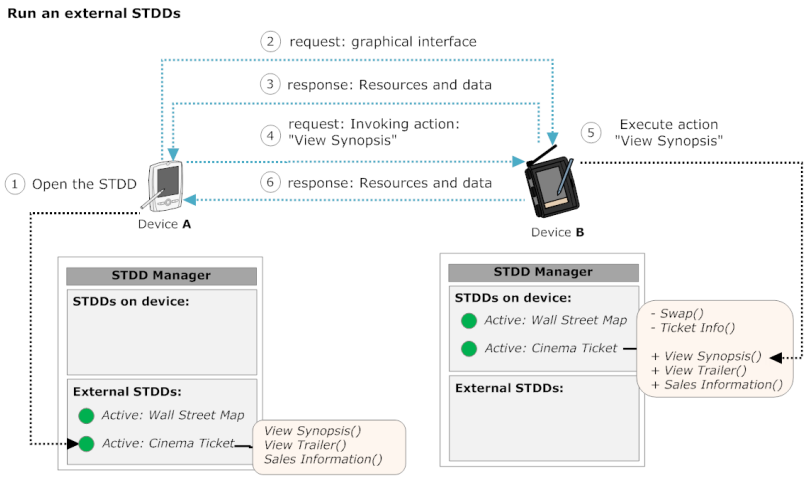

Figura 7.4 - El dispositivo A ejecuta de forma remota un objeto virtual STDD activo que se encuentra alojado en el Dispositivo B.

1. Cuando se abre a uno de los objetos virtuales que han sido previamente descubiertos, se informa al Gestor STDD de que es necesaria la interfaz pública del objeto "Cinema Ticket" alojado en el Dispositivo **B**.

2. El Gestor de STDD solicita al servidor, dispositivo **B,** la interfaz gráfica pública principal y los recursos asociados a ella (principalmente material multimedia).

3. El servidor envía todos los archivos necesarios para que el dispositivo **A** pueda interpretar la interfaz grafica. Una vez recibida la interfaz gráfica pública, el Gestor STDD la interpreta y la muestra por pantalla. Si el usuario manipula alguno de los elementos de la interfaz como por ejemplo un botón, pueden lanzarse eventos que requieran la ejecución de código, en este caso el código ejecutable no se encuentra en el dispositivo por lo que el Gestor STDD tendrá que trasladar la petición al dispositivo servidor.

4. Cada vez que se lanza un evento en la interfaz gráfica el Gestor STDD recoge la información correspondiente a la acción que debe ejecutarse, nombre del método y parámetros que recibe, posteriormente realiza una petición al servidor.

5. El Gestor STDD del dispositivo **B** recibe la petición para ejecutar la acción. De igual manera que cuando los objetos se ejecutaban localmente se sincroniza con el almacén de datos del objeto virtual, busca y ejecuta el método en el archivo de código que le ha sido indicado. En caso de que el método tenga algún tipo de retorno éste es almacenado.

6. Los cambios en la interfaz gráfica tanto pública como privada pueden ser provocados desde el código ejecutable, por lo que cada vez que se ejecuta un método puede que se hayan realizado cambios de interfaz gráfica. El Gestor STDD del dispositivo **B** envía el retorno del método (en caso de que lo haya) y la nueva interfaz pública en caso de que haya habido modificaciones en ella, para que sea cargada de nuevo en el cliente.

A continuación se explicará lo que ocurre desde que el cliente lanza un evento hasta que percibe el resultado de la acción asociada (Fig.7.5).

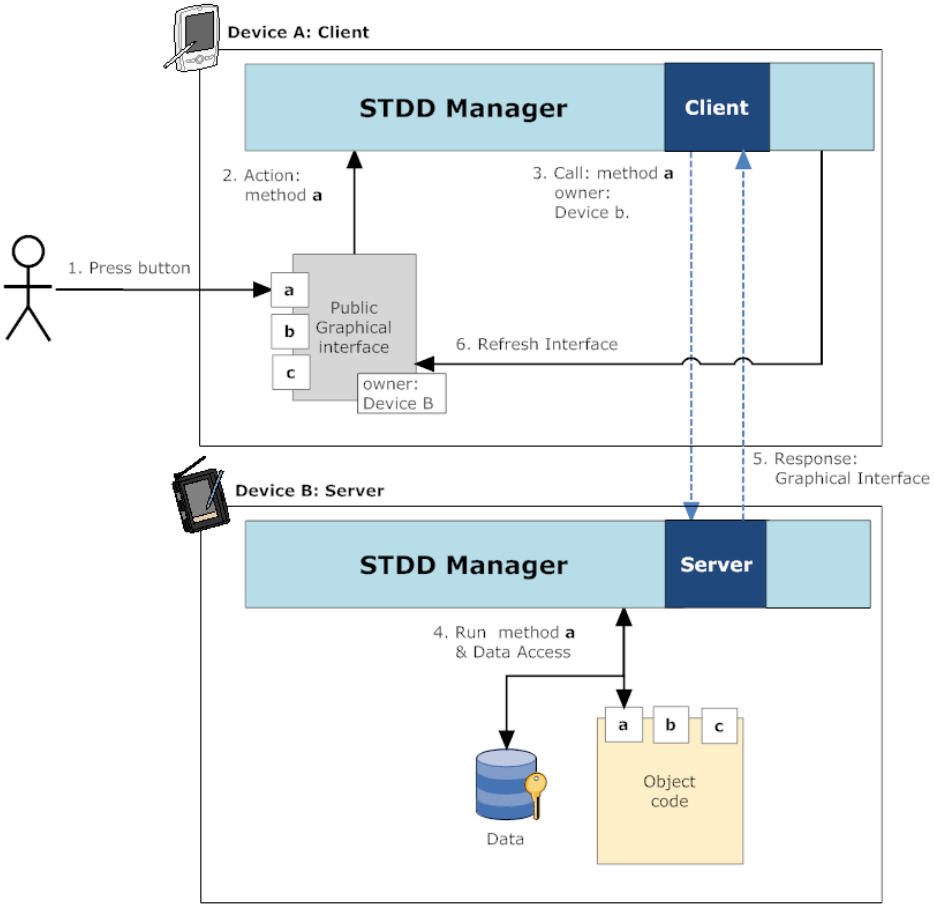

Figura 7.5 - El dispositivo A ejecuta una acción de forma remota en un objeto virtual STDD que se encuentra alojado en el Dispositivo B.

1. Al abrir uno de los objetos virtuales cargados en el dispositivo, se muestra por pantalla su interfaz gráfica pública principal, que ha sido enviada previamente junto con los recursos asociados. Si el usuario manipula alguno de los elementos de la interfaz, como por ejemplo un botón, pueden lanzarse eventos que requieran la ejecución de código.

2. Cada vez que se lanza un evento en la interfaz gráfica, el Gestor STDD recoge la información correspondiente a la acción que debe ejecutarse, nombre del método, parámetros que recibe, fichero de código donde se encuentra.

3. El Gestor STDD del cliente envía una llamada al Gestor STDD del servidor, indicándole que acción debe ejecutar.

4. El Gestor STDD del servidor se sincroniza con el almacén de datos del objeto virtual, busca y ejecuta el método indicado en la petición recibida. En caso de que el método tenga algún tipo de retorno éste es almacenado. Los cambios en la interfaz gráfica pública también se realizan desde el código ejecutable, por lo que cada vez que se ejecuta un método pueden realizarse cambios en ella. El Gestor STDD pregunta al código a través de unos métodos predefinidos cual es la interfaz gráfica pública actual.

5. El servidor envía a una respuesta al cliente, la cual contiene la interfaz gráfica pública que debe mostrar (si es que ha habido cambios) en consecuencia con la acción que acaba de ejecutarse.

6. Después de obtener la respuesta del servidor el cliente debe refrescar la interfaz gráfica que se le muestra al usuario, para ello el Gestor STDD carga la interfaz gráfica pública que acaba de recibir.

Durante la interacción de un usuario con un objeto virtual estos pasos se repiten en forma de bucle.

7.4 OBJETO VIRTUAL STDD

7.4.1 Estructura

La estructura del objeto virtual STDD está compuesta por un conjunto de archivos (Fig.7.6).

Tipo	Archivos:	Descripción
Descriptor	1	Fichero XML, contiene información acerca de la configuración, despliegue y ejecución del objeto.
Interfaces Gráficas	1..*	Ficheros XML de tipo interfaz gráfica STDD. Cada uno representa una pantalla.
privadas		Se utilizan en la ejecución local del objeto virtual STDD.
públicas		Se utilizan en la ejecución remota del objeto virtual STDD.
Interfaces		Interfaces para que las acciones del objeto virtual puedan ser accedidas por aplicaciones y otros objetos virtuales.
WSDL	0..*	Descripción de servicios web.
API	0..*	Describe acciones que pueden ser directamente invocadas por otros objetos virtuales STDD.
Código ejecutable	1..*	Ficheros que contienen el código ejecutable correspondiente a la lógica del objeto virtual STDD. El código puede suministrarse en varios lenguajes de programación para que pueda ser ejecutado en dispositivos de distintas características.
Almacén de datos	0..1	Fichero que tiene como misión encargarse de la persistencia de datos, sólo debe incluirse en el caso de que sea requerido.
Recursos	0..*	Pueden incluirse un número de recursos ilimitados. Por lo general serán recursos multimedia: imágenes, iconos y videos. Serán utilizados para completar el apartado gráfico del objeto virtual STDD.

Tabla 7.1 – Conjunto de ficheros que forman la estructura del objeto virtual STDD

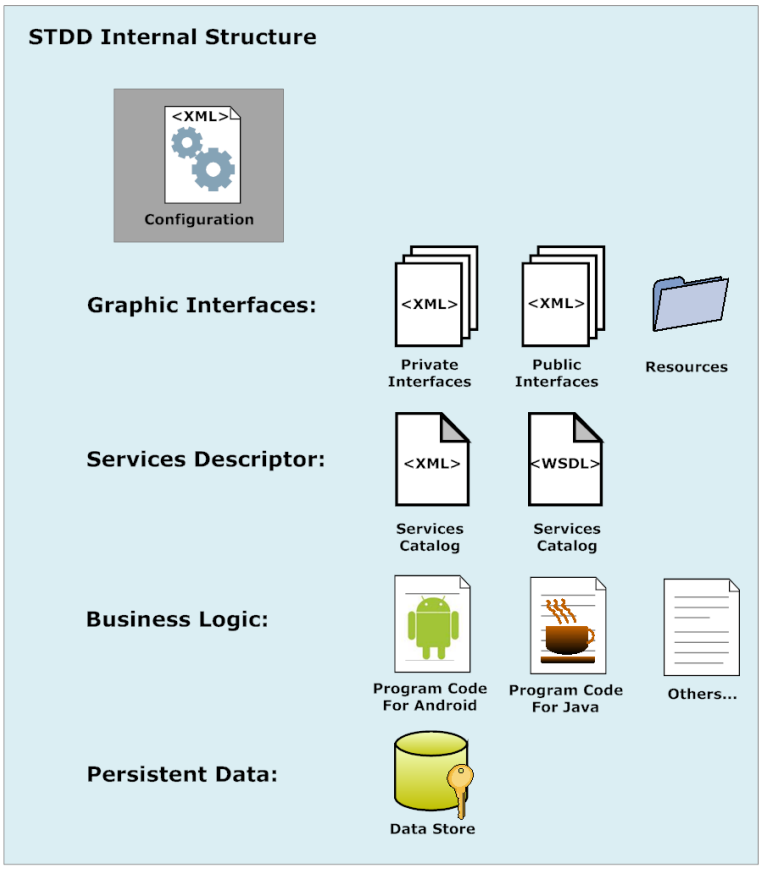

Figura 7.6 - Ejemplo de un conjunto de archivos que forman un objeto virtual STDD.

7.4.2 Propiedades generales

El fichero de configuración es un documento XML que contiene información acerca de la configuración, despliegue y ejecución del objeto virtual STDD.

Para cargar un objeto virtual STDD en un dispositivo es necesario abrir su fichero de configuración con el explorador de archivos a través del Gestor STDD.

A continuación se definen los campos obligatorios y optativos de dicho fichero:

Nombre	Obligatorio	Descripción
name	X	Nombre: cadena de texto. El nombre elegido debe resultar descriptivo para el objeto virtual STDD.
type		Tipo: cadena de texto (formato dominio). Se utiliza para que el Manager STDD y ciertas aplicaciones puedan agrupar o clasificar los objetos virtuales STDD.
id	X	Identificador del objeto virtual STDD: cadena de texto.
unique		Único: booleano, indica si puede o no existir otro objeto virtual STDD con el mismo identificador.
transferable		Transferible a otros dispositivos: booleano, indica si el objeto virtual puede transferirse a otro dispositivo. Si se omite este campo la propiedad tomara el valor NO.

expiration		Expiración: puede que ciertos objetos virtuales STDD deban anularse pasada una fecha. Si se omite este campo no habrá fecha de expiración
edit		Editable: booleano, indica si pueden editarse los ficheros o recursos que componen el objeto virtual STDD. Si se omite este campo la propiedad tomara el valor NO.
copy		Copiable: booleano, indica si el objeto virtual STDD puede duplicarse. Si se omite este campo la propiedad tomara el valor NO.
interface	X	Referencia a las distintas interfaces que puede tener el objeto virtual STDD, debe tener al menos una.
private		Ruta de fichero .XML de tipo interfaz gráfica STDD. Esta interfaz gráfica sería la primera pantalla en cargarse cuando se quiera ejecutar el objeto virtual STDD desde el propio dispositivo que lo contiene. La ausencia de este atributo indica que el objeto virtual no puede ejecutarse de manera local.
public		Ruta de fichero .XML de tipo interfaz gráfica STDD. Esta interfaz gráfica sería la primera pantalla en cargarse cuando se quiera ejecutar el objeto virtual STDD de manera remota desde un cliente. La ausencia de este atributo indica que otros dispositivos no pueden ejecutarlo de manera remota.
WSDL		Ruta de fichero .wsdl Describe los servicios web que ofrece el objeto virtual STDD.
API		Ruta de fichero .XML de tipo API STDD. Describe los servicios que ofrece el objeto virtual STDD y que pueden ser invocados desde otros objetos virtuales.
code		Indica las rutas a los ficheros de código que pueden existir, al menos debe haber uno. El mismo código ejecutable puede suministrarse en varios lenguajes de programación.
android		Ruta código ejecutable Java para sistemas Android.
java		Ruta código ejecutable Java.
iPhone		Ruta código ejecutable Objetive-c.
data		Ruta del archivo de datos. En caso de que el código ejecutable utilice persistencia debe especificarse la ruta del almacén de datos.
main		Clase principal: paquete.clase. El código ejecutable puede estar compuesto por una estructura de varios paquetes y clases. Si se incluye este atributo su contenido se tomará como referencia a la hora de realizar invocaciones a métodos en las que no haya sido especificado el paquete y clase.
icon		Ruta del icono .png que se asociara al objeto virtual tanto para la ejecución local como remota. En ausencia de este atributo se asignara el icono por defecto.

Tabla 7.2 – Campos obligatorios y opcionales del fichero de configuración de un objeto virtual STDD.

A continuación se presenta un posible fichero de configuración para un objeto virtual STDD.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<STDD>
    <name>Cinema Ticket: Avatar</name>
    <type>www.CinemaTicketSTDD.com</type>
    <id unique="YES">12A78CHJ12</id>
    <transferable>YES</transferable>
    <expiration>10-12-2010</expiration>
    <editable>NO</editable>
    <copy>NO</copy>
    <interface
        private="privateInt.xml"
        public="publicInt.xml"
        wsdl="interface.wsdl"
        api="methods.xml"/>
    <code
        android="Ticket12A78CHJ12_android.apk"
        java="Ticket12A78CHJ12_j2me.jar"/>
    <data>data.obj</data>
    <main>com.Ticket.TicketAvatar</mainClass>
    <icon>icon_ticket.png</icon>
</STDD>
```

7.4.3 Interfaces

La misión de las interfaces es que el objeto virtual pueda comunicarse con otros objetos, usuarios y aplicaciones.

Se dispone de tres tipos de interfaces distintas que pueden incluirse en el objeto virtual STDD: Interfaces gráficas, WSDL y API-STDD. Cada una de ellas tiene un propósito específico y no tienen por qué contener los mismos servicios. No resulta obligatorio introducir los tres tipos de interfaces, habrá que adaptarse a la naturaleza del objeto virtual que se esté modelando.

Interfaces gráficas: es la forma principal de interacción con el objeto virtual, proporciona un entorno visual para permitir la comunicación entre usuario y objeto virtual (Fig.7.7). A través de la interacción por pantalla el usuario puede comunicarse con el objeto y desencadenar acciones que llevarían a la ejecución del código asociado.

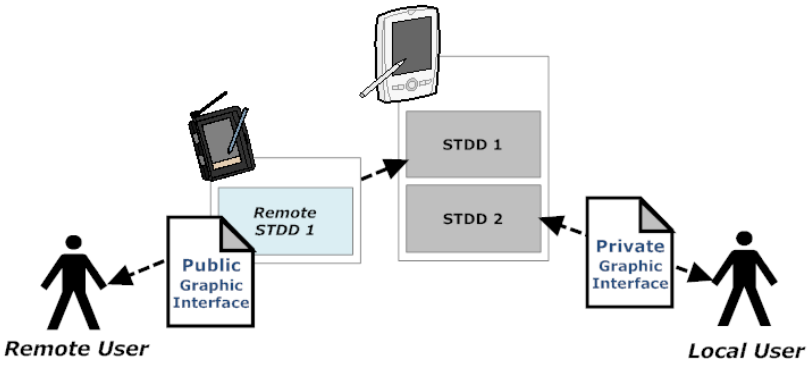

Figura 7.7 - La interacción con los usuarios locales y remotos se realiza por medio de las Interfaces gráficas privadas y públicas.

Servicios Web - WSDL: son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web. Su misión es mostrar los servicios que el objeto desea publicar para que otras aplicaciones puedan acceder a ellos a través de Internet. El servidor web implementado en el Gestor STDD pública el catálogo de servicios para que estos puedan ser invocados por otras aplicaciones, siempre y cuando el usuario propietario del objeto virtual lo autorice (Fig.7.8).

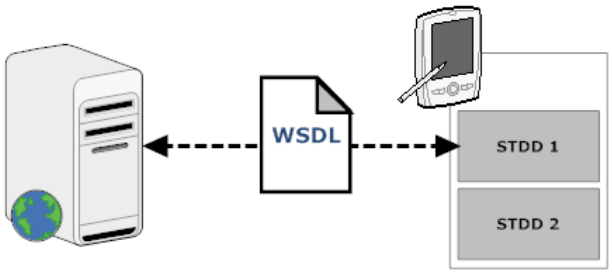

Figura 7.8 - Las aplicaciones pueden conocer las acciones asociadas al objeto virtual a través de su catálogo de servicios WSDL.

APIs: es un descriptor de servicios con un propósito similar al del WSDL, con la diferencia de que los servicios públicos descritos pueden ser consumidos únicamente por otros objetos virtuales STDD que se encuentren cargados en el dispositivo (Fig.7.9).

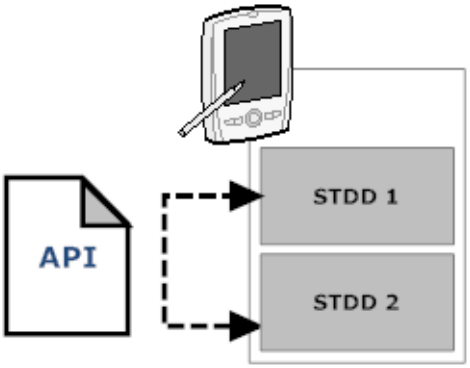

Figura 7.9 - Los objetos virtuales STDD se pueden comunicar entre sí utilizando las acciones descritas en sus APIs.

7.4.4 Interfaces gráficas

Este tipo de interfaces tienen como objetivo comunicar a las personas con los objetos virtuales. Cada fichero XML describe exclusivamente una pantalla, la cual puede mostrarse durante la ejecución del objeto virtual STDD. Durante la ejecución es posible que se produzcan varias transiciones entre pantallas, por lo que pueden incluirse varios ficheros de interfaz gráfica.

El formato utilizado para la definición de las interfaces gráficas es igual para las de tipo privado y público.

7.4.4.1 Sintaxis

Para describir los elementos gráficos y su comportamiento se ha partido de una versión reducida del lenguaje de definición de interfaces utilizado por el sistema Android, el cual permite describir interfaces gráficas de forma relativa. Se han introducido nuevas propiedades en los elementos gráficos que permiten asociar eventos con acciones incluidas en el código ejecutable del objeto virtual STDD.

La manera de definir la disposición y expresar la jerarquía de elementos en la pantalla es mediante un archivo XML, similar a un archivo HTML. Cada elemento en el XML representa un objeto que se mostrará en la pantalla. A cada tipo de elemento se le puede asociar una serie de propiedades como márgenes, colores, tamaño, posición relativa, etc.

7.4.4.1.1 Elementos gráficos comunes

Existe un gran conjunto de elementos que pueden ser utilizados para componer las interfaces gráficas, los más comunes son los siguientes:

Nombre	Descripción
Layout	La pantalla: es el objeto principal que contiene al resto.
TextView	Cadena de texto.
EditText	Cajón editable donde el usuario puede introducir texto.
ImageView	Imagen.
Button	Botón
RadioButton	Botón selección excluyente.
CheckBox	Botón selección Activo / Inactivo.

Tabla 7.4 – Elementos gráficos comunes.

Los elementos gráficos pueden ser combinados con una serie de propiedades:

Propiedad	Descripción
Elemento	
id	Identificador único, cadena que identifica de manera única a un elemento, este identificador se utilizará para hacer referencia al objeto desde otras propiedades.
text	Texto asociado al elemento. Esta propiedad solo tiene valor en ciertos elementos, TextView, Button, RadioButton… no lo tiene por ejemplo en un ImageView.
color	Color de texto en formato RGB (0-256, 0-256, 0-256). Si no se específica esta propiedad el color por defecto será el blanco.
size	Tamaño de la fuente del texto, solo está activo en aquellos tipos de objetos que contienen elementos textuales.
action/onClick	Especifica que acción debe invocarse al pulsar sobre el elemento. Aunque lo más común es que este atributo se aplique sobre elementos de tipo Button, también puede incluirse en otro tipo de elementos.
onTouch	Especifica que acción debe invocarse al pasar el foco por el área del elemento.
file	Ruta del fichero imagen, exclusivo para objetos del tipo ImageView.
xcrollBars	Introducir barras de Scroll en el elemento.
width	Ancho del elemento.
height	Largo del elemento.
background	Fondo del elemento, puede asignarse la ruta de una imagen o un color en formato RGB (0-256, 0-256, 0-256). Si no se específica este atributo, el color por defecto será el negro.
visible	Define si el elemento esta visible en pantalla. Toma un valor booleano, por defecto si se omite tomara el valor true.
enabled	Define si el elemento esta activo. Toma un valor booleano por defecto si se omite toma el valor true.
Posicionamiento	
margin	Márgenes del elemento, se especifican de manera relativa usando porcentajes.
marginTop	Margen parte superior.
marginRight	Margen a la derecha.
marginLeft	Margen a la izquierda.
marginBottom	Margen parte inferior.
padding	Relleno del elemento, se especifican de manera relativa usando porcentajes.
paddingTop	Relleno parte superior.
paddingRight	Relleno a la derecha.
paddingLeft	Relleno a la izquierda.
paddingBottom	Relleno parte inferior.
below	Posiciona el elemento debajo de: Id del elemento.
toRightOf	Posiciona el elemento a la derecha de: Id del elemento.
toLeftOf	Posiciona el elemento a la izquierda de: Id del elemento.
above	Posiciona el elemento encima de: Id del elemento.

Tabla 7.5 – Propiedades de los elementos gráficos comunes.

A continuación se muestra un posible contenido del archivo .XML que representa una interfaz gráfica muy simple, la cual está formada por varios de los elementos descritos en las tablas anteriores (Fig.7.10).

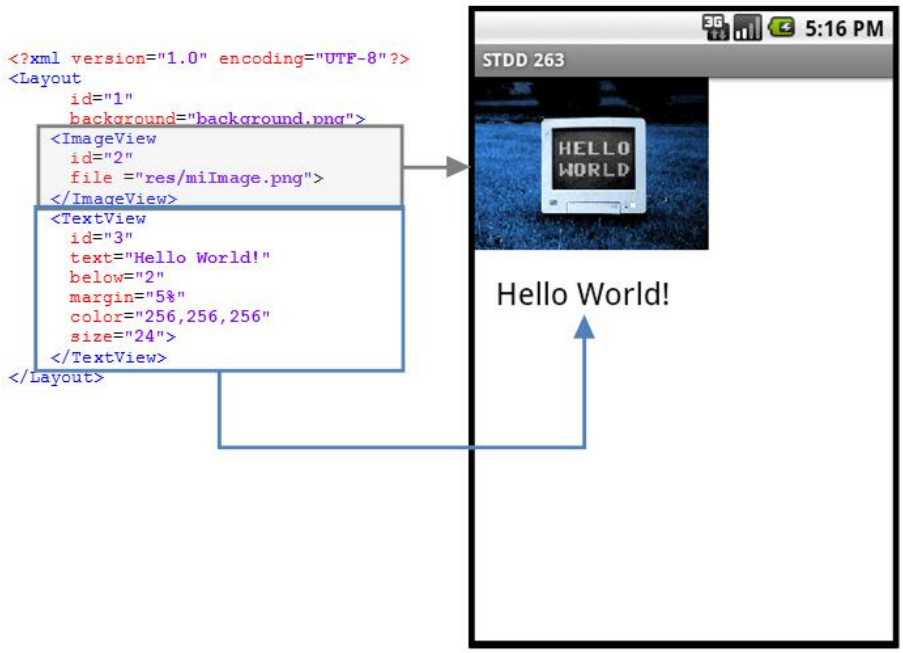

Figura 7.10 - Código de la interfaz gráfica y su interpretación en la pantalla del dispositivo.

7.4.4.1.2 Llamadas al código ejecutable desde la interfaz gráfica

Desde un elemento de la interfaz gráfica pueden realizarse llamadas al código ejecutable del objeto virtual.

Para indicar que no se quiere usar una cadena literalmente sino que se está invocando a una acción que requiere ejecución de código el valor del atributo debe comenzar por el carácter '#' y tener el formato: #[paquete.clase]/método(parámetro, parámetro).

Por ejemplo:

```
<TextView
    id="3"
    text="#com.entrada.entradaAvatar.Entrada/getSynopsis()">
</TextView>
```

En el caso de que se haya especificado un paquete y una clase principal en el fichero de configuración no es necesario especificar el paquete.clase en cada una de las llamadas, ya que por omisión el código ejecutable se buscará en la clase establecida por defecto.

El código asociado a ciertos atributos puede ejecutarse antes de que se produzca la carga de la interfaz gráfica, si el método tiene retorno éste se incluirá en el valor del atributo correspondiente (Fig.7.11).

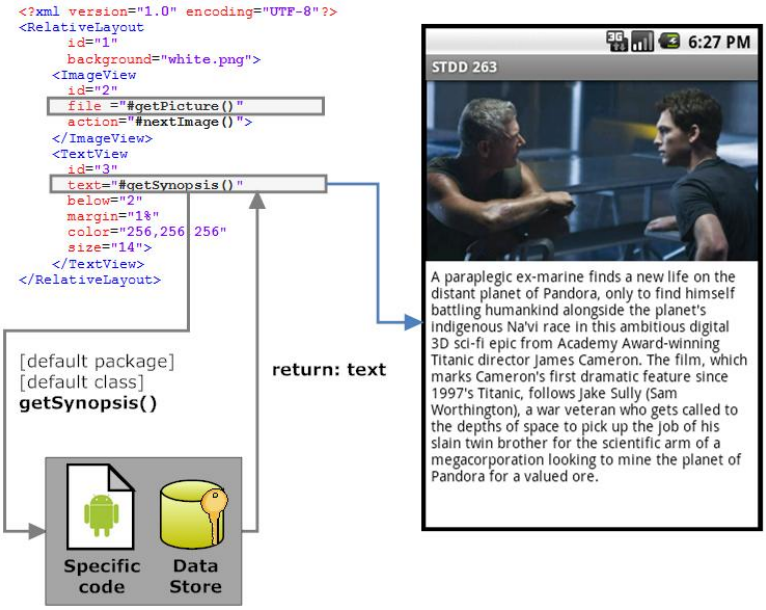

Figura 7.11 - Código de la interfaz gráfica que hace referencia al código ejecutable. A la derecha se muestra la interpretación en la pantalla del dispositivo

En el caso de la propiedad **action** el código correspondiente se ejecutara al pulsar sobre el elemento. En este ejemplo al pulsar sobre la imagen se ejecutará el método **nextImage()** (Fig.7.11).

En el caso de que el método al que se hace referencia reciba parámetros existen dos variantes.

1. Introducir los parámetros estáticamente en la llamada. De la forma, tipo: valor.

```
<TextView
    id="3"
    text="#com.entrada.entradaAvatar.Entrada/getText(int:4)">
</TextView>
```

2. Introducir parámetros relativos a otros elementos de la interfaz. Este sería el caso si se quisiera desarrollar por ejemplo un formulario de entrada. Para hacer referencia al texto de otro elemento se debe introducir como parámetro la expresión: ?+id, que tomará el valor del texto del elemento con la ID correspondiente en el caso de que éste sea un EditText o del valor true/false en caso de que sea un Checkbox o RadioButton.

```
<EditText
    id="2"
    text="Tu nombre aqui"
    width="20%"
    height="10%">
</EditText>
<Button
    id="3"
    below="2"
    width="20%"
    height="10%"
    action="#setNombre(String:?2)">
</Button>
```

Ventajas

- Al utilizar una descripción relativa para establecer las posiciones de los elementos en la interfaz ésta puede visualizarse de manera muy similar en dispositivos con distintas resoluciones y tamaños de pantalla.
- La estructura que sigue el XML es legible y resulta similar a un HTML.
- Permite construir interfaces ricas para los objetos virtuales STDD. Desde el punto de vista del cliente es como si se tratase de una aplicación instalada en su terminal.

7.4.4.2 WSDL

WSDL son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web.

WSDL describe la interfaz pública de los servicios Web. Está basado en XML y describe la forma de comunicación. Es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen de forma abstracta y se ligan después al protocolo concreto de red y al formato del mensaje.

El objeto virtual STDD puede incluir este tipo de interfaz pública para que otras aplicaciones puedan interactuar con él accediendo directamente a sus acciones.

7.4.4.3 API STDD

Opcionalmente puede incluirse un archivo XML que describa las acciones asociadas al objeto virtual STDD con el propósito de que éstas puedan ser directamente ejecutadas por otros objetos STDD que se encuentren cargados en el dispositivo.

El API STDD permite establecer de manera sencilla procesos de comunicación entre objetos virtuales alojados en un mismo dispositivo. Ofrece una interfaz para que los objetos cargados en un dispositivo puedan realizar directamente llamadas a la lógica de negocio de otros objetos.

La manera de describir las acciones públicas a las cuales otros objetos pueden acceder será mediante un descriptor XML con el siguiente formato:

```
<!-- Declaración de datos complejos en caso de que existan-->
<complexData name="Objeto 1" android="paquete.clase" />
<!--  Declaración de Operaciones -->
<operation name="Metodo1" android="paquete.clase.metodo">
    <description>Descripción método Opcional</description>
    <!—Parámetros de entrada -->
    <input>
        <param name="param1" type="string"/>
    </input>
    <!-- Retorno -->
    <output>
        <param name="return" type="string"/>
    </output>
</operation>
```

7.4.5 Código ejecutable

La lógica de negocio del objeto virtual STDD está encapsulada en el fichero de código, el cual puede hacer uso de un fichero de persistencia si fuese necesario.

Dentro de un mismo objeto virtual STDD se ofrece la posibilidad de incluir varios ficheros de código, cada uno en un lenguaje de programación para que el código pueda ser ejecutado por dispositivos que tengan distintos sistemas operativos. El propio Gestor STDD será el encargado de seleccionar el código fuente que sea adecuado para el dispositivo en cuestión.

7.4.5.1 Fichero de código

El fichero de código puede estar formado por un conjunto de paquetes, clases y librerías como si se tratara de un programa ordinario en el lenguaje de programación adecuado. Como consideraciones adicionales deben de tenerse en cuenta los siguientes puntos:

- Aquellas clases que vayan a ser accedidas desde las interfaces deben heredar de la clase base STDD. Esta clase ofrece un catálogo de métodos que permiten: modificar o cambiar la interfaz gráfica actual, controlar la persistencia de atributos y acceder a los elementos hardware del dispositivo que permiten capturar información de contexto y realizar acciones de comunicación.

- Aquellos métodos que vayan a ser accedidos desde cualquiera de las interfaces deben ser declarados como públicos.

- Los atributos que necesiten ser persistentes deben declararse de una manera particular, utilizando los métodos de soporte a la persistencia que ofrece la clase STDD.

7.4.5.1.1 Clase Base STDD

Como primer paso para comenzar la implementación de la lógica de negocio se debe importar la librería específica STDDlib, la cual contiene la clase base STDD. Deberán heredar de esta clase base:

- Las clases que vayan a ser directamente referenciadas desde las interfaces de usuario.

- Las clases que necesiten gestionar los elementos hardware del dispositivo.

El siguiente ejemplo muestra una codificación java básica (Fig.7.12):

```
<?xml version="1.0" encoding="UTF-8"?>
<Layout id="1">
    <TextView
        id="2"
        text="#package.DatosPersonales/getNombre()"
        size="20">
    </TextView >
</Layout>
```

```
public class DatosPersonales extends STDD{

    private String nombre="Hola mundo";

    // Obligatorio constructor publico sin parametros
    public DatosPersonales(){
        super();
    }

    public String getNombre(){
        return nombre;
    }
}
```

Figura 7.12 - La interfaz gráfica hace referencia a un método público contenido en el código ejecutable.

7.4.5.1.2 Persistencia

Como resulta habitual, los atributos de una clase se inicializan cada vez que se crea una nueva instancia de ésta, este proceso se repite cada vez que se carga un objeto virtual STDD. Si alguno de los datos usados es tratado como persistente mantendrá su valor sin importar las veces que el objeto virtual STDD se cargue, se duplique o se transfiera.

Para comenzar a utilizar la persistencia hay que hacer una llamada en el constructor al constructor de la clase base enviándole como parámetro una cadena de texto que actuará como token de seguridad. Esta cadena será autogenerada por una aplicación y se procesará cuando se realicen lecturas o escrituras en el almacén de datos, con el fin de comprobar que el código del servicio que está intentando acceder al almacén de datos es quien debe ser y que el almacén al que se accede es el que ha sido asociado inicialmente a el código y no otro.

El almacén de datos guarda pares: clave, valor, el valor almacenado puede ser cualquier tipo de dato u objeto, incluso propio, siempre y cuando éste sea serializable, para acceder a los elementos almacenados es necesario apoyarse en los métodos que implementa la clase STDD.

A continuación se listan algunos de los métodos utilizados para dar soporte a la persistencia que ofrece la clase STDD en la plataforma Java & Java-Android:

Modificar valor	Obtener valor
saveDataString(key,String)	loadDataString()
saveDataInt(key,int)	loadDataInt()
saveDataFloat(key,float)	loadDataFloat()
saveDataChar(key,char)	loadDataChar()
saveDataArrayString(key,String[])	loadDataString()
saveDataArrayInt(key,int[])	loadDataArrayInt()
saveDataArrayFloat(key,float[])	loadDataArrayFloat()
saveDataArrayChar(key,char[])	loadDataArrayChar()
saveDataObject(key,Object)	loadDataObject()
saveDataList(key,List<T>)	loadDataList()
saveData HasMap (key, HasMap<K,V>)	saveDataList(key,List)

Tabla 7.6 – Métodos de la clase base STDD para la gestión de la persistencia.

En el siguiente ejemplo se crea una clase STDD, (es decir que será directamente accedida desde una interfaz) la cual tiene un atributo **nombre** de tipo String que será tratado como persistente.

```
public class DatosPersonales extends STDD{

    private long serialVersionUID = 3406473650726246331L;

    // Obligatorio constructor público sin parámetros
    public DatosPersonales(){
        /** Constructor con el token de seguridad
            para sincronizarse con el almacén
            datos **/
        super("322AF872C789812342SJE");
    }

    public void setNombre(String nombreUsuario){
        /** Obtener el valor del atributo persistente
            "nombre" de tipo String */
        saveDataString("nombre",nombreUsuario);
    }

    public String getNombre(){
        /** Obtener el valor del atributo persistente
            "nombre" de tipo String */
        return loadDataString("nombre");
    }
}
```

7.4.5.1.3 Manejo de la interfaz de usuario

Como en cualquier aplicación tradicional es posible que durante la interacción con un objeto virtual STDD se deseen hacer transiciones entre distintas pantallas o modificar alguno de los componentes presentas en la interfaz de usuario. Cada vez que se realiza una llamada a algún método de una clase desde la interfaz gráfica o se ejecuta una acción el Gestor STDD pregunta a al código si se ha producido algún cambio en la interfaz gráfica.

Para realizar estos cambios es necesario utilizar los siguientes métodos que se encuentran implementados en la clase base STDD.

Método	Descripción
setPrivateContentView(String)	Cambiar la pantalla actual para la ejecución local del servicio, recibe como parámetro la ruta del fichero XML con la nueva interfaz gráfica.
setPublicContentView(String)	Cambiar la pantalla actual para la ejecución remota del servicio, recibe como parámetro la ruta del fichero XML con la nueva interfaz gráfica.
getPrivateContentView()	Retorna el nombre del fichero XML que está cargado actualmente como vista en la ejecución local del STDD.
getPublicContentView()	Retorna el nombre del fichero XML que está cargado actualmente como vista en la ejecución remota del STDD.
editViewElement(id, propiedad, valor)	Edita una propiedad de un elemento. Recibe como parámetros, la id del elemento, el nombre de la propiedad y el nuevo valor que se le quiere asignar.

Tabla 7.7 - Métodos de la clase base STDD para la gestión de la interfaz gráfica.

En el siguiente ejemplo se crea una clase STDD, la cual hace varias modificaciones a la interfaz gráfica. Desde el método **DesaparecerNombre()** modifica la propiedad visible del elemento con id = 3 estableciéndole el valor false.

El método **Siguiente()** cambia la interfaz actual por la descrita en el fichero privado2.xml si el objeto virtual STDD se está ejecutando de manera local y por public2.xml si lo está haciendo de manera remota.

```
public class DatosPersonales extends STDD{

    private long serialVersionUID = 3406473650726246331L;

    // Obligatorio constructor público sin parámetros
    public DatosPersonales(){
        super();
    }

    public void DesaparecerNombre(){
        /** Edita el elemento de la interfaz grafica
            que posee la id="3" cambiando su propiedad
            visible a false.
        */
        editViewElement("3","visible",false);
    }

    public void Siguiente(){
        /** Si el método do se invoca el cliente local
            se cambia la interfaz gráfica a
            la contenida en el fichero privado2.xml.
        **/
        setPrivateContentView("privado2.xml");
        /** Si el método se invoca un cliente remoto
            se cambia la interfaz gráfica a
            la contenida en el fichero privado2.xml.
        **/
        setPublicContentView("public2.xml");
    }
}
```

7.4.5.1.4 Gestión de elementos hardware

La clase base STDD contiene varios métodos que pueden ser utilizados para gestionar el manejo de los elementos hardware del dispositivo cliente. A través de llamadas a los métodos de la clase base STDD se realizan invocaciones directas al API de la plataforma (Fig.7.13), permitiendo de esta forma crear una capa de abstracción que permite el manejo de los elementos hardware del dispositivo cliente.

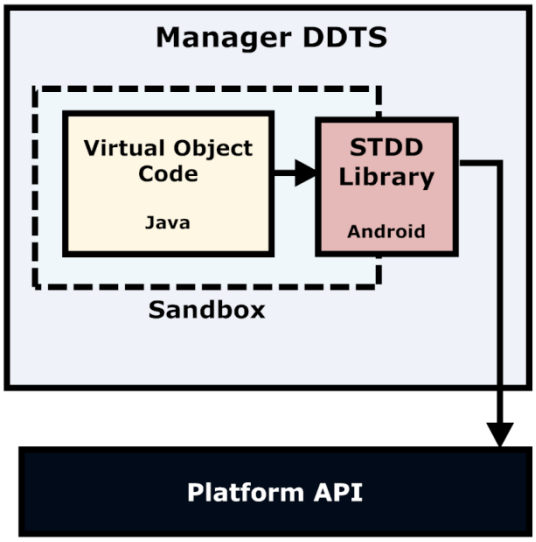

Figura 7.13 – El código del objeto virtual utiliza los métodos de la clase base STDD para gestionar los elementos hardware del dispositivo.

A continuación se listan algunos de los métodos de la clase STDD que habilitan la gestión de los elementos hardware del dispositivo cliente:

Método	Descripción
getMagneticFieldSensorValue()	Devuelve el valor actual registrado por el sensor de campo magnético. **Retorno**: Cadena de texto que contiene el valor del campo magnético en los ejes X, Y y Z expresado en µT microteslas , los valores se separan utilizando el carácter ",".
getProximitySensorValue()	Devuelve el valor actual registrado por el sensor de proximidad. Retorno: Valor numérico expresado en cm.
getOrientationSensorValue()	Devuelve el valor actual registrado por el sensor de orientación. **Retorno**: Cadena de texto que contiene el valor la orientación del dispositivo en los ejes X, Y y Z expresado en grados, los valores se separan utilizando el carácter ",".
getAccelerometerSensorValue()	Devuelve el valor actual de la aceleración del dispositivo. **Retorno**: Cadena de texto que contiene el valor la aceleración del dispositivo en los ejes X, Y y Z. Las mediciones en cada uno de los ejes pueden tomar valores positivos o negativos dependiendo de la orientación de la aceleración, los valores se separan utilizando el carácter ",".

getGravitySensorValue()	Devuelve el valor actual registrado por el sensor de gravedad. **Retorno**: número decimal, el valor de retorno esta expresado en m/s^2.
getLightSensorValue()	Devuelve el valor actual registrado por el sensor de luz. Retorno: número decimal comprendido entre 0 y 1. 0 corresponde al nivel de luz más bajo y 1 al más alto.
getTemperatureSensorValue()	Devuelve el valor actual registrado por el sensor de temperatura. **Retorno**: Número entero, el valor de retorno esta expresado grados centígrados.
getCameraPicture()	Devuelve una fotografía tomada por la cámara principal del dispositivo. **Retorno**: Fichero de tipo imagen, preferiblemente estará en formato .jpg u .png aunque puede variar dependiendo de las características técnicas del dispositivo.
getLocation()	Devuelve la localización actual del dispositivo. **Retorno**: Cadena de texto con las coordenadas geográficas longitud y latitud, separadas por el carácter “,”.

Tabla 7.8 - Métodos de la clase base STDD para la gestión de los elementos hardware del dispositivo.

Método	Descripción
btScan();	Escanea la red en busca de dispositivos bluetooth. **Retorno**: lista de dispositivos bluetooth , nombre del dispositivo + dirección
btPair (DeviceName / Adress)	Fuerza el emparejamiento con un dispositivo bluetooth. **Parámetros de entrada:** (1) Nombre del dispositivo o dirección del dispositivo (estándar IEEE 802 MAC-48) **Retorno**: booleano emparejamiento realizado con éxito true / false.
btUnPair (DeviceName / Adress)	Fuerza el desemparejamiento de un dispositivo bluetooth emparejado. **Parámetros de entrada:** (1) Nombre del dispositivo o dirección del dispositivo (estándar IEEE 802 MAC-48) **Retorno**: booleano desemparejamiento realizado con éxito true / false.
btConnect(DeviceName / Adress , UUID)	Establece una conexión con un dispositivo bluetooth previamente emparejado. **Parámetros de entrada:** (1) Nombre del dispositivo o dirección del dispositivo (estándar IEEE 802 MAC-48) (2) UUID Identificador universal del servicio Bluetooth que se ejecuta en el dispositivo destino. **Retorno**: booleano conexión realizada con éxito true / false.
btDisConnect(DeviceName / Adress)	Corta la conexión con un dispositivo Bluetooth conectado. **Parámetros de entrada:** (1) Nombre del dispositivo o dirección del dispositivo (estándar IEEE 802 MAC-48) **Retorno**: booleano desconexión realizada con éxito true / false.
btWrite(DeviceName / Adress, Bytes)	Envía una cadena de datos (bytes) a un dispositivo Bluetooth conectado. **Parámetros de entrada:** (1) Nombre del dispositivo o dirección del dispositivo (estándar IEEE 802 MAC-48) (2) Bytes **Retorno**: booleano si la escritura realizada con éxito true / false.
btListen(DeviceName / Adress)	Escucha hasta recibir una cadena de datos (bytes) procedente de un dispositivo Bluetooth conectado. **Parámetros de entrada:** (1) Nombre del dispositivo o dirección del dispositivo (estándar IEEE 802 MAC-48) **Retorno**: Cadena de bytes recibida.

Tabla 7.9 - Métodos de la clase base STDD para la gestión de los elementos hardware del dispositivo.

7.4.6 Almacén de datos

El almacén de datos es un fichero que puede incluirse de forma optativa en la estructura del objeto virtual STDD, resulta necesario incluirlo si se pretende tratar como persistente algún dato.

Para añadir un almacén de datos es necesario declararlo en el fichero XML de configuración estableciendo su ruta, o rutas en el caso de que fuese necesario incluir más de uno, por los diferentes lenguajes de programación.

El fichero contiene pares de elementos clave: valor. Las claves identifican a cada valor, por lo que deben de ser únicas, el valor puede ser cualquier tipo de objeto que sea serializable.

La aplicación incluida en el kit de desarrollo utilizada para crear el almacén de datos generara un fichero .obj y una clave de seguridad que debe introducirse en el constructor de la clase STDD. Esta clave formará pareja con otra contenida en el almacén de datos y se utilizará para realizar comprobaciones de seguridad.

7.4.7 Proceso de ejecución local

La ejecución local se produce cuando un usuario ejecuta un objeto virtual STDD cargado en su propio dispositivo.

Para que un objeto virtual STDD pueda ser ejecutado de manera local es necesario que se cumplan las siguientes condiciones:

1. El objeto virtual STDD debe estar cargado.

2. El fichero descriptor del STDD debe haber sido definida una interfaz privada principal.

```
<interface private="privateInt.xml" />
```

3. Contener una versión de código ejecutable compatible con el dispositivo contenedor.

```
<code android="Ticket12A78CHJ12_android.apk"
    java="Ticket12A78CHJ12_j2me.jar"/>
```

7.4.8 Proceso de ejecución remota

La ejecución remota se produce cuando un usuario: **dispositivo cliente,** ejecuta un objeto virtual STDD que se encuentra cargado en un **dispositivo servidor**.

Para que un objeto virtual STDD pueda ser ejecutado remotamente desde otro terminal es necesario que se cumplan las siguientes condiciones:

Dispositivo Servidor:

1. El objeto virtual STDD debe estar cargado y encontrarse activo para su descubrimiento.

2. En el fichero descriptor del objeto virtual STDD cargado, debe haber sido definido una interfaz pública.

```
<interface public="publicInt.xml" />
```

3. Contener una versión de código ejecutable compatible con el dispositivo servidor.

```
<code android="Ticket12A78CHJ12_android.apk"
    java="Ticket12A78CHJ12_j2me.jar"/>
```

Dispositivo Cliente:

1. Encontrarse dentro del área de cobertura del servidor.

2. Descubrir el servicio.

7.4.9 Transferencia de objetos virtuales STDD

En algunos casos, dependiendo de la lógica de negocio puede resultar conveniente ofrecer la posibilidad de que los objetos virtuales STDD puedan ser transferidos entre dispositivos.

Para que un objeto virtual STDD pueda ser transferido a otro dispositivo ha de especificarse esta posibilidad en su fichero de configuración por medio de la propiedad **transferable,** la cual puede tomar el valor YES/NO:

Las posibilidades a la hora de realizar las transferencias dependen del valor de otras dos propiedades, además de la antes mencionada **transferable.** Si el objeto virtual STDD es considerado como único tampoco debería permitirse su copia.

```
<id unique="YES">12A78CHJ12</id>
<transferable>YES</transferable>
<copy>NO</copy>
```

En el ejemplo anterior, el intercambio de ficheros se realizaría por medio de una transacción atómica. El servidor perdería el objeto virtual STDD para cedérselo al cliente transfiriéndole todos los archivos que forman el objeto virtual STDD.

Otra posible forma de intercambio sería transfiriendo al cliente una copia del objeto virtual STDD. Para poder realizar este tipo de envío habría que especificar en el fichero de configuración que el objeto no es único y permite copiarse además de transferirse.

```
<id unique="NO">1</id>
<transferable>YES</transferable>
<copy>YES</copy>
```

Este último ejemplo permitiría realizar los dos tipos de transferencia, dando a elegir entre enviar los archivos originales del objeto virtual STDD o una copia de los mismos.

7.5 GESTOR STDD

El Gestor STDD, es la aplicación encargada de administrar los objetos virtuales en el dispositivo contenedor. Esta aplicación debe estar diseñada para poder ser instalada y ejecutada en el dispositivo en cuestión, teniendo en cuenta sus características, sistema operativo y lenguajes de programación que admite.

En este apartado se hará una descripción de las funciones del Gestor STDD, detallando su comportamiento y explicando los módulos principales que forman esta aplicación, no obstante la descripción que aquí se realiza no tiene por qué ser la única implementación posible. El Gestor STDD se podría optimizar y ampliar con nuevos módulos o funcionalidades, siempre y cuando maneje los objetos virtuales DDTS de forma estándar y siga poseyendo todas las responsabilidades que se le atribuyen en esta especificación.

7.5.1 Funciones principales

A continuación se realizará una descripción de las funciones principales que se le atribuyen al Gestor STDD.

7.5.1.1 Carga de objetos virtuales

El primer paso para comenzar a utilizar un objeto virtual es cargarlo en el Gestor STDD. Para ello los ficheros que forman la estructura del objeto deben encontrarse ubicados en el dispositivo cliente.

Cartera de objetos: el Gestor STDD dispone de una lista a la que denominaremos "cartera de objetos", una vez se cargan los ficheros correspondientes al objeto, este aparecerá en esa lista, que un objeto aparezca en esta lista significa que está listo para ser usado. El usuario puede administrar esta lista añadiendo nuevos objetos, eliminando o copiando los existentes.

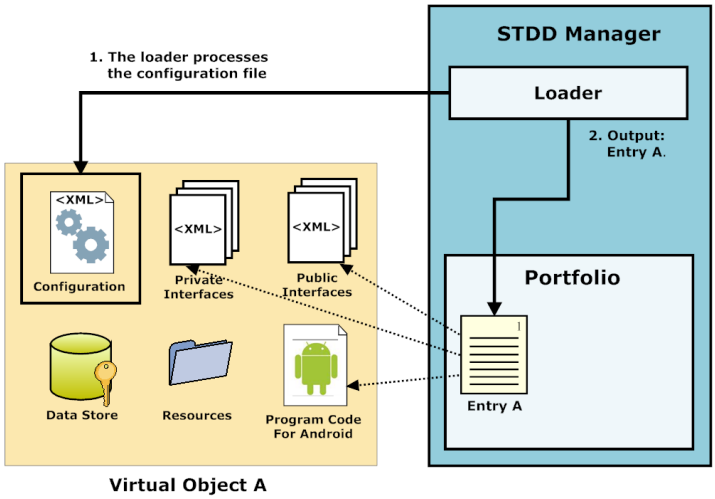

Figura 7.14 - Objeto virtual STDD cargado.

Para introducir un nuevo objeto en la cartera de objetos el Gestor STDD procesa el fichero XML de configuración, el cual contiene las propiedades del objeto virtual (Fig.7.14).

El cargador (Loader) es el módulo del Gestor STDD encargado de procesar los ficheros de configuración. Este módulo realiza varias comprobaciones sobre el archivo:

- Está en formato correcto.
- Contiene todas las propiedades de carácter obligatorio (nombre, id, ruta del código ejecutable y al menos una interfaz).
- Todas las propiedades tienen un valor que encaja dentro del rango esperado.
- Comprueba que el objeto no se encuentra cargado actualmente.

Si las comprobaciones son positivas el cargador crea una entrada, en la que almacena todas las propiedades del objeto (nombre, id, transferible, copiable, rutas a interfaces y ficheros de código, etc.), tanto las descritas en el fichero de configuración como las que toman un valor por defecto.

Una vez la entrada al objeto ha sido generada se introduce en la cartera de objetos (Fig.7.14). El usuario puede acceder a la cartera de objetos para ver qué objetos virtuales se encuentran cargados en su dispositivo, administrarlos y comenzar a interactuar con ellos.

7.5.1.2 Ejecución de código

La lógica de negocio de los objetos virtuales se encuentra encapsulada en los ficheros de código ejecutable. Durante la interacción con el objeto virtual las llamadas al código ejecutable se suceden.

Las ejecuciones de código comienzan una vez el objeto virtual está cargado en el dispositivo, el Gestor STDD es el encargado de seleccionar el fichero de código ejecutable adecuado en caso de que el objeto virtual contenga varios, si no se encuentra un código ejecutable compatible con el sistema operativo o características del dispositivo no resultara posible interpretar el objeto.

El módulo responsable de interactuar con el código ejecutable es el **lanzador** (launcher), su misión es invocar fragmentos de código o métodos y almacenar los retornos en caso de que existan.

El proceso seguido para la ejecución de código es el siguiente (Fig.7.15):

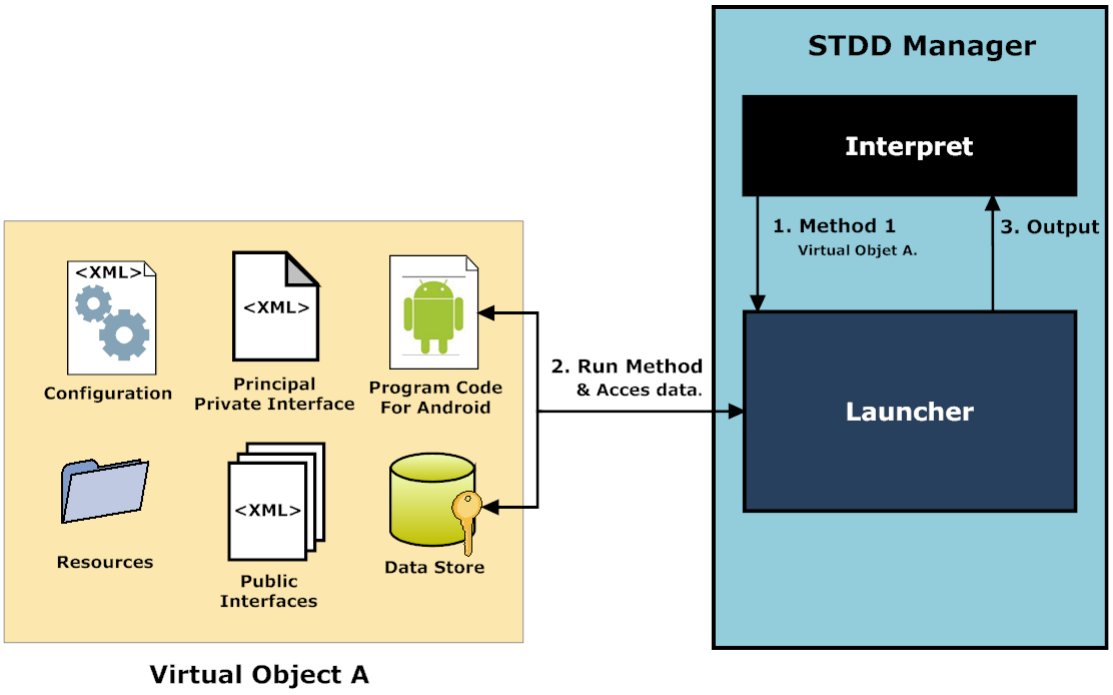

Figura 7.15 - El Lanzador se encarga de acceder al código fuente y al almacén de datos.

1. El **lanzador** recibe la petición de invocación al método perteneciente al objeto virtual.

2. Con la información que se ha extraído del fichero de configuración y se ha almacenado en la entrada correspondiente al objeto virtual, se conoce la ruta donde se encuentra el código ejecutable y almacén de datos. El **lanzador** carga el código ejecutable y lo sincroniza con el almacén de datos (si es que ha sido definido) y a continuación invoca el método indicado.

   Ejecución de código: el acceso a las clases y métodos del código ejecutable debe ~~de~~ realizarse mediante la carga dinámica de clases, concretamente esta aplicación Gestor STDD está construida para correr sobre la plataforma Android, por lo que utiliza el cargador de clases DEXClassLoader similar al ClassLoader de java.

   Por razones de seguridad el código de los objetos virtuales no debería de tener permisos para acceder al sistema de ficheros del dispositivo. La responsabilidad de la sincronización entre el código y el almacén de datos recae sobre el **lanzador**.

3. El **lanzador** notifica al **intérprete** que la ejecución de código ha finalizado y le envía el valor obtenido como consecuencia de la ejecución de código, en caso de que haya habido algún retorno.

7.5.1.3 Interpretación de objetos virtuales STDD

Los objetos virtuales que se encuentran cargados en el dispositivo pueden ser interpretados, entendiendo por interpretar leer la estructura del objeto y representarlo correctamente, de forma que el usuario pueda interactuar con él.

La interpretación de los objetos comienza cuando el Gestor STDD procesa las interfaces que el objeto virtual contiene y muestra por la pantalla del dispositivo de una forma adecuada los elementos gráficos que hay declarados en ellas. Los elementos de las interfaces pueden estar ligados a acciones que el objeto virtual contiene en su código, por ello la interpretación puede implicar también que se realicen llamadas al código ejecutable.

El proceso seguido para la interpretación es el siguiente (Fig.7.16):

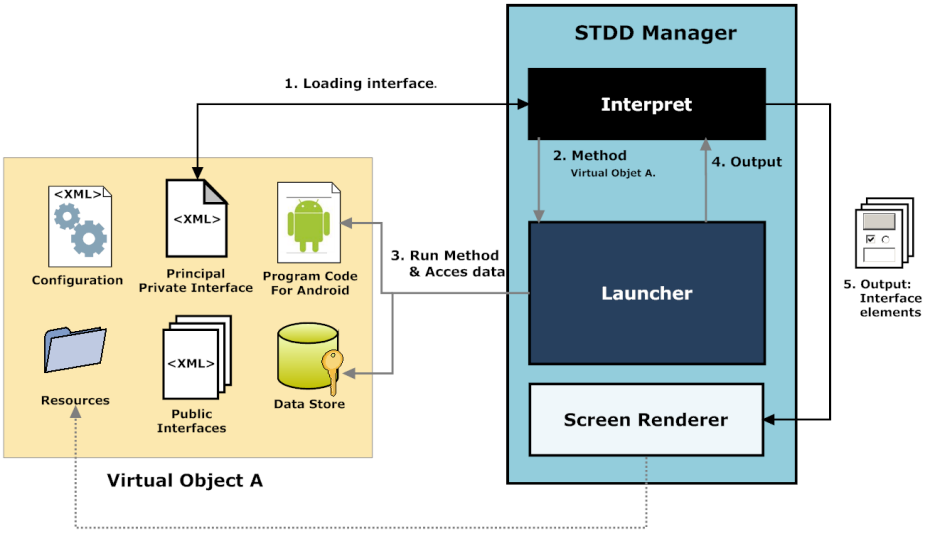

Figura 7.16 - Esquema del proceso: Interpretación de objetos virtuales STDD

1. Al comenzar a interpretar un objeto virtual cargado, el **intérprete** carga el fichero correspondiente a su interfaz privada principal. El nombre de la interfaz principal ha sido declarado en el fichero de configuración, por lo que esta información se encuentra almacenada en la entrada correspondiente al objeto. Una vez cargado, se comienza a procesar el fichero XML el cual describe los controles que debe incluir la interfaz gráfica.

2. Como ya se explicó en la especificación de las interfaces gráficas, estas pueden realizar llamadas al código ejecutable. Por ejemplo el valor de un campo de texto podría ser el retorno de la ejecución de un método. Si esto ocurre el **intérprete** enviará la petición al **lanzador**.

3. El **lanzador** carga el código ejecutable, lo sincroniza con el almacén de datos (si es que el objeto virtual lo posee) e invoca el método correspondiente.

4. El **lanzador** notifica al **intérprete** que la ejecución de código ha finalizado y le envía el valor de retorno obtenido, en caso de que haya habido.

5. Los pasos anteriores (3-4) se repiten tantas veces como llamadas haya al código ejecutable dentro de la interfaz gráfica. Una vez se han ejecutado los fragmentos de código, el **intérprete** puede finalizar el procesamiento de la interfaz gráfica, como resultado genera un "elemento de interfaz" por cada elemento descrito en el XML. Los elementos de interfaz son descripciones de controles: botones, listas, imágenes, etc.

Cada elemento de interfaz encapsula la información relativa a la representación de un elemento en la pantalla y a los comportamientos que se deben adoptar frente a ciertos eventos, por ejemplo: si debe ejecutar alguna acción cuando alguien presione el elemento, etc.

Cuando los elementos de interfaz son representados en la pantalla del dispositivo es posible que utilicen recursos incluidos en el objeto virtual, por ejemplo como imágenes u otro material multimedia.

7.5.1.4 Almacenamiento de objetos virtuales STDD

El Gestor STDD actúa como un contendor de objetos virtuales, sería lógico que a menudo un usuario poseyese varios objetos virtuales en un mismo momento, por lo que deben ofrecerse mecanismos para que sean almacenados.

Los objetos virtuales deben tener carácter persistente, es decir, una vez cargados tienen que estar disponibles. Cuando se carga un objeto virtual el Gestor STDD genera su entrada correspondiente, la cual debe guardarse en un almacén de datos persistente. Cada vez que el Gestor STDD comienza una nueva ejecución accede al almacén de datos para recuperar las entradas de los objetos virtuales que deben estar cargados en el dispositivo.

7.5.1.5 Gestión de objetos virtuales STDD

Se pueden realizar ciertas operaciones contra los objetos virtuales cargados, todos ellos pueden ser eliminados y de manera opcional pueden ser transferidos, copiados o modificados. Si un objeto virtual posee una de estas tres últimas propiedades lo indicara de manera específica en su fichero de configuración. Esta información también se almacena en la entrada correspondiente al objeto, ya que en ella se han copiado todos los valores de su configuración.

- **Eliminar:** Las entradas de la cartera de objetos pueden ser eliminadas. Existen dos tipos de eliminación. La primera solo elimina la entrada correspondiente al objeto virtual del Gestor STDD, por lo que podría volver a ser cargado a partir de sus ficheros. La segunda es una eliminación total que borra todos los archivos y ficheros correspondientes al objeto virtual.
    - **Expiración:** algunos objetos asignan una fecha a esta propiedad, esto significa que al llegar a la fecha señalada el Gestor STDD realiza de forma automática un borrado total sobre el objeto en cuestión.
- **Transferir:** la transferencia de un objeto a otro dispositivo consiste en que el objeto deje de estar disponible en el emisor y pase a estarlo en el receptor, conservando el mismo estado y propiedades que tenía. Al realizar la transferencia la entrada del objeto se elimina del dispositivo emisor y este transfiere de forma atómica los archivos que componen el objeto virtual. El receptor recibe los archivos del objeto virtual y lo carga en su Gestor STDD.

- **Copiar:** los objetos virtuales no únicos pueden ser copiados, la copia consiste en replicar los archivos que forman el objeto y posteriormente cargar en la cartera la copia realizada.

- **Modificar:** cuando un objeto no es modificable el Gestor STDD no debe permitir la carga ni ejecución del mismo si detecta que se ha modificado alguno de los archivos que lo forman (interfaces, código, almacén de datos).

7.5.1.6 Servidor de objetos virtuales STDD

Si en el fichero de configuración del objeto virtual se ha especificado permitir la ejecución de forma remota, el Gestor STDD debe ofrecer mecanismos para que los objetos virtuales cargados en el dispositivo puedan ser consumidos remotamente por otros usuarios, normalmente esta acción se realizará vía Bluetooth o Wi-Fi.

Los objetos virtuales cargados pueden encontrarse en estado activo o inactivo, que un objeto este activo significa que puede ser descubierto y ejecutado de forma remota por otro dispositivo. No todos los objetos pueden ser invocados de forma remota sino que debe ser especificado en su fichero de configuración. Además el objeto debe incluir una interfaz gráfica pública principal que utilizarán para este tipo de ejecución.

7.5.1.7 Descubrimiento de objetos virtuales STDD

Para poder ejecutar remotamente un objeto virtual el primer paso consiste en descubrir el dispositivo servidor; una vez establecido el contacto con el dispositivo se solicita a su Gestor STDD una lista con los objetos virtuales que se encuentran cargados y pueden ser ejecutados de forma remota.

El proceso seguido en el descubrimiento de objetos virtuales es el siguiente (Fig.7.17):

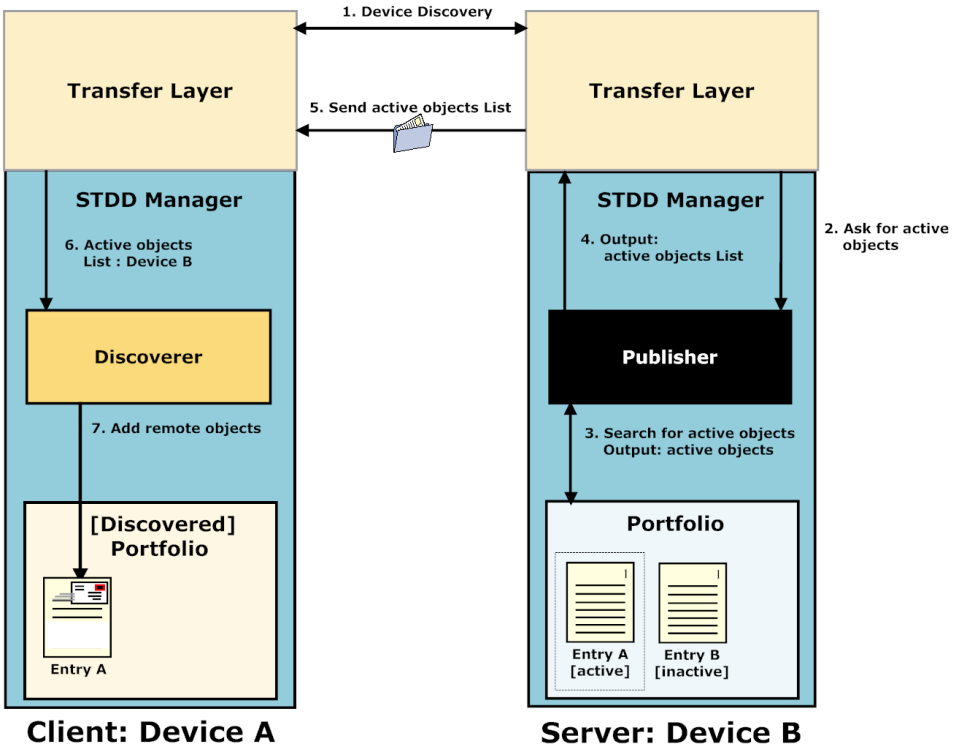

Figura 7.17 - Esquema del proceso: Descubrimiento de objetos virtuales STDD.

1. El cliente busca dispositivos en su rango de cobertura. En caso de encontrarlos les pregunta si disponen de objetos virtuales válidos para realizar una ejecución remota.

2. El dispositivo descubierto, envía la petición al **publicador** (Publisher) solicitando información sobre los objetos virtuales activos que posee cargados en ese momento.

3. El **publicador** realiza una búsqueda sobre la cartera de objetos, seleccionando aquellos que están activos y poseen una interfaz pública, es decir, aquellos que permiten ejecutarse de forma remota. El resultado es una lista con las entradas correspondientes a los objetos publicados.

4. Partiendo de la lista de entradas se generan unos nuevos tipos de entradas con una información más limitada, las cuales contienen el nombre, identificador y el icono del objeto virtual, así como los datos del dispositivo servidor.

5. El servidor envía al dispositivo cliente la lista con las entradas correspondientes a los objetos virtuales publicados, además puede enviar otro tipo de recursos asociados, como las imágenes de los iconos pertenecientes a los objetos.

6. El **descubridor** (discoverer) del cliente procesa la lista recibida y añade las entradas a la cartera de objetos descubiertos.

7. Una vez añadidas las entradas, el usuario cliente puede visualizar los objetos virtuales descubiertos y seleccionar uno de ellos para comenzar la ejecución remota.

7.5.1.8 Ejecución remota

La ejecución remota comienza cuando el cliente selecciona uno de los objetos virtuales que ha descubierto; al hacer la selección envía una petición al dispositivo servidor, el cual le envía la interfaz gráfica pública principal correspondiente al objeto y los recursos asociados a ella en caso de que los tenga. Una vez el cliente dispone de la interfaz pública, la interpreta como si se tratara de una ejecución local.

El proceso seguido en la ejecución remota de objetos virtuales es el siguiente (Fig.7.18):

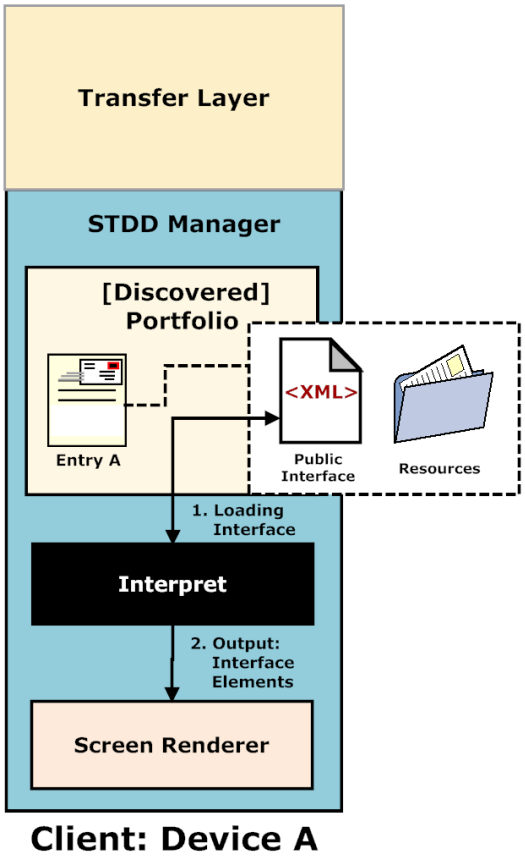

Figura 7.18 - Esquema del proceso: Ejecución remota 1 de objetos virtuales STDD.

1. Para comenzar la ejecución remota del objeto virtual el **Intérprete** carga el fichero correspondiente a la interfaz pública principal. Una vez cargado, comienza a procesar el fichero XML, el cual define que controles deben representarse en la pantalla.

   Como se explico en la especificación de las interfaces gráficas, en ocasiones, estas pueden hacer referencia al código ejecutable. Por ejemplo el valor de un campo de texto podría ser el retorno de la ejecución de un método. Si esto ocurre el **intérprete** enviará una petición al dispositivo servidor, esta situación se detallara más adelante.

2. Cuando el **intérprete** procesa la interfaz gráfica publica, genera como resultado un "elemento de interfaz" por cada elemento descrito en el XML. Los elementos de interfaz son descripciones de controles: botones, listas, imágenes, etc. Cada elemento de interfaz encapsula la información sobre la representación en la pantalla del elemento y el comportamiento que debe adoptar este frente a ciertos eventos, por ejemplo: si debe realizar alguna llamada a un código cuando alguien lo presione, etc.

Ciertas acciones como la carga de interfaces gráficas o la interacción con elementos de la interfaz (tocar un botón), pueden requerir llamadas a la lógica de negocio del objeto. En el caso de la ejecución remota, la lógica se encuentra alojada en el dispositivo servidor. El proceso seguido para llevar a cabo las ejecuciones de código remoto es el siguiente (Fig.7.19):

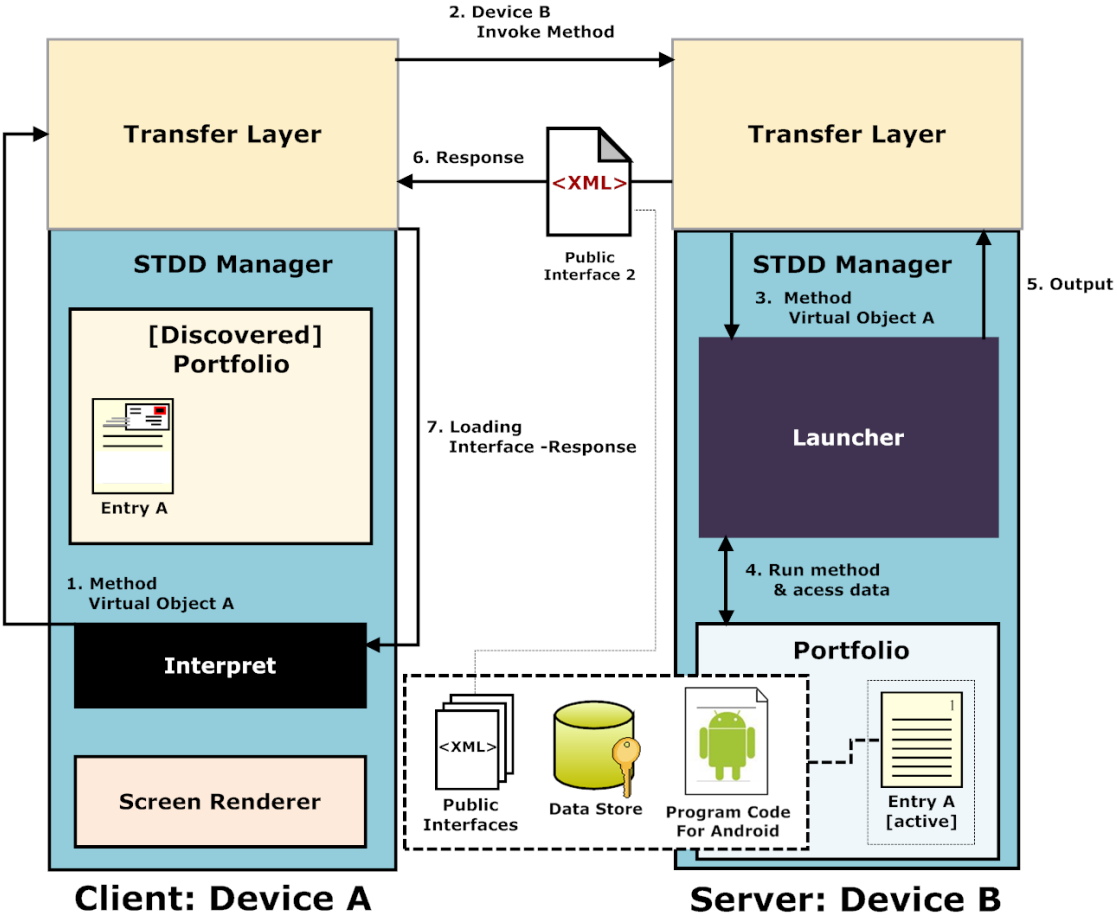

Figura 7.19 - Esquema del proceso: Ejecución remota 2 de objetos virtuales STDD.

1. La ejecución de código puede ser requerida por la representación de una interfaz gráfica, por ejemplo: el valor de un campo de texto podría ser el retorno de la ejecución de un método. También podría producirse una ejecución de código como consecuencia de un evento que se ha disparado, por ejemplo al pulsar un botón. Cuando el **intérprete** requiere una ejecución de código para completar la interpretación del objeto virtual remoto, construye una petición con la información del método que necesita ejecutar y la envía al dispositivo dueño del objeto (servidor).

2. La petición que se envía desde el cliente al servidor, contiene la cabecera del método, valores que toman sus entradas, si es que las tiene, e información sobre la identidad. Es necesario identificar el objeto virtual ya que un dispositivo puede servir a varios objetos simultáneamente.

3. Cuando el servidor recibe la petición envía al **lanzador** la información para comenzar la ejecución.

4. De igual manera que cuando se ejecuta un objeto de forma local el **lanzador** carga el código ejecutable, lo sincroniza con el almacén de datos (si es que el objeto virtual lo posee) e invoca el método correspondiente.

La ejecución de código puede devolver o no algún valor y también implicar un cambio de interfaz gráfica, por lo que se almacenan los valores de retorno y se comprueba si se han efectuado cambios de interfaz gráfica pública, en el caso de que se haya realizado algún cambio, la nueva interfaz gráfica será enviada al dispositivo cliente así como los recursos asociados a ella.

5. El **lanzador** selecciona la información que ha de ser enviada como respuesta al dispositivo cliente. La respuesta estará formada por datos, interfaces y recursos, dependiendo de la acción que se halla ejecutado.

6. El servidor envía la respuesta al dispositivo cliente.

7. Una vez el cliente recibe la respuesta, esta es procesada por el **intérprete**. Si la respuesta contiene una nueva interfaz, el **intérprete** la carga y procesa para que el usuario la visualice. Si la información requerida era un valor correspondiente a la ejecución de un método el cual se necesitaba para la interpretación de una interfaz, inserta el valor recibido dentro del elemento gráfico correspondiente.

7.6 SOLUCIÓN ADOPTADA

7.6.1 Introducción

El objetivo de este capítulo es presentar la aplicación práctica de la estructura propuesta "objeto virtual STDD". Para ilustrar el caso práctico se ha seleccionado una lógica de negocio concreta: la venta de entradas de cine vía Internet; la estructura propuesta será aplicada en la construcción del objeto virtual entrada de cine. En este capítulo se detallaran los pasos seguidos a lo largo del proceso de desarrollo y posterior uso del objeto.

El prototipo del objeto virtual entrada de cine será ejecutado sobre un dispositivo móvil Android que cuenta con una implementación básica del Gestor STDD, aplicación necesaria para la interpretación de los objetos virtuales STDD.

7.6.2 Desarrollo de un objeto virtual STDD

7.6.2.1 Análisis del objeto

El primer paso consiste en obtener todos los detalles del objeto que debemos modelar, este análisis se divide en tres pasos.

- **Acciones,** que usuarios, aplicaciones u otros objetos pueden realizar y en las que el objeto que estamos diseñando interviene de algún modo. Las acciones vendrían a ser similares a los casos de uso de una aplicación convencional. Se diferencian dos tipos de acciones, aquellas que solo el usuario propietario del objeto las puede ejecutar y las públicas que son ejecutadas por otros usuarios externos o aplicaciones.

- **Datos y recursos asociados:** una vez conocidas las acciones, puede que algunas requieran incluir recursos como: fotos, videos, etc.

- **Propiedades del objeto:** este apartado va destinado a asignar un valor a ciertas propiedades generales asociadas a los objetos virtuales STDD. Estas propiedades resultan importantes a la hora de tratar el objeto y son: nombre, tipo, ¿es único?, ¿se puede duplicar?, ¿se puede editar? ¿Transferir a otro usuario?, ¿Tiene fecha de expiración?, etc.

7.6.2.1.1 Acciones

Si pensamos en un objeto convencional "entrada de cine", obtenemos dos casos de uso básicos:

- **Consultar información de entrada:** la entrada tiene asociada información que puede ser consultada: el número de sala, la hora, el número de la butaca, etc.

- **Canjear entrada:** entregar la entrada en la taquilla del cine, la rompen y el resultado es una entrada con la misma información pero que ya no podrá volver a ser canjeada. En este caso y siguiendo con la tendencia a la automatización de tareas, la acción de

canjear la entrada se realizará mediante la conexión ~~a~~ vía Bluetooth a un sistema embebido situado en la entrada a la sala de cine.

Las dos acciones básicas anteriores podrían por si solas definir un objeto entrada de cine, aun así se añadirán más acciones para conseguir un objeto más completo.

- **Consultar información de película**: otros usuarios, los cuales no son dueños de la entrada, podrían querer ver información sobre la película: sinopsis, tráiler fotos… También se podría incluir información sobre la web donde ha sido comprada la entrada o las salas de cine donde se proyecta la película.

Dentro de este apartado contemplamos que acciones serán realizadas por el usuario dueño de la entrada, y cuales por usuarios externos u otras aplicaciones.

Acción	Inicia la acción
Consultar información de entrada	Propietario
Canjear entrada	Propietario
Consultar información de película	Usuario externo

Tabla 7.10 – Acciones del objeto virtual.

7.6.2.1.2 Datos y recursos

Para dar soporte a las acciones definidas anteriormente es necesario contar con una serie de datos:

Consultar información de entrada

Dato	Tipo
Nombre del cine	Cadena de texto
Nº De sala	Numérico
Nº Butaca	Numérico
Sesión	Fecha + Hora
Usada	Booleano

Tabla 7.11 – Datos y recursos del objeto virtual.

Canjear entrada

Estos datos podrían variar dependiendo del método que se utilice para canjear/validar la entrada.

Dato	Tipo
Identificar de la entrada	Cadena de texto
Usada	Booleano

Tabla 7.12 – Datos y recursos de la acción canjear entrada.

Consultar información de película

Dato	Tipo
Sinopsis	Cadena de texto
Fotos	Imagen

Tabla 7.13 – Datos y recursos de la acción consultar información de película.

7.6.2.1.3 Comportamiento General

En este apartado se le asignara valor a una serie de propiedades asociadas al objeto virtual STDD. Estas propiedades resultan especialmente importantes a la hora de establecer la identidad y comportamiento del objeto.

Nombre	Obligatorio	Descripción
name	X	
type		www.entradasDeCineObjetosVirtuales.com . Este campo puede ser útil si se desea organizar los objetos por tipo o para aplicaciones que procesen objetos de un tipo determinado.
id	X	47236271 – Un identificador alfanumérico. En este caso será único, no podrá haber otra entrada con el mismo.
unique		YES: La entrada será un objeto único. Solo existe una entrada con estas características.
transferable		YES: Sera transferible a otros dispositivos
expiration		12-10-2009: Pasada esta fecha el objeto quedara invalidado. Una opción sería establecer como fecha de expiración el día siguiente a la proyección de la película.
edit		NO: Indica que no se pueden editar los ficheros o recursos que componen el objeto, si se detecta alguna modificación el objeto quedaría inutilizado.
copy		NO: El objeto entrada no podrá duplicarse, algo obvio si tenemos en cuenta que cada entrada es única.

Tabla 7.14 – Valores de los elementos de configuración del objeto virtual.

7.6.2 Proceso de desarrollo

7.6.2.1 Introducción

En este apartado se describirán los pasos seguidos para el desarrollo concreto del objeto virtual STDD entrada de cine.

7.6.2.2 Lógica de negocio y datos

El objeto virtual contará con un dato de carácter persistente de clave "usada", será de tipo booleano y tomara como valor inicial false. El almacén de datos se genera utilizando la aplicación de datos contenida en el kit de desarrollo. El resultado será: un archivo **info.obj** y una clave que debe introducirse en la llamada al constructor de la clase base STDD.

La lógica de negocio ira encapsulada dentro del fichero de código, en este caso la implementación se realizará únicamente en java ya que las pruebas con el prototipo se realizaran sobre dispositivos Android.

En este caso se crea un nuevo proyecto en el entorno de programación eclipse (java-android) e importamos la librería STDD.

En primer lugar declaramos todos los atributos no persistentes que tiene el objeto: entrada, ID, nombre, posición en la sala, horario y código identificador del cine. Además de estos atributos habrá que añadir el valor **"usada"** que tendrá carácter persistente; este valor indica si la entrada ya ha sido usada o no. ¿Por qué es persistente? El objeto virtual puede cambiar de dispositivo o ser eliminado y vuelto a cargar, este valor no puede reiniciarse tomando de nuevo su valor inicial (false), los cambios realizados sobre el valor "usada" deben de ser persistentes.

Los métodos más significativos son los siguientes:

- **Canjear entrada:** debe comprobar que la entrada no ha sido utilizada previamente, para ello tiene que acceder al valor del registro "usada". Si efectivamente la entrada no ha sido usada debe conectarse utilizando Bluetooth al sistema de la sala de cine y dar luz verde para entrar a la sala. Después de realizar esta acción con éxito el registro "usada" debe tomar el valor true.

- **GetPicture y SiguienteImagen:** como se había definido en el apartado de análisis, el objeto contendrá una galería de imágenes sobre la película. El array picures[] almacena los nombres de las imágenes mientras que la variable pPicture contiene la posición de la fotografía actual.

A continuación se muestra la implementación de la clase Java EntradaCine:

```
public class EntradaCine extends STDD{

// Atributos para ejecución local
    private String eID = "47236271";
    private String nombre="Robot Movie";
    private int posicionCine = 23;
    private String horario = "12/10/2009 18:00";
    private int codigoCine = 1234;

//  Atributos para ejecución remota
    private String synopsis = "A paraplegic ex-marine finds a ";
    // Recursos Fotos
    private String[] pictures= {"Sc1.png","Sc2.png","Sc3.png"};
    // Foto Actual.
    private int pPicture = 0;


    public EntradaCine(){
        /** Constructor con el token de seguridad
```

```
            para sincronizarse con el almacén
            datos **/
        super("322AF872C789812342SJE");
    }

    public boolean CanjearEntrada() throws Exception{
        // Cargamos el registro "usada" del almacén de datos
        if(!loadDataBoolean("usada")){
            // Emparejamiento
            boolean emparejado = btPair("iMac de admin");
            if( emparejado ){
                // Conexión
                if ( btConnect("iMac de admin",
                        "5ee01872-4c9e-48d6-9f61-015ff816b278"){
                    // Enviar ID de la entrada
                    btWrite("iMac de admin",eID.getBytes("UTF-16LE"));
                    // Leer ¿ID valida?
                    int ret = btListen("iMac de admin");
                    if (ret == 1){
                        // Grabar el valor true en el registro "usada"
                        saveDataBoolean("usada",true);
                        return true;
                    }
                }
            }

            return false;
        } else {
            throw new Exception("This ticket has already been used");
        }
    }

    public String getPicture(){
        return pictures[pPicture];
    }

    public void siguienteImagen(){
        if( pPicture < pictures.length-1)
            pPicture++;
        else
            pPicture = 0;
    }


    // GETTERS
    public String getNombre() {
        return nombre;
    }

    public int getPosicionCine() {
        return posicionCine;
    }

    public String getHorario() {
        return horario;
    }

    public int getCodigoCine() {
        return codigoCine;
    }
```

```
    public String getSynopsis() {
        return synopsis;
    }

}
```

7.6.2.3 Interfaces

7.6.2.3.1 Interfaces gráficas privadas

Las interfaces privadas se visualizan cuando el objeto virtual está siendo ejecutado por su propietario. En este caso declararemos una única interfaz gráfica privada desde la que el usuario puede ver toda la información relativa a su entrada e iniciar la acción de canjearla.

En el documento privateInterfaz.xml se describirán los elementos gráficos que se mostraran por pantalla y su comportamiento. Varios de estos elementos hacen referencia a los métodos que se encuentran en el código ejecutable. Los atributos text de varios TextView tomarán su valor de los métodos get. Se referenciará el método que debe invocarse cuando se pulse el botón de canjear la entrada. En este caso no será necesario definir el paquete ni la clase a la que pertenecen los métodos cuando se realicen referencias al código, ya que la única clase que tiene el objeto virtual será declarada como principal en el fichero de configuración.

Esta es la interfaz gráfica privada completa:

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout id="1" background="fondoentrada.png">
    <ImageView id="2" width="80" height ="80" file ="cartel.png"
        margin="20"></ImageView>
    <TextView id="3" text="Title: " toRightOf="2" marginTop="20"
        color="0,0,0" size="17"></TextView>
    <TextView id="4" text="Date: " toRightOf="2" bellow="3"
        color="0,0,0" size="17"></TextView>
    <TextView id="5" text="Position: " toRightOf="2" bellow="4"
        color="0,0,0" size="17"></TextView>
    <TextView id="6" text="Cinema: " toRightOf="2" bellow="5"
        color="0,0,0" size="17"></TextView>
    <TextView id="7" text="Used: " toRightOf="2" bellow="6"
        color="0,0,0" size="17"></TextView>
    <TextView id="8" text="#getNombre()" toRightOf="3" marginTop="20"
        color="0,0,0" size="17"></TextView>
    <TextView id="9" text="#getHorario()" toRightOf="4" bellow="3"
        color="0,0,0" size="17"></TextView>
    <TextView id="10" text="#getPosicionCine()" toRightOf="5"
        bellow="4" color="0,0,0" size="17"></TextView>
    <TextView id="11" text="#getCodigoCine()" toRightOf="6" bellow="5"
        color="0,0,0" size="17"></TextView>
    <TextView id="12" text="#isUsada()" toRightOf="7" bellow="6"
        color="0,0,0" size="17"></TextView>
    <Button id="13" width="280" height="100"
        background="#getEntradaImage()" bellow="2"
        action="#CanjearEntrada()" margin="20"></Button>
</RelativeLayout>
```

Para darle un mejor aspecto a la interfaz hemos añadido un método que devolverá la imagen del botón que se utiliza para canjear el ticket. El nuevo método devolverá la referencia a una imagen distinta si la entrada esta usada o si no lo está (Fig.7.20).

```
public String getEntradaImage() throws Exception{
        // Comprobar el valor del dato "usada"
        if(loadDataBoolean("usada"))
            return "ticket_cortado.png";
        else
            return "ticket_entero.png";
    }
```

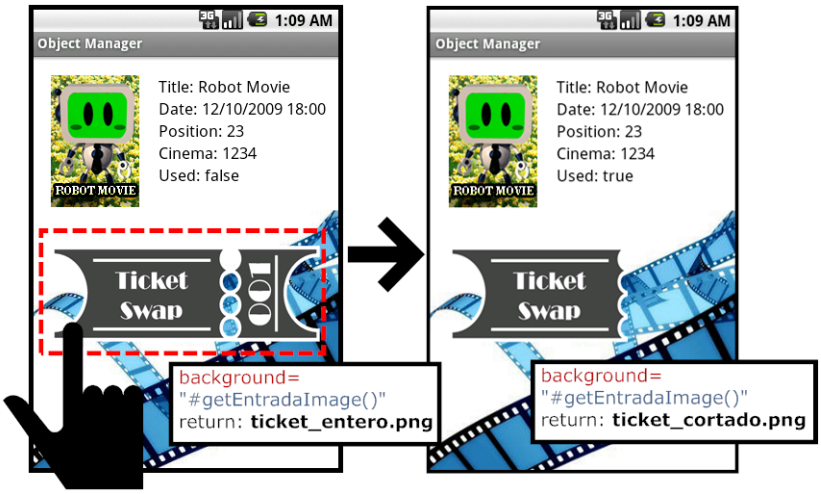

Figura 7.20 - Interfaz gráfica privada antes de usar el ticket y después.

7.6.2.3.2 Interfaces gráficas públicas

Las interfaces gráficas públicas se visualizan cuando el objeto está siendo ejecutado de forma remota por otro usuario. En este caso se declarará una única interfaz gráfica pública, desde la que los usuarios remotos podrán ver información relativa a la película (no a la entrada).

En el documento publicInterfaz.xml se describen los elementos gráficos que se mostrarán por pantalla y su comportamiento. Al igual que en la interfaz privada, dos de los elementos que se definirán hacen referencia a métodos que se encuentran en el código ejecutable.

Esta es la interfaz gráfica pública completa:

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout id="1" background="white.png">
    <ImageView id="2" file ="#getPicture()"
         action="#nextImage()"></ImageView>
    <TextView id="3" text="#getSynopsis()"
   bellow="2" margin="5" color="256,256,256"
   size="14"></TextView>
</RelativeLayout>
```

En este caso la propia imagen actuará como botón para pasar a la siguiente fotografía (Fig.7.21).

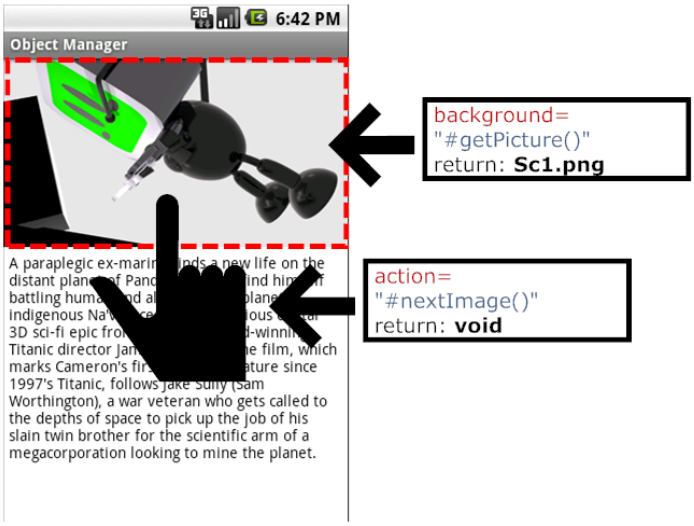

Figura 7.21 - Interfaz gráfica pública del objeto virtual STDD entrada de cine.

7.6.2.4 Fichero de configuración

El fichero de configuración es un documento XML, contiene información acerca de la identidad y comportamiento del objeto virtual STDD.

Los valores que toman las propiedades más importantes tienen que ver con el comportamiento que le ha sido asignado al objeto virtual STDD durante la fase de análisis. Propiedades como: único, copiable, transferible, etc. El resto de propiedades son referencias a ficheros y recursos. El fichero de configuración para el objeto virtual entrada de cine es el siguiente.

```
<?xml version="1.0" encoding="UTF-8"?>
<STDD>
    <name>Ticket: Robot Movie</name>
    <type>www.CinemaTicketSTDD.com</type>
    <id unique="YES">47236271</id>
    <transferable>YES</transferable>
    <expiration>12-10-2009</expiration>
    <editable>NO</editable>
    <copy>NO</copy>
    <interface
        private="privateInterface.xml"
        public="publicInterface.xml"
        api="methods.xml">
</interface>
    <code android="robotMovie9780.apk"></code>
    <data>info.obj</data>
    <mainClass>com.robotmovie.EntradaCine</mainClass>
    <icon>icon_tiket.png</icon>
</STDD>
```

7.6.3 Ejemplo de uso

A continuación se describirá como podría implantarse y utilizarse este tipo de objeto virtual "Entrada de cine".

7.6.3.1 Adquisición de la entrada de cine

El archivo correspondiente al objeto entrada de cine podría adquirirse por uno de estos medios (Fig.7.22):

- **Internet:** A través de una página web, realizando el correspondiente pago se podría descargar el fichero de la entrada virtual.
- **Transferencias**: En el caso de que el dispositivo móvil no dispusiese de conexión a Internet, la entrada podría descargarse desde un ordenador y luego ser transferida al teléfono móvil.
- **Servidores específicos:** Otra posibilidad sería a través de sistemas embebidos de venta de entrada, en el caso de los centros comerciales podrían instalar "servidores de venta de entradas", los usuarios podrían acceder a ellos utilizando sus dispositivos móviles a través de Bluetooth u otros protocolos.

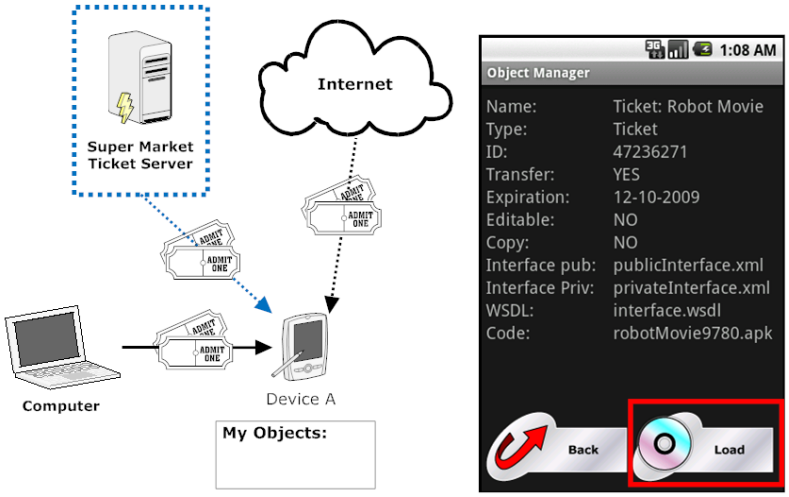

Figura 7.22 - Adquisición y carga de objeto virtual STDD en el dispositivo.

7.6.3.2 Canjear la entrada

Los objetos cargados aparecen en una lista al abrir el Gestor STDD. Para abrir un objeto virtual el usuario tiene que seleccionarlo en la lista. La siguiente pantalla le llevará a un menú general desde donde puede abrir el objeto y realizar las acciones de tipo general: borrar, enviar… solo aquellas que hayan sido especificadas por el desarrollador (Fig.7.23).

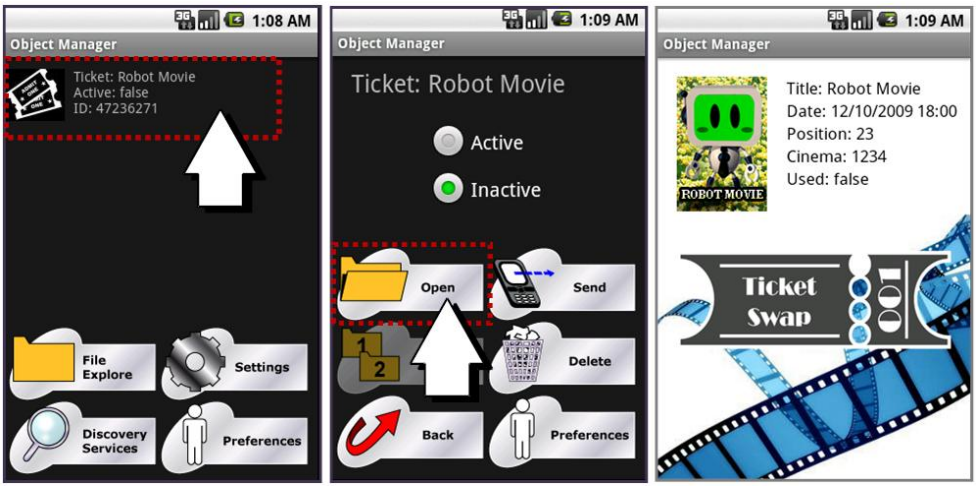

Figura 7.23 - Transición de pantallas: selección y apertura del objeto virtual STDD entrada de cine.

Para utilizar la entrada el usuario debe pulsar en el botón **Ticket Swap** cuando se encuentre cerca del servidor del cine; esta acción iniciará la sincronización con el servidor para tratar de validar la entrada de cine. Si se finaliza con éxito la validación el valor **Used** de la entrada cambia a true y el servidor dará luz verde para entrar en la sala de cine (Fig.7.24).

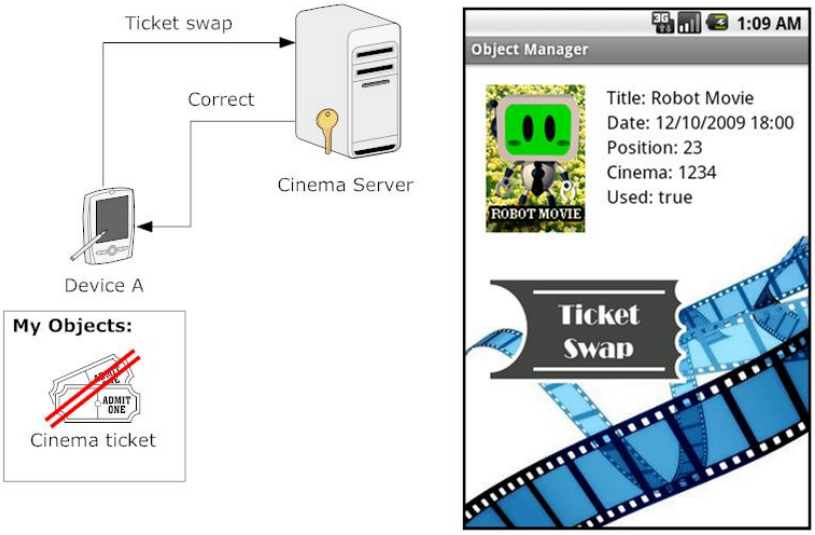

Figura 7.24 - Validación del objeto virtual STDD entrada de cine contra el servidor de la sala de cine.

7.6.3.3 Transferencia de entradas

La entrada podría ser transferida al dispositivo de otro usuario y este cargarla en su Gestor STDD para posteriormente canjearla. Para realizar la transferencia de objetos virtuales STDD hay que seleccionar la entrada en la lista de objetos cargados y pulsar el botón **Send**, esta acción dará comienzo a la búsqueda de dispositivos que se encuentren en el radio de cobertura Bluetooth. Cuando se encuentren dispositivos compatibles se podrá transferir la entrada; en este caso se trata de un objeto virtual único por lo que solo se podrá transferir el objeto original, no se permiten copias.

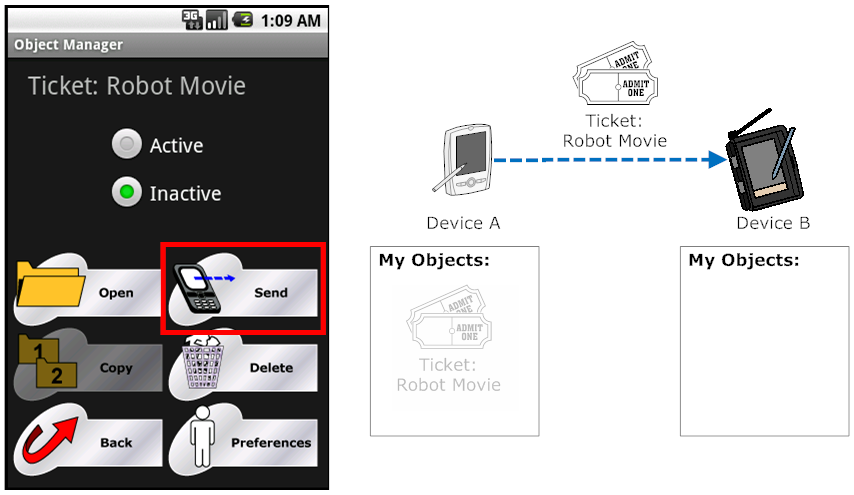

Figura 7.25 - Transferencia entre dispositivos del objeto virtual STDD entrada de cine.

7.6.3.4 Consulta de entrada por usuarios

Como el objeto virtual tiene habilitada una interfaz gráfica pública otros usuarios pueden descubrirlo, siempre que se encuentre en estado activo. Para activarlo hay que seleccionar la entrada en la lista de objetos cargados y pulsar el botón **Active**. Una vez hecho esto otros usuarios pueden descubrir el objeto virtual entrada y ejecutarlo de forma remota.

Desde la visión del usuario que no es dueño del objeto virtual, la perspectiva sería la siguiente: el usuario puede descubrir objetos virtuales alojados en otros dispositivos, una vez descubiertos estos objetos se le muestran en una lista de manera muy similar a los objetos virtuales propios. Al pulsar sobre uno de los objetos descubiertos se inicia la interacción con el servidor y la interfaz pública del objeto se muestra en pantalla (Fig.7.26).

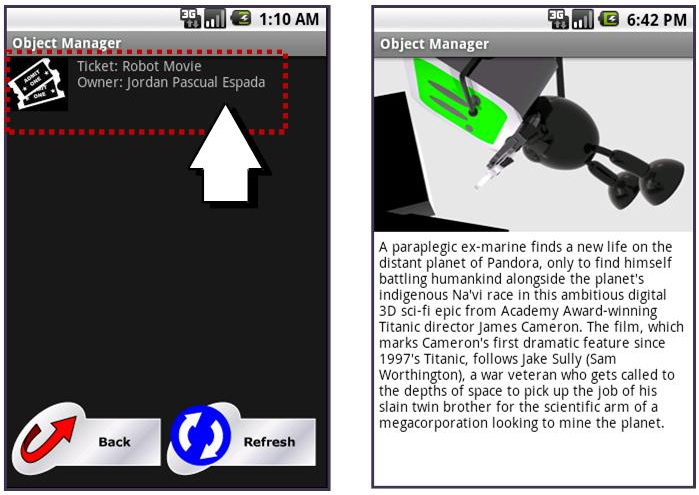

Figura 7.26 - Descubrimiento y conexión remota al objeto virtual STDD entrada de cine.

CAPÍTULO 8

OBJETOS VIRTUALES V2.0 DISTRIBUIDOS

8.1 EVOLUCIÓN DEL MODELO PROPUESTO

La primera versión de la propuesta **Objetos Virtuales v1.0 STDD** presentaba una serie de deficiencias. Las deficiencias más visibles se encontraban en el ámbito de la seguridad, debido a que el modelo propuesto caía en la necesidad de intercambiar módulos de código ejecutable de manera frecuente. A pesar de barajar varios enfoques capaces de combatir varias de las potenciales deficiencias de seguridad, se llegó a la conclusión de que el modelo planteado no era óptimo y debía ser rediseñado. Otro de los aspectos que contribuyó a tomar esta decisión fueron los notables costes de desarrollo necesarios para generar objetos virtuales multiplataforma, ya que una parte importante del objeto virtual (la lógica de negocio) debía ser implementada en varios lenguajes de programación para garantizar la compatibilidad de este con distintas plataformas.

Por lo tanto, se planteó una segunda fase de diseño que tenía como objetivo solucionar los problemas detectados en la versión inicial de la propuesta, a través del desarrollo de un nuevo modelo que se adaptase de una manera más óptima al cumplimiento de los objetivos fijados en esta tesis.

La arquitectura **Objetos Virtuales v2.0 Distribuidos** traslada la ejecución de la lógica de negocio (código ejecutable) de los objetos virtuales a un servidor externo. Este nuevo enfoque sigue la línea de las aplicaciones distribuidas, suprimiendo la necesidad de transferir de manera constante ficheros con código ejecutable potencialmente inseguro (Fig.8.1). El nuevo enfoque permite mantener parte de los elementos definidos en la especificación **Objetos Virtuales v1.0 STDD,** como interfaces gráficos, propiedades de los objetos, etc. Sin embargo requiere cambios significativos en la especificación, arquitectura del **Manager de objetos virtuales** y el desarrollo de un nuevo tipo de servidor, diseñado específicamente para la ejecución de objetos virtuales distribuidos.

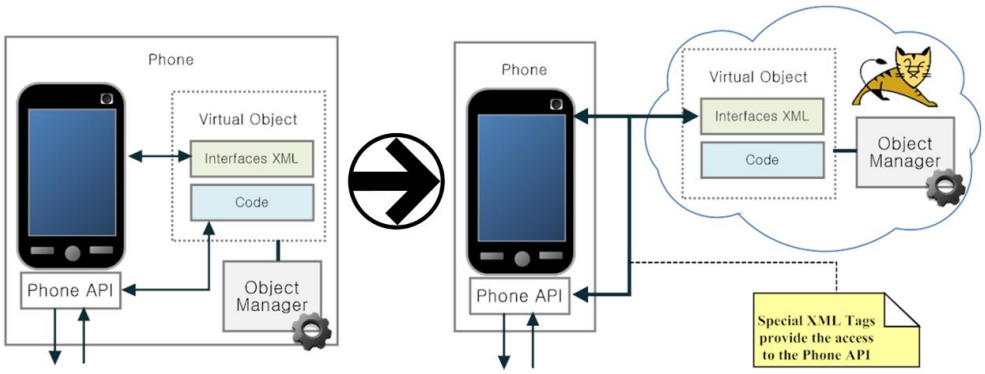

Figura 8.1 – Evolución del modelo Objetos virtuales STDD a Objetos virtuales distribuidos.

En este nuevo modelo el **servidor de objetos virtuales distribuidos** será el responsable de ejecutar la mayor parte de la lógica de negocio del objeto virtual, a la vez que gestiona el intercambio de peticiones y respuestas con el dispositivo cliente. Al utilizar una especificación única basada en Apache Tomcat[10] para desarrollar el servidor de objetos virtuales, la lógica del objeto virtual deberá ser implementada en java, independientemente de la plataforma usada por el dispositivo cliente.

Pero no toda la lógica de negocio del objeto virtual puede ser ejecutada en la parte del servidor. Algunas acciones, como las que implican la gestión de los elementos hardware del dispositivo deben ser ejecutadas en el propio dispositivo cliente. Para ello, la especificación definirá un conjunto propio de etiquetas denominado "*elementos de contexto*", las cuales indicarán al dispositivo cliente que debe ejecutar un determinado proceso. Estos procesos estarán definidos en el dispositivo cliente de antemano, cada uno de ellos implicará utilizar el API de la plataforma para gestionar uno de los elementos hardware del dispositivo, sensores, GPS, Bluetooth, etc. A través de este modelo el objeto virtual que se ejecuta remotamente en el servidor web será capaz de definir acciones que empleen los elementos hardware del propio dispositivo cliente.

Para poder ejecutar los objetos virtuales distribuidos el dispositivo cliente deberá ejecutar una versión modificada del antiguo **Gestor STDD,** renombrado como **visor de objetos virtuales distribuidos.** Esta aplicación permitirá solicitar e interpretar las interfaces gráficas XML de los objetos virtuales distribuidos. Una de las responsabilidades del visor de objetos será detectar las etiquetas XML especiales "*elementos de contexto*", utilizadas para indicar que se debe proceder a la ejecución de una de las tareas predefinidas que gestionan los elementos hardware del dispositivo. La lógica de estas nuevas tareas estará definida en el propio visor de objetos virtuales. Tras la ejecución de la tarea específicada por el elemento de contexto el visor de objetos virtuales distribuidos remitirá la respuesta generada al objeto virtual alojado en el servidor de objetos virtuales distribuidos. Mediante este procedimiento el objeto virtual podrá incluir en sus procesos de negocio acciones que empleen las respuestas generadas por las tareas especificas ejecutadas en el dispositivo cliente. Estas respuestas serán dependientes del tipo de tarea que se haya ejecutado, por ejemplo: si la tarea ha consultado el valor actual del sensor de aceleración la respuesta será una cadena de texto, si se ha tomado una fotografía con la cámara del dispositivo la respuesta será un fichero png, si se ha realizado un escaneo en busca de dispositivos Bluetooth la respuesta será una lista con los nombres y direcciones de los dispositivos descubiertos (Fig.8.2).

[10] http://tomcat.apache.org/

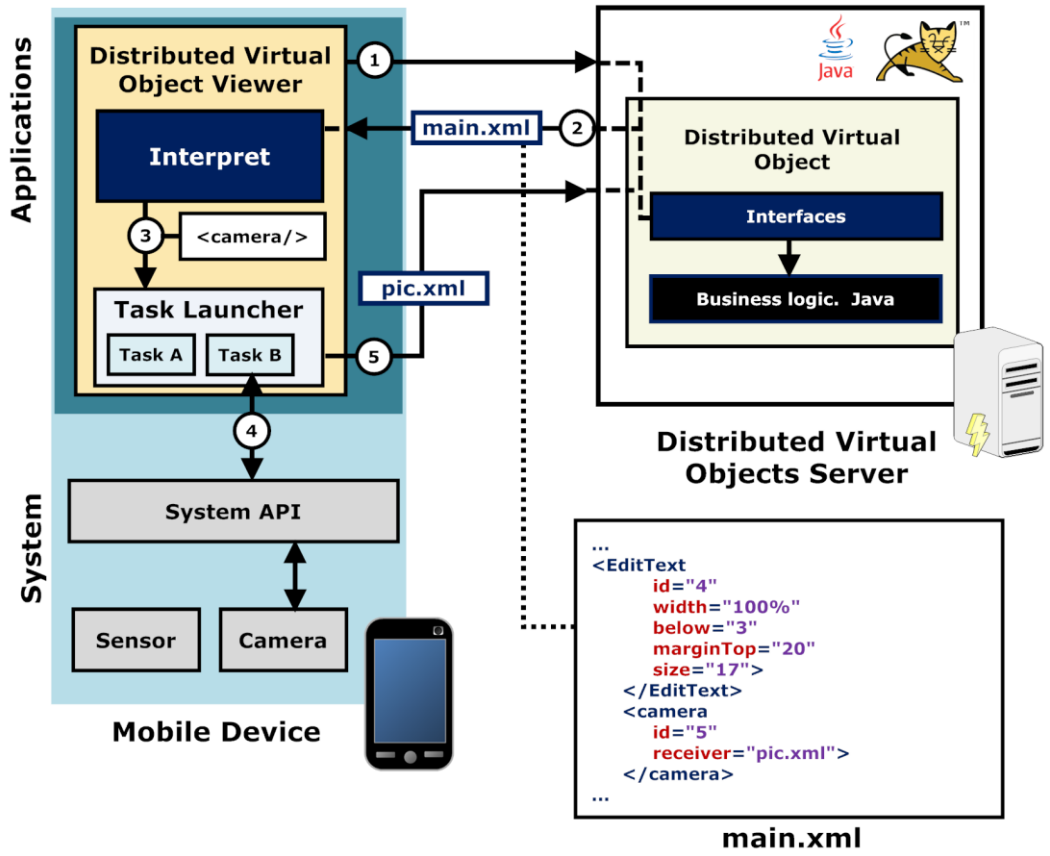

Figura 8.2 - (1) El visor de objetos virtuales distribuidos realiza una petición al servidor. (2) El servidor responde con un fichero XML que contiene elementos gráficos comunes y elementos de contexto. (3) El intérprete procesa el contenido del fichero XML, representado el interfaz por pantalla y ejecutando las tareas programadas correspondientes a los elementos de contexto. (4) Las tareas programadas utilizan los elementos hardware del dispositivo para capturar información de contexto o para comunicar al cliente con otros dispositivos. (5) El resultado de la ejecución de la tarea es procesado y enviado al servidor de objetos virtuales distribuidos.

8.2 OBJETOS VIRTUALES DISTRIBUIDOS

La estructura de un **Objeto Virtual v1.0 STDD** estaba compuesta por un conjunto de archivos: descriptor, interfaces gráficas públicas y privadas, ficheros de código ejecutable, etc. Cada uno de los archivos que formaban parte de la estructura cumplía un propósito específico. Con la evolución de la propuesta **Objetos Virtuales v 2.0 Distribuidos** y la nueva orientación hacia los sistemas distribuidos, la estructura inicial precisa de una serie de modificaciones.

En lugar de en el propio dispositivo cliente los nuevos objetos virtuales distribuidos estarán alojados en un servidor remoto, denominado **servidor de objetos virtuales distribuidos**, este servidor presenta una arquitectura mixta que fusiona aspectos de un servidor de aplicaciones Apache Tomcat y un Gestor de objetos virtuales STDD. La lógica de negocio de los objetos virtuales distribuidos estará implementada empleando el lenguaje Java, al ejecutarse de manera remota en un servidor externo no resulta necesario incluir implementaciones en diversos lenguajes de programación. Por lo tanto la nueva estructura del objeto virtual contendrá un único fichero de código ejecutable Java.

Uno de los cambios más significativos en la estructura del objeto virtual afecta a la definición de las interfaces gráficas de usuario. Los ficheros que definen estas interfaces estarán alojados en la parte del servidor y serán transmitidos al dispositivo cliente para que éste los interprete. La aplicación encargada de realizar el procesamiento de los interfaces será el **visor de objetos virtuales distribuidos**. Como resultado del procesamiento se cargara en la pantalla del dispositivo cliente el interfaz de usuario, a partir de la interacción del usuario con determinados elementos de esta interfaz (pulsar un botón, marcar un CheckBox, etc) se generarán respuestas que serán enviadas al **servidor de objetos virtuales distribuidos.** Como en este nuevo modelo los objetos virtuales siempre se ejecutan de manera remota, deja de tener justificación la utilización de interfaces privadas (si se accede desde el propio dispositivo) y públicas (si se accede remotamente desde el otro dispositivo). Los objetos virtuales distribuidos contendrán un único tipo de interfaces de usuario, el control de acceso a éstas podrá ser implementado en la lógica de negocio del propio objeto, mediante contraseñas, validaciones, etc.

El conjunto de elementos que componen el objeto virtual distribuido serán encapsulados en un fichero .war.

Tipo	Archivos:	Descripción
Descriptor	1	Fichero XML, contiene información acerca de la configuración, despliegue y ejecución del objeto.
Interfaces Gráficas	1..*	Ficheros XML de tipo interfaz gráfica STDD. Cada uno representa una pantalla.
Interfaces		Interfaces para que las acciones del objeto virtual puedan ser accedidas por aplicaciones y otros objetos virtuales.
WSDL	0..*	Descripción de servicios web.
Código ejecutable	1..*	Fichero Java que contiene el código ejecutable correspondiente a la lógica del objeto virtual.
Almacén de datos	0..1	Fichero que tiene como misión encargarse de la persistencia de datos, sólo debe incluirse en el caso de que sea requerido.
Recursos	0..*	Pueden incluirse un número de recursos ilimitados. Por lo general serán recursos multimedia: imágenes, iconos y videos. Serán utilizados para completar el apartado gráfico del objeto virtual STDD.

Tabla 8.1 – Conjunto de ficheros que forman la estructura del objeto virtual distribuido .war .

8.2 INTERFACES DE USUARIO Y ELEMENTOS DE CONTEXTO

La versión anterior de la propuesta definía un gran conjunto de elementos que podían ser utilizados para componer las interfaces de usuario, entre los que se encuentran: layouts, TextView, EditText, ImageView, etc.

Estos elementos se utilizaban para especificar interfaces de usuario en XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout id="1" background="fondocuestionario.png">
    <ImageView
        id="2"
        width="100%"
        height ="20&"
        file ="logotipo1.png"
        margin="3%">
    </ImageView>
    <TextView
        id="3"
        width="100%"
        text="Nombre: "
        below="2"
        margin="55"
        color="0,0,0"
        size="17">
    </TextView>
    <EditText
        id="4"
        width="100%"
        below="3"
        marginTop="20"
        size="17">
    </EditText>
    <Button
        id="5"
        width="100%"
        height="10%"
        bellow="6"
        action="#procesarUsuario(string:4)"
        margin="20">
    </Button>
</RelativeLayout>
```

El modelo de los **Objetos virtuales v 2.0 distribuidos** incluye de manera adicional un nuevo grupo de elementos llamado “elementos de contexto” que hacen referencia a acciones que el dispositivo cliente puede ejecutar mediante la gestión de sus elementos hardware. Como: obtener una ubicación, la medición de un sensor, enviar una cadena de datos utilizando la conexión Bluetooth. A diferencia de los elementos definidos en la versión anterior de la propuesta los elementos de contexto no son puramente gráficos, además tienen como objetivo especificarle al dispositivo cliente una tarea que debe llevar a cabo, para ello pueden contener diversos atributos que detallan la configuración de la misma.

Los elementos de contexto definidos son los siguientes:

Nombre	Descripción
magneticfield	Obtiene el valor actual registrado por el sensor de campo magnético.
proximity	Obtiene el valor actual registrado por el sensor de proximidad.
orientation	Obtiene el valor actual registrado por el sensor de orientación.
accelerometer	Obtiene el valor actual de la aceleración del dispositivo en los ejes X, Z y Y.
gravity	Obtiene el valor actual registrado por el sensor de gravedad.
lightSensor	Obtiene el valor actual del sensor de luz del dispositivo.
temperature	Obtiene el valor actual del sensor de temperatura.
camera	Utiliza la cámara del dispositivo para tomar una foto.
location	Obtiene la localización actual del dispositivo.
bluetooth	Utiliza el módulo de comunicaciones Bluetooth del dispositivo para realizar diferentes acciones de comunicación.
wi-fi	Utiliza el módulo de comunicaciones Wi-Fi del dispositivo para realizar diferentes acciones de comunicación.
nfc	Utiliza el módulo de comunicaciones NFC del dispositivo para realizar procesos de lectura y escritura de etiquetas.

Tabla 8.2 – Elementos de contexto.

Cada uno de los elementos definidos podrá ser combinado con un conjunto de atributos propios los cuales permiten especificar diversos detalles, estos atributos se especificarán en el apartado *9.4 Etiquetas XML de contexto*.

Los elementos de contexto se incluirán en los ficheros XML que definen las interfaces de usuario:

```
<EditText
    id="4"
    width="100%"
    below="3"
    marginTop="20"
    size="17">
</EditText>
<camera
    id="5"
    receiver="imagen.xml">
</camera>
<Button
    id="6"
    width="100%"
    height="10%"
    bellow="6"
    action="#procesarUsuario(string:4)"
    margin="20">
```

8.3 CONCLUSIONES

La versión **Objetos Virtuales v2.0 Distribuidos** no llegó a ser especificada ni implementada de manera completa, ya que durante su desarrollo se decidió realizar un cambio de enfoque, evitando que la especificación redefiniera elementos similares a los presentes en otro tipo de tecnologías empleadas en el desarrollo de aplicaciones distribuidas. Se decidió centrar el alcance de la especificación exclusivamente en las características y funcionalidades no contempladas (o al menos no de la manera requerida) en la mayor parte de las tecnologías aplicadas en el desarrollo de aplicaciones distribuidas. Estas características y funcionalidades, en las que se centraría la nueva versión del modelo **Objetos virtuales v3.0 En la web** eran principalmente aquellas relativas a la gestión de los elementos hardware del dispositivo cliente.

CAPÍTULO 9

OBJETOS VIRTUALES V3.0 EN LA WEB

9.1 INTRODUCCIÓN

La tercera versión de la propuesta **Objetos virtuales v3.0 En la web** se centra en aquellas características y funcionalidades requeridas para la construcción objetos virtuales que no están soportadas por las tecnologías web actuales. En esta versión de la propuesta se presenta un modelo capaz de aportar un conjunto de nuevas funcionalidades relativas a la gestión de los elementos hardware del dispositivo cliente, este modelo podrá ser combinado con la mayoría de las tecnologías web más populares empleadas en el desarrollo de aplicaciones distribuidas (Fig.9.1).

Introduciendo el uso de las tecnologías web tradicionales en el proceso de desarrollo de los objetos virtuales se consigue adaptar el modelo de una manera más óptima a varios de los objetivos planteados en esta tesis. Adicionalmente, este nuevo enfoque también puede ser aplicado en el desarrollo de otro tipo de aplicaciones web, o incluso añadido a aplicaciones web ya desarrolladas con el fin de optimizar su funcionalidad. La posibilidad de integrar el modelo propuesto con una amplia gama de tecnologías web mejora su difusión y facilita el desarrollo de los objetos virtuales. La independencia del modelo **Objetos virtuales v3.0 En la web** permitirá integrarlo con diferentes tecnologías web y facilitará que las aplicaciones web generadas puedan ser ejecutadas en servidores de aplicaciones tradicionales sin requerir ninguna modificación o plug-in adicional.

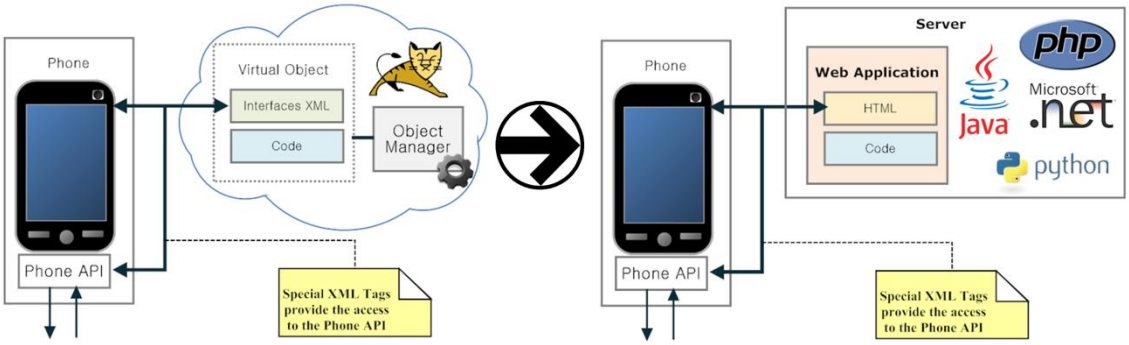

Figura 9.1 – Evolución del modelo Objetos virtuales distribuidos a Objetos virtuales en la web.

Los interfaces de usuario de los **Objetos Virtuales v2.0 Distribuidos** empleaban un grupo especifico de etiquetas XML que permitían especificar requerimientos que implicaban el uso de los elementos hardware del dispositivo cliente, tales como cámara de fotos, sensor de movimiento, GPS, NFC, etc. El procesamiento de estas etiquetas XML era realizado por un módulo de software especifico, el cual ejecutaba determinadas tareas programadas en función de las etiquetas XML procesadas. El nuevo modelo presentado en este capítulo permite integrar la utilización del conjunto especial de etiquetas XML con

gran parte de las tecnologías web actuales. Esta nueva versión de la propuesta presenta una especificación de diseño de un Navegador Web Sensible al contexto que adopta parte de las funcionalidades propias de la arquitectura de los **Objetos Virtuales v2.0 Distribuidos** (Fig.9.2).

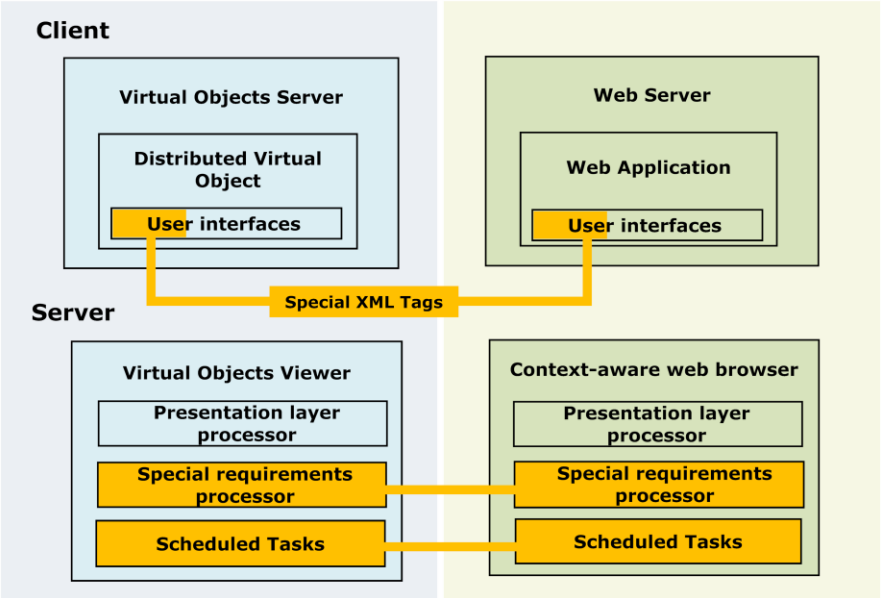

Figura 9.2 – Comparación entre los Objetos virtuales Distribuidos y los Objetos virtuales En la web. Las partes resaltadas señalan los componentes comunes o equivalentes presentes en ambos modelos.

En el capítulo 7 se definió el término **objeto virtual** como: "*Componente software que se combina con un dispositivo electrónico con el objetivo de desarrollar una determinada funcionalidad, la cual puede llegar a hacer las veces de un objeto físico, implicando la interacción con objetos físicos, dispositivos electrónicos u otros elementos del entorno*". En ese mismo capítulo se presentó un modelo especialmente concebido para el desarrollo de objetos virtuales adaptados a los objetivos formulados en esta tesis. A este modelo se le denominó **Objetos Virtuales v1.0 STDD.** En el capítulo 8, la propuesta inicial **Objetos Virtuales v1.0 STDD** fue modificada sustancialmente, dando lugar al nuevo modelo **Objetos virtuales v2.0 Distribuidos**. Ambos modelos permitían la implementación y uso de objetos virtuales, pero lo hacían bajo diferentes arquitecturas y principios. Más adelante, se llego a la conclusión de que no resultaba óptimo especificar un modelo "completo" (que contemplara todos los niveles del desarrollo software), sino que se podía partir de una tecnología adecuada a la que dotar de aquellas características requeridas para el desarrollo y uso de objetos virtuales. Finalmente, en este capítulo se presenta el modelo **Objetos virtuales v3.0 En la web**. Éste puede ser combinado con las tecnologías web comunes para permitir el desarrollo de objetos virtuales basados en aplicaciones web. Debido a las propiedades de este modelo su uso no tiene porque limitarse a la creación de objetos virtuales puros, sino que puede ser empleado en el desarrollo de otro tipo de aplicaciones web, o para dotar de nuevas características a aplicaciones web ya desarrolladas.

9.2 MODELO

Este modelo está dividido en dos partes (Fig.9.3).

- **Servidor**. Por un lado se definen un **conjunto abierto de etiquetas XML,** que se incluyen en la capa de presentación de los objetos virtuales y aplicaciones web para expresar requerimientos que implican información de contexto o acciones de comunicación. Este conjunto de etiquetas es una evolución del subconjunto de etiquetas que se utilizaban en la definición de las interfaces de usuario en la versión anterior de la propuesta "Objetos virtuales v2.0 Distribuidos". En la versión anterior de la propuesta estas etiquetas eran procesadas por el ***visor de objetos virtuales distribuidos***, en función de la etiqueta procesada el componente determina las acciones que se deben ejecutar en el dispositivo cliente. Todas estas acciones tenían un aspecto común: empleaban un elemento hardware del dispositivo; el chip GPS, el módulo de comunicaciones Bluetooth, sensores de movimiento, la cámara, etc.
  Las etiquetas XML propuestas en este modelo con el objetivo de definir requerimientos de información de contexto o acciones de comunicación serán denominadas a partir de este momento como **etiquetas XML de contexto.**

- **Cliente**. En el otro lado está el **navegador web sensible al contexto**, esta aplicación es la responsable de procesar las etiquetas XML de contexto. Cuando el navegador sensible al contexto procesa una etiqueta XML de contexto conocida, notifica al usuario el requerimiento. Si el usuario acepta el requerimiento significa que permite al navegador web sensible al contexto gestionar el elemento hardware concreto capaz de satisfacerlo. Una vez el usuario acepta un requerimiento, el navegador web sensible al contexto ejecuta la ***tarea programada de contexto*** asociada a la etiqueta XML de contexto concreta. Por lo general las tareas programadas de contexto gestionan alguno de los elementos hardware del dispositivo, con el objetivo de capturar información de contexto o realizar algún tipo de comunicación. Cuando el navegador sensible al contexto finaliza la tarea programada de contexto remite el resultado a la aplicación web (Fig.9.4).
  Esta versión del navegador web sensible al contexto está basada en parte de la arquitectura del ***visor de objetos virtuales distribuidos***. El navegador web sensible al contexto ha sido específicamente diseñado para visualizar aplicaciones web, a diferencia del antiguo componente este navegador permite procesar lenguajes web (HTML, CSS, Javascript) de la misma forma que un navegador tradicional (Internet Explorer, Chrome, Safari, etc). Sin embargo cuando el navegador web sensible al contexto procesa una etiqueta XML de contexto lo hace de una forma similar al ***visor de objetos virtuales***.

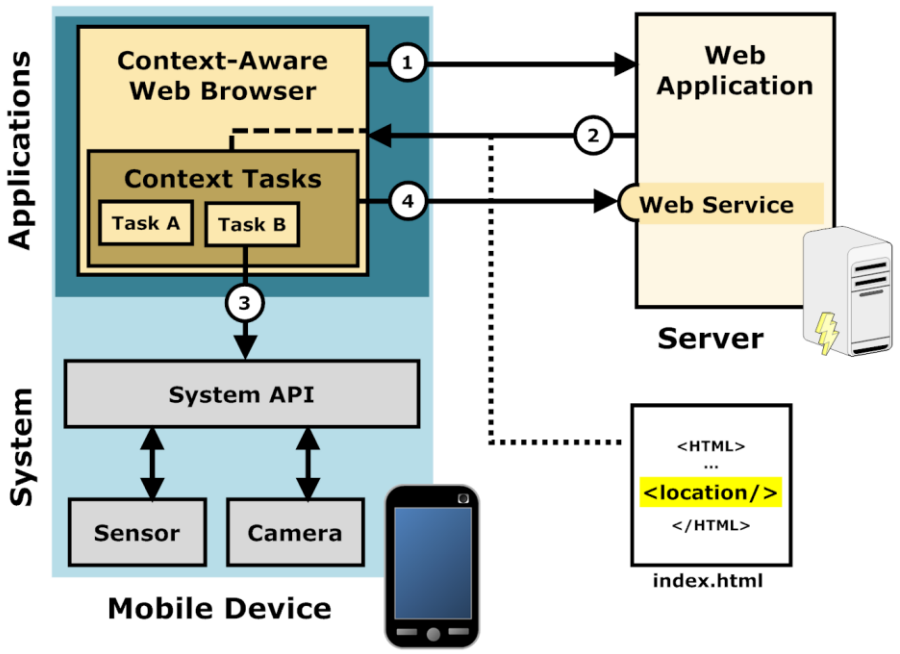

Figura 9.3 - (1) El navegador sensible al contexto realiza una petición al servidor web (2) El servidor web responde con una página web HTML que contiene etiquetas XML de contexto. (3) El navegador web sensible al contexto procesa el contenido del fichero HTML y ejecuta las tareas programadas de contexto correspondientes. (4) Las tareas programadas de contexto utilizan los elementos hardware del dispositivo para capturar la información de contexto, el resultado es procesado y enviado al servidor web.

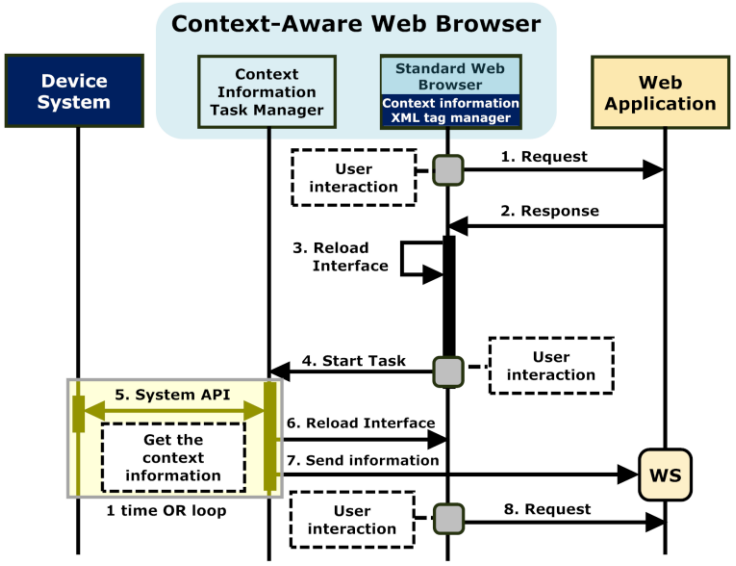

Figura 9.4 - Diagrama de secuencia simplificado. Procesamiento de una petición de información de contexto.

9.3 APLICACIONES WEB Y ETIQUETAS XML DE CONTEXTO

Las etiquetas XML de contexto sirven para expresar requerimientos de información de contexto o acciones de comunicación; estas etiquetas se incluyen en la capa de presentación de los objetos virtuales y las aplicaciones web, como puede ser el código HTML. El uso de estas nuevas etiquetas XML de contexto es totalmente compatible con los navegadores web tradicionales que no implementan la especificación del navegador sensible al contexto. Los navegadores web tradicionales simplemente se dedican a ignorar las etiquetas XML que desconocen. Cada etiqueta XML de contexto notifica al navegador sensible al contexto un tipo de requerimiento específico. En muchas ocasiones un mismo tipo de información de contexto puede ser capturada de diferentes modos, aplicando distintos procedimientos o configuraciones. Para que los requerimientos puedan ser expresados con la suficiente precisión cada etiqueta XML de contexto estará asociada a un conjunto de atributos, estos atributos pueden ser incluidos de manera opcional con el fin de configurar los procesos de captura, procesamiento o respuesta. Además, algunos de los atributos que pueden ser asociados a las etiquetas XML de contexto se utilizan para otros propósitos, como el de identificar una etiqueta XML de contexto concreta dentro de una aplicación web.

Esta propuesta define un conjunto abierto de etiquetas XML de contexto, en el que tienen cabida la mayoría de los tipos de información de contexto y acciones de comunicación que los Smartphones actuales son capaces de manejar. Seguramente no todos los dispositivos móviles serán capaces de dar soporte al conjunto completo de etiquetas XML de contexto definido, el factor de compatibilidad será dependiente de las características técnicas del dispositivo en cuestión. El navegador web sensible al contexto será el encargado de notificar al usuario si existe algún requerimiento especial por parte de la aplicación web que el dispositivo móvil cliente no esté en condiciones de satisfacer, por ejemplo debido a que no dispone del hardware adecuado, o que el elemento hardware necesario no está preparado; un caso típico seria la ausencia de cobertura GPS cuando una aplicación web solicita la localización actual.

El conjunto de etiquetas XML de contexto se divide en tres categorías:

- **Etiquetas XML de contexto básicas**. Corresponden a los requerimientos básicos de información de contexto.

- **Etiquetas XML de contexto complejas**. Corresponden a los requerimientos de información de contexto que implican ciertos filtrados o procesamientos de los datos capturados.

- **Etiquetas XML de contexto basadas en acciones de comunicación**. Corresponden a los requerimientos de acciones de comunicación destinadas a establecer o recibir comunicaciones con dispositivos o elementos cercanos.

9.3.1 Etiquetas XML de contexto básicas

Las etiquetas XML de contexto enmarcadas en esta categoría sirven para especificar requerimientos de información de contexto que podríamos denominar como básicos. Estos requerimientos solicitan información de contexto que será obtenida utilizando los elementos hardware del dispositivo. Una vez completado el proceso de obtención de la información de contexto, esta será enviada a la aplicación web sin aplicar sobre ella ningún tipo de procesamiento.

El conjunto de etiquetas XML de contexto que expresan los requerimientos de información de contexto básica son los siguientes:

Posición.

<magneticfield/>
Obtiene el valor actual registrado por el sensor de campo magnético. Este valor consta de tres componentes, los cuales contienen el valor de la fuerza del campo geomagnético a lo largo de los ejes X, Y y Z; el valor de las mediciones esta expresado en **µT** microteslas. Este sensor es útil para detectar la cercanía de ciertos objetos metálicos.

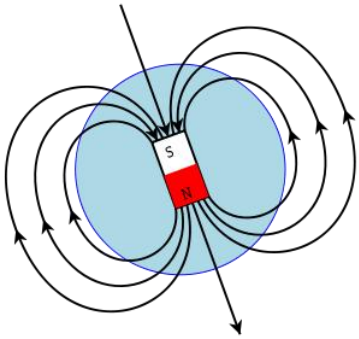

Figura 9.5 - Las líneas de la fuerza del campo magnético de la Tierra son mostradas en un corte longitudinal que pasa a través del eje magnético

<proximity/>
Obtiene el valor actual registrado por el sensor de proximidad. Los sensores de proximidad son capaces de detectar algunos objetos físicos cercanos sin que llegue a producirse contacto alguno. Este sensor suele estar localizado cerca del altavoz del teléfono. El valor de su medición esta expresado en cm.

<orientation/>
Obtiene el valor actual registrado por el sensor de orientación. Este valor está compuesto por tres coordenadas que expresan la orientación del dispositivo en los ejes X, Y y Z; el valor de las mediciones está expresado en grados**.**

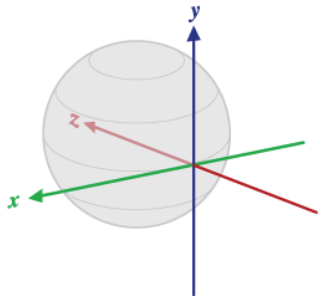

Figura 9.6 - Representación gráfica de los ejes de referencia tomados por el sensor de orientación del dispositivo.

Movimiento

<accelerometer/>
Es el sensor de movimiento más utilizado. Obtiene el valor actual de la aceleración del dispositivo en los ejes X, Z y Y; el valor de la aceleración se expresa con números enteros que pueden tomar valores negativos dependiendo de la orientación de la aceleración.

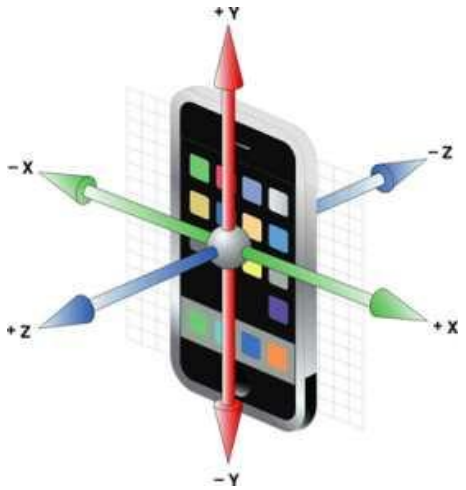

Figura 9.7 - Representación gráfica de los ejes de referencia tomados por el sensor de aceleración del dispositivo.

<gravity/>
Obtiene el valor actual registrado por el sensor de gravedad. La medición se expresa en un valor decimal m/s^2.

Entorno

<lightSensor/>
Obtiene el valor actual del sensor de luz del dispositivo. La luz ambiental se expresa en un número decimal comprendido entre 0 y 1.

<temperature/>
Obtiene el valor actual del sensor de temperatura. La medición se expresa en un valor entero que indica los grados.

Multimedia

<audiocapture/>
Obtiene una grabación de audio utilizando el micrófono del dispositivo. El usuario es el encargado de indicar cuál es el momento de inicio y de fin de la grabación. El resultado de la acción será un fichero de audio preferentemente en formato ogg o wav, aunque puede variar dependiendo de las características del dispositivo. Los atributos opcionales propios de esta etiqueta son útiles para configurar el proceso de grabación de audio permitiendo personalizar factores como el tipo de formato preferente y el tiempo máximo de grabación.

<camera/>
Utiliza la cámara del dispositivo para tomar una foto. El resultado de la acción será un fichero de tipo imagen preferentemente en formato jpeg / jpg, aunque puede variar dependiendo de las características del dispositivo. Los atributos opcionales propios de esta etiqueta son útiles para configurar la cámara de fotos antes de que se tome la fotografía permitiendo personalizar factores como el tipo de formato preferente.

Localización

<location/>
Obtiene la localización actual del dispositivo. La localización actual está definida por las coordenadas geográficas latitud y longitud ambos valores se expresan como números decimales separados por una coma. Los atributos opcionales propios de esta etiqueta permiten especificar la fuente de la localización seleccionando entre Wi-Fi o GPS.

9.3.2 Etiquetas XML de contexto complejas

Al aplicar una serie de procesos y algoritmos sobre la información de contexto capturada en muchos casos se consigue dotar de un valor añadido a la propia información. Existen varios mecanismos y procesos de tratamiento de información de contexto que están ampliamente difundidos. Por ejemplo, una imagen capturada por una cámara de fotos seria información de contexto básica, pero si la imagen fuese analizada con el fin de obtener la información contenida en un código de barras se podría considerar como información de contexto procesada o compleja (Fig.9.8). En el caso de querer reconocer un código de barras el navegador web sensible al contexto podría remitir a la aplicación web únicamente los datos decodificados contenidos en el código de barras, evitando tener que enviar el fichero con la imagen. En este caso se estaría aplicando un procesamiento sobre un tipo de información de contexto básica con el fin de obtener otro tipo de información.

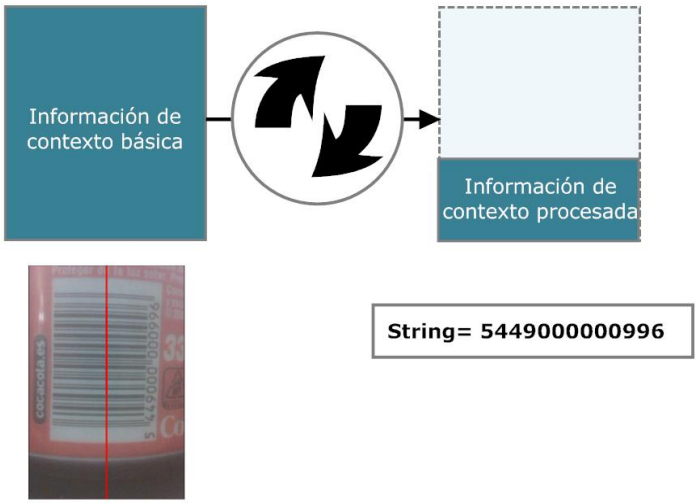

Figura 9.8 - En algunos casos el procesamiento de la información de contexto básica reduce la información capturada exclusivamente al contenido útil.

El conjunto de etiquetas XML de contexto que expresan los requerimientos de información de contexto básica son los siguientes:

Multimedia

<camerabarcode/>
Utiliza la cámara del dispositivo como escáner visual para decodificar un código de barras lineal. El resultado de la acción será una cadena de texto con la información contenida en el código de barras.

<cameraqrcode/>
Utiliza la cámara del dispositivo como escáner visual para decodificar un código QR, estos códigos almacenan información en una matriz de puntos bidimensional. El resultado de la acción será una cadena de texto con la información contenida en el código QR.

Figura 9.9 - Ejemplo de código QR

< audiocaptureSpeech/>
Utiliza el micrófono del dispositivo para capturar la voz de las personas y traducirla a texto. El usuario es el encargado de indicar cuál es el momento de inicio y de fin de la grabación. El resultado de la acción será una cadena de texto con el contenido de la grabación aplicando técnicas de reconocimiento de voz. Los atributos de esta etiqueta permiten configurar el idioma de entrada que se utilizará en el proceso de reconocimiento de las palabras en la grabación de audio.

9.3.3 Etiquetas XML de contexto basadas en acciones de comunicación

Las etiquetas XML de contexto basadas en acciones de comunicación sirven para especificar diversos procesos que implican la recepción o transmisión de datos entre el dispositivo cliente y otros dispositivos electrónicos o etiquetas. Para realizar estos intercambios de datos se utilizarán los módulos de comunicación del dispositivo móvil, las principales especificaciones de comunicación con las que cuentan los Smartphones actuales son: Bluetooth, Wi-Fi y en menor medida NFC.

<bluetooth/>
Utiliza el módulo de comunicaciones Bluetooth del dispositivo para realizar diferentes acciones, entre las que se encuentran: emparejamientos, escaneos en busca de dispositivos cercanos, conexiones, desconexiones, recepción y envío de datos.

<wi-fi/>
Utiliza el módulo de comunicaciones Wi-Fi del dispositivo para realizar diferentes acciones, entre las que se encuentran: escaneos en busca de dispositivos cercanos, conexiones a dispositivos ad-hoc, desconexiones, recepción y envío de datos.

<nfc/>
Utiliza el módulo de comunicaciones NFC del dispositivo para realizar procesos de lectura y escritura de etiquetas.

9.3.4 Configuración y comportamiento de las etiquetas XML de contexto

Los usuarios visualizan los requerimientos expresados por las etiquetas XML de contexto contenidas en la capa de presentación de la aplicación web o el objeto virtual como botones, estos controles tienen unas características diferentes a las del propio código HTML por ello se encuentran fuera del propio proceso de visualización de la página web. Estos botones expresan de manera visual utilizando un icono identificativo (aunque puede ser modificado) el tipo de información de contexto o acción de comunicación que está siendo requerida por parte de la aplicación web. Cuando el usuario presiona sobre uno de estos botones está dando su consentimiento expreso para que el navegador sensible al

contexto proceda a satisfacer el requerimiento. El navegador web sensible al contexto ejecutará las tareas programadas de contexto oportunas que permitan satisfacer el requisito. Posteriormente el navegador web sensible al contexto se encarga de remitir la información capturada (o la respuesta) a la aplicación web. Las etiquetas XML de contexto pueden especificar una URL a donde será enviada la información una vez obtenida, esta información viajará encapsulada en una petición POST. En caso de no especificarse una URL destino la información será enviada a la URL por defecto asignada a cada etiqueta XML de contexto.

Se ha seleccionado la petición HTTP POST como medio de encapsulación de la información debido a su flexibilidad y popularidad. Las etiquetas XML de contexto pueden ser combinadas con multitud de lenguajes que habilitan el desarrollo de aplicaciones web, por ello se ha seleccionado un mecanismo de envío de datos ampliamente difundido. Desde un punto de vista técnico el procesamiento de las peticiones POST puede ser integrado de manera sencilla en la mayoría de tecnologías web (PHP, .Net, JSPs, Servlets, etc)

9.3.4.1 Clave de sesión de dispositivo

Las peticiones POST que encapsulan la información de contexto capturada por los elementos hardware del dispositivo contienen un código de identificación presente en todas las cabeceras HTTP, denominamos a este código como DSK (Device Session Key) "Clave de sesión de dispositivo". Este código se utiliza para que los servicios o aplicaciones que procesan la información de contexto puedan identificar al dispositivo que ha remitido la información y también desde qué sesión lo ha hecho, algo similar al objeto session que se aplica en las aplicaciones web.

Session: la mayoría de lenguajes que permiten el desarrollo de aplicaciones web incluyen un elemento de tipo session o similar, este elemento resulta muy útil para almacenar y posteriormente consultar información relativa a un usuario concreto. Cada session tiene un identificador único que se asigna a cada hilo de navegación, normalmente estos hilos son las peticiones que provienen de un navegador web concreto en una máquina. El proceso a seguir en la mayoría de lenguajes de programación es el siguiente: si existe una sesión activa para ese hilo de navegación se retoma la sesión anterior, si no existe se genera una sesión que contiene un identificador único y se asigna al hilo de navegación.

El DSK no sólo está adjunto en todas las cabeceras de las peticiones POST que transmiten información de contexto sino que también lo está en todas las peticiones del navegador web sensible al contexto. De esta forma se puede relacionar de manera relativamente simple a un usuario que accede a la aplicación web con la información de contexto que está siendo recibida en un momento determinado.

El DSK está formado por una combinación de caracteres que se obtienen a partir del código Imei único del dispositivo más un identificador que corresponde a la sesión actual del hilo de navegación. Procesando el DSK la aplicación web es capaz de saber si dos peticiones provienen de una misma sesión, incluso si provienen de un mismo dispositivo aunque sea en sesiones diferentes.

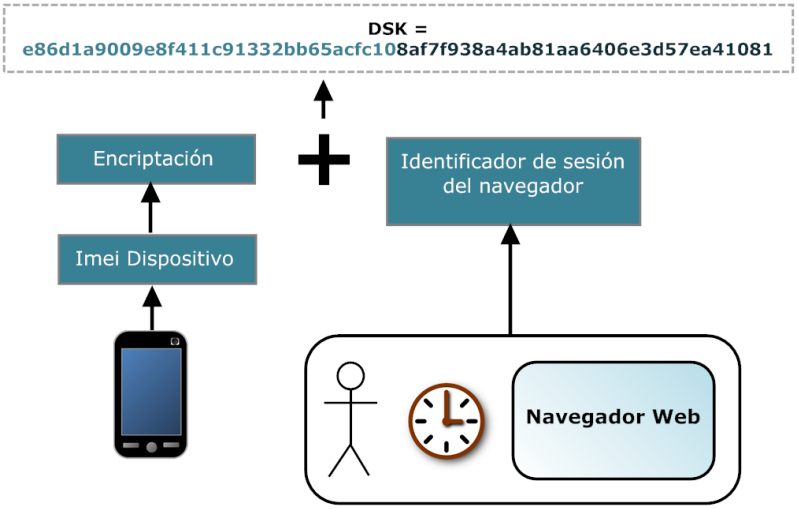

Figura 9.10 - El DSK se forma a partir de la combinación del identificador único del dispositivo y él identificador de sesión del navegador.

9.3.4.2 Receptor de datos

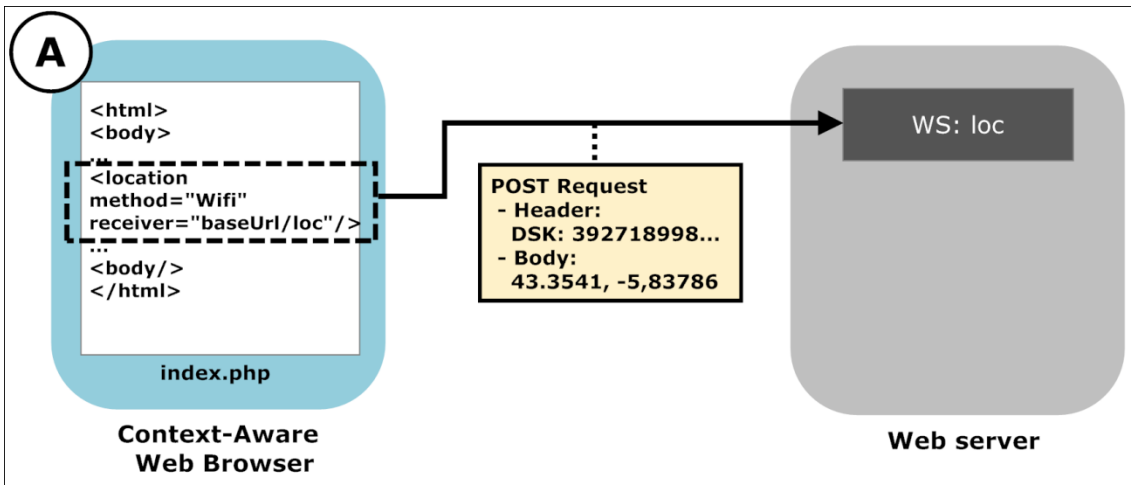

Figura 9.11 - Envío de datos por defecto.

9.3.4.3 Identificador

El atributo ID puede añadirse de manera opcional a las etiquetas XML de contexto. Si las etiquetas contienen este atributo su valor también viaja junto al DSK en las cabeceras de las peticiones POST que remiten la información de contexto a la aplicación web. De esta forma, el servicio encargado de procesar la información de contexto puede detectar como respuesta a qué etiqueta XML de contexto corresponde la información de contexto que ha recibido.

Cuando una aplicación web o un objeto virtual incluye varias etiquetas XML de contexto del mismo tipo, por ejemplo dos etiquetas <camera/>, es necesario incluir un atributo "id" en cada una de ellas, de esta forma pueden ser identificadas. El identificador de la etiqueta XML de contexto se incluye como cabecera en la petición POST que el navegador sensible al contexto envía a la aplicación web como respuesta del requerimiento. La aplicación web utilizará estos identificadores para determinar el origen de la información recibida.

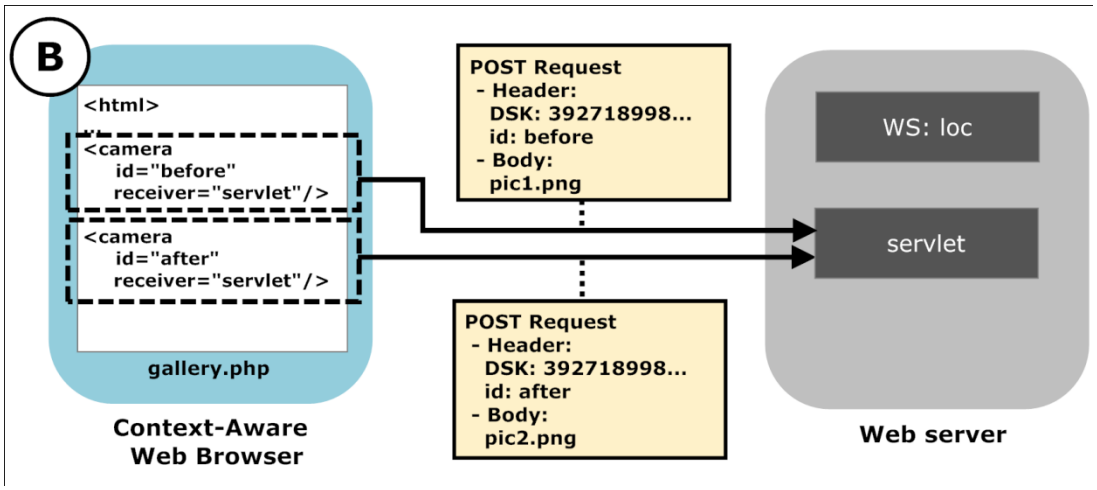

Figura 9.12 – Envío de datos basado en el uso del atributo ID para identificar la etiqueta XML de contexto.

9.3.4.4 Grupos

Conceptualmente las etiquetas XML de contexto tienen aspectos similares a los campos de un formulario web. En los formularios web clásicos la información de todos los campos se envía de manera conjunta cuando el usuario pulsa el botón de envío, obviamente este mecanismo resulta mucho más eficiente que realizar un envío individual por cada uno de los campos del formulario.

Cada uno de los botones que el navegador web sensible al contexto asocia a las etiquetas XML de contexto son en esencia entradas de datos, solo que estos datos en lugar de ser introducidos por los usuarios se obtienen de manera automática utilizando los elementos hardware del dispositivo móvil. Una misma página web puede contener varias etiquetas XML de contexto, por ejemplo un objeto virtual que detecta la contaminación acústica en una ciudad podría solicitar la ubicación actual y un fichero de audio. En este caso tanto la ubicación como el fichero de audio podrían ser procesados por un mismo receptor (servicio web), por lo que no sería estrictamente necesario enviar cada dato en una petición individual. En estos casos los datos pueden ser combinados en un único envío y de esta forma optimizar la comunicación entre el navegador y la aplicación receptora de los datos.

Combinando el uso del atributo "id" con el atributo "group" un grupo de etiquetas XML de contexto puede actuar de forma conjunta, realizando los envíos de datos en una única petición POST. Para definir un grupo compuesto por varias etiquetas XML de contexto, se debe incluir en las etiquetas un atributo "group", el valor de éste será el mismo en todas ellas: el identificador del grupo. Cuando los requerimientos de todas las etiquetas XML de contexto pertenecientes al grupo han sido satisfechos el navegador sensible al contexto envía una petición POST a la aplicación web, esta petición contiene en su cabecera el identificador del grupo "group" y el número de miembros "members". En el cuerpo de la petición se adjunta la respuesta correspondiente a cada una de las etiquetas XML de contexto.

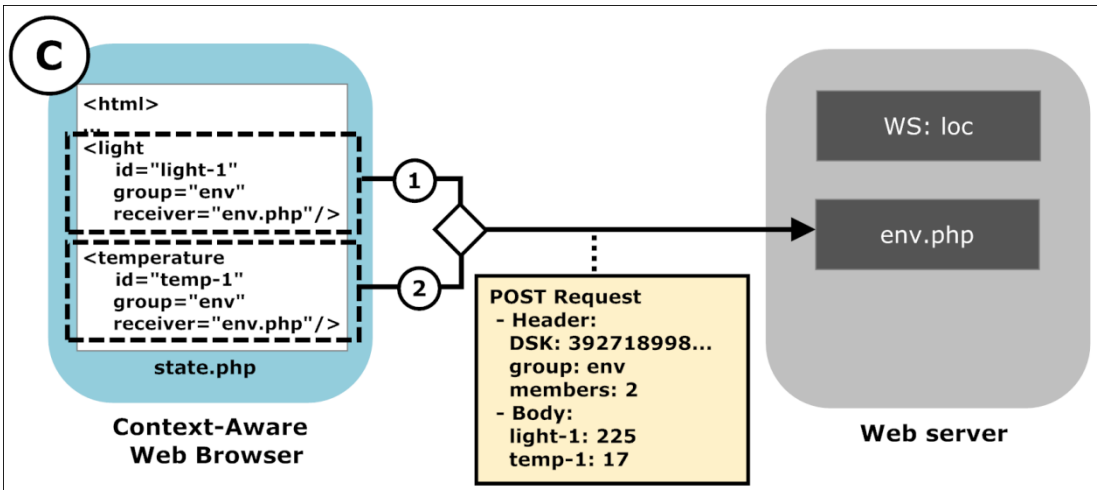

Figura 9.13 - Envío de datos basado en la utilización de grupos para unificar en una sola petición la respuesta de varias etiquetas XML de contexto.

9.4 Especificación: Etiquetas XML de contexto

9.4.1 Atributos comunes

Identificación y grupos

- id
- group

Bucles

- timeInterval

Recepción de información

- receiver

Representación grafica

- text
- image
- x
- y

Id

El atributo id se puede asignar a una etiqueta XML de información de contexto para identificarla de forma única.
El valor del atributo id definido en la etiqueta se adjuntará en la cabecera de todas las peticiones que se generan a raíz de la ejecución de la tarea programada asociada a la propia etiqueta.

Valores:

- **Identificador:** cadena de texto.

group

El atributo group se utiliza para indicar que varias etiquetas XML de contexto operan de manera conjunta.
Este atributo debe incluirse con el mismo valor en todas las etiquetas que forman parte de un mismo grupo. Las tareas programadas asociadas a las etiquetas de un mismo grupo realizan el envío de información de contexto de forma conjunta concentrando toda la información capturada en una única petición, de esta forma puede optimizase el tráfico de datos entre el navegador y el servidor.

Valores:

- **Nombre/Identificador del grupo:** cadena de texto.

Combinaciones de atributos:

- **Group** debe combinarse con el atributo **members.**
- **Group** debe combinarse con el atributo **id.**
- **Group** debe combinarse con el atributo **receiver.**

timeInterval

El atributo timeInterval se utiliza para establecer un intervalo de tiempo para aquellas capturas de información que deban repetirse periódicamente en forma de bucle.
Si el usuario da su permiso para satisfacer un requerimiento de información de contexto expresado por una etiqueta que contiene el atributo timeInterval la tarea programada asociada se repetirá en el intervalo de tiempo especificado hasta que la aplicación web se cierre.

Valores:

- **Intervalo de tiempo:** número entero de segundos.

receiver

El atributo receiver se utiliza para especificar la URL del servicio que procesará la respuesta del navegador web sensible al contexto al requerimiento expresado en la etiqueta XML de contexto.
En ausencia de la definición de atributo receiver cada etiqueta XML de contexto le asignará un valor por defecto.

Valores:

- **URL del servicio receptor.** La URL especificada en este atributo puede ser absoluta (http://www.miapplication.com/receptor) o relativa a la URL base de la aplicación (/receptor equivalente a URL Base/receptor).

text

El atributo text se utiliza para sustituir la imagen por defecto que se le asigna a la representación gráfica de la etiqueta XML de contexto por un botón textual básico.
En ausencia de la definición de este atributo se asignará la correspondiente imagen por defecto a la representación gráfica de la etiqueta XML de contexto.

Valores:

- **Texto del botón:** cadena de texto.

image

El atributo image se utiliza para sustituir la imagen por defecto que se le asigna a la representación gráfica de la etiqueta XML de contexto por una imagen personalizada preferentemente en formato png o jpg.
En ausencia de la definición de este atributo se asignará la correspondiente imagen por defecto a la representación gráfica de la etiqueta XML de contexto.

Valores:

- **URL de la imagen:** URL de una imagen.

x

El atributo x se utiliza para asignar una posición absoluta o relativa en el eje X a la posición central de la representación gráfica de la etiqueta XML de contexto.
En ausencia de la definición de este atributo el navegador web aplicará un algoritmo de colocación para situar en pantalla la representación gráfica de la etiqueta XML de contexto.

Valores:

- **Posición en el eje X:** Un % relativo al tamaño de la pantalla, o una medida absoluta en pixeles (px). Por ejemplo: x="50%", x="200px"

y

El atributo y se utiliza para asignar una posición absoluta o relativa en el eje Y a la posición central de la representación gráfica de la etiqueta XML de contexto.
En ausencia de la definición de este atributo el navegador web aplicará un algoritmo de colocación para situar en pantalla la representación gráfica de la etiqueta XML de contexto.

Valores:

- **Posición en el eje Y:** Un % relativo al tamaño de la pantalla, o una medida absoluta en pixeles (px). Por ejemplo: y="10%", y=30px"

9.4.2 Etiquetas XML de contexto básicas

Posición

- magneticfield
- proximity
- orientation

Movimiento

- accelerometer
- grativy
- light
- temperature

Multimedia

- audioCapture
- camera

Localización

- location

9.4.2.1 Magneticfield <magneticfield/>

Introducción
Esta etiqueta se utiliza para obtener el valor actual registrado por el sensor de campo magnético.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con un sensor de campo magnético.

Retorno:
Cadena de texto que contiene el valor del campo magnético en los ejes X, Y y Z expresado en µT microteslas, los valores se separan utilizando el carácter ",".

Atributos

accuacy

Especifica el rango de variación mínimo para considerar dos mediciones como diferentes.
La comparación entre las mediciones obtenidas se realiza de manera individual en cada uno de los tres ejes. Es suficiente con que una de las tres mediciones haya experimentado una diferencia igual o mayor a la especificada en el atributo accuacy para que el conjunto del campo magnético sea considerado como una medición diferente.

Valores:
- **Número positivo**

Combinaciones de atributos:
- **Accuacy** con **sensitive**

sensitive

Condiciona el envío de información de contexto a que la última medición obtenida sea diferente a la anterior.

Valores:
- **true** en caso de que la medición obtenida sea igual a la última enviada, no la envía.
- **false** envía todas las mediciones

Combinaciones de atributos :
- **sensitive** con **timeInterval**

Tarea programada asociada: CI_magneticfield

URL de recepción de datos por defecto: URL base /magneticfield

9.4.2.2 Proximity <proximity/>

Introducción
Esta etiqueta se utiliza para obtener el valor actual registrado por el sensor de proximidad.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con un sensor de proximidad.

Retorno:
Valor numérico expresado en cm.

Atributos

accuacy

Especifica el rango de variación mínimo para considerar dos mediciones como diferentes.

Valores:
- **Número positivo**

Combinaciones de atributos:
- **Accuacy** con **sensitive**

sensitive

Condiciona el envío de información de contexto a que la última medición obtenida sea diferente a la anterior.

Valores:
- **true** en caso de que la medición obtenida sea igual a la última enviada, no la envía.
- **false** envía todas las mediciones

Combinaciones de atributos:
- **sensitive** con **timeInterval**

Tarea programada asociada: CI_proximity

URL de recepción de datos por defecto: URL base / proximity

9.4.2.3 Orientation <orientation/>

Introducción
Esta etiqueta se utiliza para obtener el valor actual registrado por el sensor de orientación.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con un sensor de orientación.

Retorno:
Cadena de texto que contiene el valor y la orientación del dispositivo en los ejes X, Y y Z expresados en grados, los valores se separan utilizando el carácter ",".

Atributos

accuacy

Especifica el rango de variación mínimo para considerar dos mediciones como diferentes.
La comparación entre las mediciones obtenidas se realiza de manera individual en cada uno de los tres ejes. Es suficiente con que una de las tres mediciones haya experimentado una diferencia igual o mayor a la especificada en el atributo accuacy para que el conjunto del campo magnético sea considerado como una medición diferente.

Valores:
- **Número positivo**

Combinaciones de atributos:
- **Accuacy** con **sensitive**

sensitive

Condiciona el envío de información de contexto a que la última medición obtenida sea diferente a la anterior.

Valores:
- **true** en caso de que la medición obtenida sea igual a la última enviada, no la envía.
- **false** envía todas las mediciones

Combinaciones de atributos:
- **sensitive** con **timeInterval**

Tarea programada asociada: CI_orientation

URL de recepción de datos por defecto: URL base / orientation

9.4.2.4 Accelerometer <accelerometer/>

Introducción
Esta etiqueta se utiliza para obtener el valor actual de la aceleración del dispositivo.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con un acelerómetro.

Retorno:
Cadena de texto que contiene el valor de la aceleración del dispositivo en los ejes X, Y y Z. Las mediciones en cada uno de los ejes pueden tomar valores positivos o negativos dependiendo de la orientación de la aceleración, los valores se separan utilizando el carácter ",".

Atributos

accuacy

Especifica el rango de variación mínimo para considerar dos mediciones como diferentes.
La comparación entre las mediciones obtenidas se realiza de manera individual en cada uno de los tres ejes. Es suficiente con que una de las tres mediciones haya experimentado una diferencia igual o mayor a la especificada en el atributo accuacy para que el conjunto del campo magnético sea considerado como una medición diferente.

Valores:
- **Número positivo,** este valor se tomara como absoluto en caso de que la aceleración en alguno de los dos tres ejes sea negativa.

Combinaciones de atributos:
- **Accuacy** con **sensitive**

sensitive

Condiciona el envío de información de contexto a que la última medición obtenida sea diferente a la anterior.

Valores:
- **true** en caso de que la medición obtenida sea igual a la última enviada, no la envía.
- **false** envía todas las mediciones

Combinaciones de atributos:
- **sensitive** con **timeInterval**

Tarea programada asociada: CI_accelerometer

URL de recepción de datos por defecto: URL base / accelerometer

9.4.2.5 Gravity <gravity/>

Introducción
Esta etiqueta se utiliza para obtener el valor actual registrado por el sensor de gravedad.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con un sensor de campo gravitatorio.

Retorno:
Número decimal, el valor de retorno esta expresado en m/s^2.

Atributos

accuacy

Especifica el rango de variación mínimo para considerar dos mediciones como diferentes.

Valores:
- **Número positivo**

Combinaciones de atributos:
- **Accuacy** con **sensitive**

sensitive

Condiciona el envío de información de contexto a que la última medición obtenida sea diferente a la anterior.

Valores:
- **true** en caso de que la medición obtenida sea igual a la última enviada, no la envía.
- **false** envía todas las mediciones

Combinaciones de atributos:
- **sensitive** con **timeInterval**

Tarea programada asociada: CI_gravity

URL de recepción de datos por defecto: URL base / Gravity

9.4.2.6 Light<light/>

Introducción
Esta etiqueta se utiliza para obtener el valor actual registrado por el sensor de luz.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con un sensor de luz.

Retorno:
Número decimal comprendido entre 0 y 1. 0 corresponde al nivel de luz más bajo y 1 al más alto.

Atributos

accuacy

Especifica el rango de variación mínimo para considerar dos mediciones como diferentes.

Valores:
- **Número positivo decimal menor que uno.**

Combinaciones de atributos:
- **Accuacy** con **sensitive**

sensitive

Condiciona el envío de información de contexto a que la última medición obtenida sea diferente a la anterior.

Valores:
- **true** en caso de que la medición obtenida sea igual a la última enviada, no la envía.
- **false** envía todas las mediciones

Combinaciones de atributos:
- **sensitive** con **timeInterval**

Tarea programada asociada: CI_light

URL de recepción de datos por defecto: URL base / light

9.4.2.7 Temperature <temperature/>

Introducción
Esta etiqueta se utiliza para obtener el valor actual registrado por el sensor de temperatura.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con un sensor de temperatura.

Retorno:
Número entero, el valor de retorno esta expresado grados centígrados.

Atributos

accuacy

Especifica el rango de variación mínimo para considerar dos mediciones como diferentes.

Valores:
- **Número positivo.**

Combinaciones de atributos:
- **Accuacy** con **sensitive**

sensitive

Condiciona el envío de información de contexto a que la última medición obtenida sea diferente a la anterior.

Valores:
- **true** en caso de que la medición obtenida sea igual a la última enviada, no la envía.
- **false** envía todas las mediciones

Combinaciones de atributos:
- **sensitive** con **timeInterval**

Tarea programada asociada: CI_temperature.

URL de recepción de datos por defecto: URL base / temperatura.

9.4.2.8 AudioCapture <audioCapture/>

Introducción
Esta etiqueta se utiliza para obtener una grabación de audio utilizando el micrófono del dispositivo.

Restricciones
El dispositivo debe disponer de espacio de almacenamiento libre para poder guardar temporalmente el fichero de audio.

Interacción de usuario

- Indica el momento de inicio de grabación.
- Indica el momento de finalización de la grabación.

Retorno:
Fichero de audio, preferiblemente estará en formato .wav u .ogg aunque puede variar dependiendo de las características técnicas del dispositivo.

Atributos

<u>format</u>

Define el formato del fichero de audio que se obtiene como respuesta. En ausencia de especificarse un formato soportado se utilizará el formato por defecto seleccionado por el sistema operativo del dispositivo.

Valores:
- **Cadena de texto**: .ogg, .wav, .mp3, .mp4, .3gp, .aiff, .caf.

<u>maxTime</u>

Número máximo de segundos de la grabación. En caso de que la grabación actual llegue al máximo de segundos permitido se cortará de manera automática.

Valores:
- **Segundos.**

Tarea programada asociada: CI_audioCapture.

URL de recepción de datos por defecto: URL base / audioCapture.

9.4.2.9 Camera <camera/>

Introducción
Esta etiqueta se utiliza para obtener una fotografía tomada por la cámara principal del dispositivo.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con una cámara de fotos.

El dispositivo debe disponer de espacio de almacenamiento libre para poder guardar temporalmente la imagen.

Interacción de usuario

- Indica el momento de tomar la fotografía.

Retorno:
Fichero de tipo imagen que preferiblemente estará en formato .jpg u .png aunque puede variar dependiendo de las características técnicas del dispositivo.

Atributos

<u>format</u>

Define el formato del fichero de tipo imagen que se obtiene como respuesta. En ausencia de especificarse un formato soportado se utilizará el formato por defecto seleccionado por el sistema operativo del dispositivo.

Valores:
- **png**
- **jgp**

Tarea programada asociada: CI_camera.

URL de recepción de datos por defecto: URL base / camera.

9.4.2.10 Location <location/>

Introducción
Esta etiqueta se utiliza para obtener la localización actual del dispositivo

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con alguno de los elementos hardware que pueden facilitar la obtención de la localización actual, como chip GPS o Wi-Fi.

Retorno:
Cadena de texto con las coordenadas geográficas longitud y latitud, separadas por el carácter ",".

Atributos

<u>source</u>

Determina el elemento que el dispositivo utilizará para obtener la localización actual.

Valores:
- **GPS:** utiliza el chip para la obtención de la ubicación GPS (Opción por defecto).
- **Network:** obtiene la ubicación de la red a la que esté conectado el dispositivo Wi-Fi, 3G, etc.

Combinaciones de atributos:
- **Accuacy** con **sensitive**

<u>accuacy</u>

Especifica el rango de variación mínimo para considerar dos ubicaciones como diferentes.

Valores:
- **Número positivo.**

Combinaciones de atributos:
- **Accuacy** con **sensitive**

<u>sensitive</u>

Condiciona el envío de la localización a que la última medición obtenida sea diferente a la anterior.

Valores:
- **true** en caso de que la medición obtenida sea igual a la última enviada, no la envía.
- **false** envía todas las mediciones

Combinaciones de atributos:

- **sensitive** con **timeInterval**

Tarea programada asociada: CI_location.

URL de recepción de datos por defecto: URL base / location.

9.4.3 Etiquetas XML de contexto complejas

Multimedia

- audioCaptureSpeech
- cameraBarCode
- cameraQRCode

9.4.3.1 AudioCaptureSpeech <audioCaptureSpeech/>

Introducción
Esta etiqueta utiliza el micrófono del dispositivo para capturar una conversación, esta grabación es analizada con un algoritmo para reconocer el discurso del usuario.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con un micrófono.

Interacción de usuario

- Indica el momento de inicio de grabación.
- Indica el momento de finalización de la grabación.

Retorno:
Cadena de texto con el discurso obtenido a partir del análisis de la grabación realizada.

Atributos

language

Idioma.

Valores:
- **Código ISO 639-1 del idioma:** EN, ES, FR, EL, IT, JA, etc.

Tarea programada asociada: CIC_audioCaptureSpeech.

URL de recepción de datos por defecto: URL base / audioCaptureSpeech

9.4.3.2 CameraBarCode <cameraBarCode/>

Introducción
Esta etiqueta se utiliza para decodificar un código de barras lineal empleando como escáner la cámara principal del dispositivo.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con una cámara de fotos.

Interacción de usuario
- Enfocar el código de barras lineal.

Retorno:
Cadena de texto con la información decodificada que contiene el código de barras lineal.

Tarea programada asociada: CIC_cameraBarCode.

URL de recepción de datos por defecto: URL base / cameraBarCode.

9.4.3.3 CameraQRCode <cameraQRCode/>

Introducción
Esta etiqueta se utiliza para decodificar un código QR empleando como escáner la cámara principal del dispositivo.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con una cámara de fotos.

Interacción de usuario
- Enfocar el código de barras QR.

Retorno:
Cadena de texto con la información decodificada que contiene el código QR.

Tarea programada asociada: CIC_cameraQRCode.

URL de recepción de datos por defecto: URL base / cameraQRCode.

9.4.4 Etiquetas XML de contexto basadas en acciones de comunicación

Comunicación

- bluetooth
- wi-fi
- nfc

9.4.4.1 Bluetooth <bluetooth/>

Introducción
Esta etiqueta se utiliza para definir acciones de comunicación que emplean la especificación Bluetooth para establecer vinculaciones y comunicaciones entre el dispositivo objeto y otros dispositivos cercanos.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con el hardware necesario.

La utilización de algunas configuraciones avanzadas como la comunicación a través de canales de comunicación inseguros podría verse limitada a los dispositivos que contasen elementos hardware y sistemas operativos móviles adecuados a las especificaciones de Bluetooth que contengan dichas funcionalidades.

Atributos

Action

Indica la acción de comunicación que debe ejecutarse en cada momento, existe dependencia entre algunas de las acciones.

Valores:

- **scan** escanea la red en busca de dispositivos bluetooth.
- **pairDevice** fuerza el emparejamiento con un dispositivo.
- **unPairDevice** fuerza el desemparejamiento de un dispositivo.
- **connectDevice** establece una conexión con un dispositivo. En caso de requerir el emparejado lo realiza de forma automática. En caso de no especificarse el dispositivo objeto se realiza un escaneado de forma automática.
- **disconnectDevice** corta la conexión con un dispositivo.
- **write** envía una cadena de datos (bytes).
- **listen** escucha hasta recibir una cadena de datos (bytes).

Dependencia entre acciones:

- **write** previamente debe ejecutarse la acción **connectDevice.**
- **listen** previamente debe ejecutarse la acción **connectDevice.**

Retorno distinto de confirmación genérica:

- **scan** lista de dispositivos y direcciones**.**
- **pairDevice** nombre y dirección del dispositivo emparejado
- **connectDevice** nombre y dirección del dispositivo conectado.
- **listen** datos recibidos (Bytes).

Combinaciones de atributos <bluetooth/>:

- **pairDevice** con **deviceName="nombre/dirección"**
- **unPairDevice** con **deviceName="nombre/dirección"**

- **connectDevice** con **deviceName= "nombre/dirección"**
- **connectDevice** con **UUID="32dígitos hexadecimal"**
- **connectDevice** opcionalmente con **pair="boolean"**
- **disconnectDevice** con **deviceName="nombre/dirección"**

- **write** con **data="bytes"**
- **write** opcionalmente con **deviceName="nombre/dirección"**
- **listen** opcionalmente con **deviceName="nombre/dirección"**

Esquema resumen de funcionamiento:

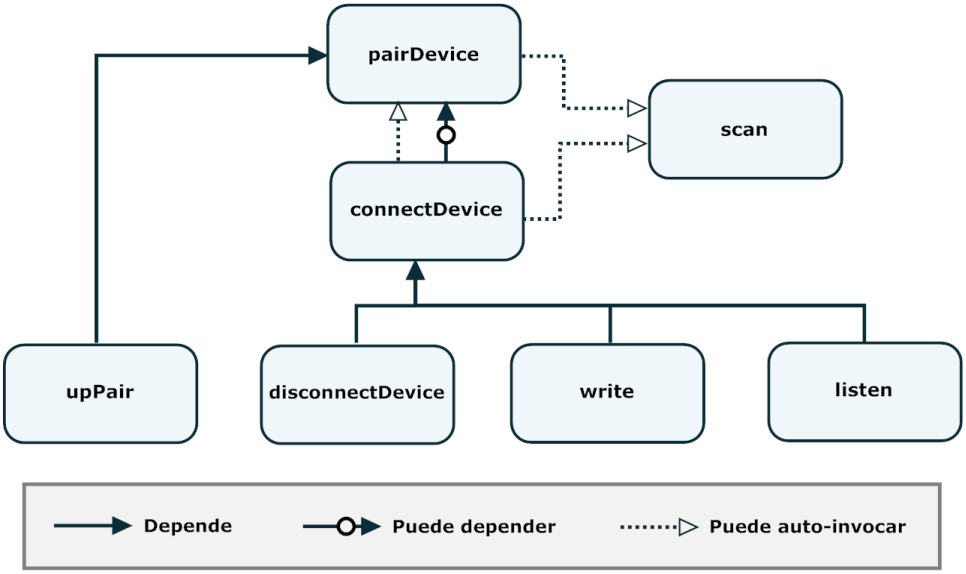

Figura 9.14 - Diagrama de dependencia entre acciones Bluetooth.

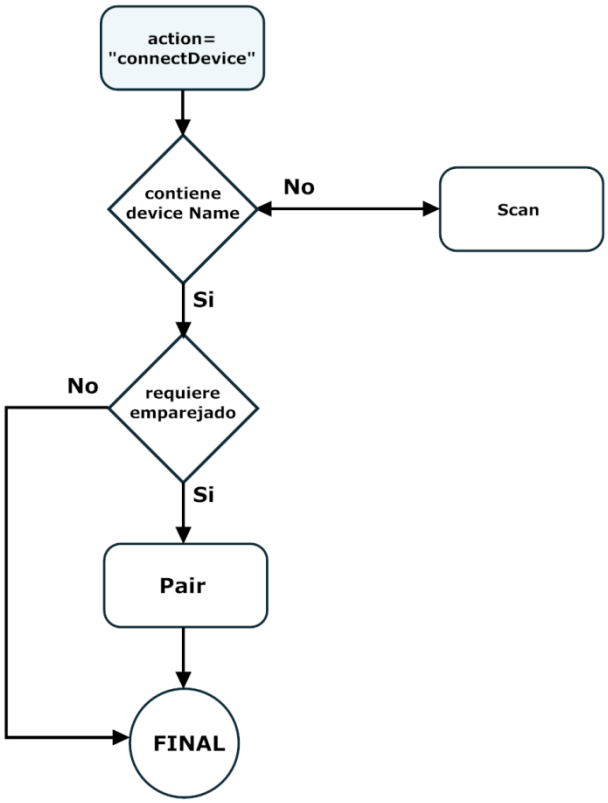

Figura 9.15 - Diagrama de flujo de la acción "connectDevice"

Pair

Este atributo puede asociarse opcionalmente a las etiquetas que contienen la acción de comunicación **connectDevice.** Por defecto, para establecer una conexión se requiere el emparejado de los dispositivos.
La especificación de Bluetooth contempla un tipo de conexión menos seguras que no requiere el emparejamiento previo entre los dispositivos para realizar la conexión y el posterior intercambio de datos.
La posibilidad de establecer conexiones sin emparejamiento previo depende de que este tipo de comunicación esté soportando tanto del cliente como del servidor.

Valores:

- **true** requiere emparejamiento (Opción por defecto).
- **false** conexión insegura que no requiere emparejamiento.

Combinaciones de atributos <bluetooth/>:

- **pair** puede combinarse únicamente con el atributo **action="connectDevice"-**

DeviceName

Este atributo se utiliza para identificar el dispositivo objeto de una determinada acción, los dispositivos puede identificarse de dos formas: (1) utilizando el nombre del dispositivo (2) por medio de una dirección bluetooth válida siguiendo el estándar IEEE 802 MAC-48, estas direcciones tienen seis grupos de dos caracteres hexadecimales separados por ":" o "-". Una dirección en el formato adecuado sería la siguiente 00:43:A8:23:10:F0.

Valores:

- **Nombre del dispositivo** cadena alfanumérica.
- **Dirección Bluetooth** una dirección bluetooth válida.

Combinaciones de atributos <bluetooth/>:

- **deviceName** puede combinarse con la etiqueta action cuando ésta tiene asociados los siguientes valores: **"pairDevice", "unPairDevice", "connectDevice" , "disconnectDevice" , "write"** y **"listen".**

  Si este atributo **action** no se combina con el uso del atributo **deviceName** el usuario deberá seleccionar el dispositivo destino de la acción de forma manual, eligiendo entre uno de los dispositivos que se mostrarán en la lista de conectados o descubiertos según el tipo de acción.
  Si la lista que va a mostrarse al usuario contiene un solo dispositivo la asignación se realizará de manera automática.

UUID

UUID es un identificador universal único (universally unique identifier). Entre otros muchos objetivos este atributo se utiliza para identificar un servicio que está alojado en un dispositivo al que se accede por medio de una conexión bluetooth.

Valores:

- **UUID** consiste en 32 dígitos hexadecimales mostrados en cinco grupos separados por guiones. Un valor del UUID seria por ejemplo: 5ee01872-4c9e-48d6-9f61-015ff816b278.

Combinaciones de atributos <bluetooth/>:

- **UUID** debe combinarse con el atributo **action="connectDevice";**

Data

Data define la cadena de bytes que será enviada al dispositivo destino de la comunicación.

Valores:

- **Data** cadena de bytes.

Combinaciones de atributos <bluetooth/>:

- **data** debe combinarse con el atributo **action="write";**

Tarea programada asociada a la etiqueta <bluetooth/> : CA_bluetooth

9.4.4.2 Wi-Fi <wi-fi/>

Introducción

Esta etiqueta se utiliza para definir acciones de comunicación que emplean la especificación Wi-Fi Ad-Hoc para establecer vinculaciones y comunicaciones entre el dispositivo objeto y otros dispositivos cercanos.

Las redes Ad Hoc habilitan la comunicación punto a punto entre dispositivos Wi-Fi, Cada uno de los dispositivos se comunica directamente con los demás a través de las señales de radio sin usar un punto de acceso. Solamente los dispositivos dentro de un rango de transmisión definido pueden comunicarse entre ellos.

Restricciones

Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con el hardware necesario.

La utilización de algunas configuraciones avanzadas podría verse limitada a los dispositivos que contasen con elementos hardware y sistemas operativos móviles adecuados a las especificaciones de Wi-Fi que contengan dichas funcionalidades.

Atributos

Action

Indica la acción de comunicación que debe ejecutarse en cada momento, existe dependencia entre algunas de las acciones.

Valores:

- **scan** escanea la red en busca de dispositivos Wi-Fi Ad-hoc.
- **connectDevice** establece una conexión con un dispositivo.
  En caso de no especificarse el dispositivo objeto se realiza un escaneado de forma automática.
- **write** envía una cadena de datos (bytes).
- **listen** escucha hasta recibir una cadena de datos (bytes).
- **disconnectDevice** fuerza el desemparejamiento de un dispositivo.

Dependencia entre acciones:

- **write** previamente debe ejecutarse la acción **connectDevice.**
- **listen** previamente debe ejecutarse la acción **connectDevice.**

Retorno distinto de confirmación genérica:

- **scan** lista de dispositivos y direcciones**.**
- **connectDevice** nombre y dirección del dispositivo conectado.
- **listen** datos recibidos (Bytes).

Combinaciones de atributos <wi-fi/>:

- **connectDevice** con **deviceName="nombre/MAC"**
- **disconnectDevice** con **deviceName="nombre"**

- **write** con **data="bytes"**
- **write** opcionalmente con **deviceName="nombre/MAC"**

- **listen** opcionalmente con **deviceName="nombre/MAC"**

Esquema resumen de funcionamiento:

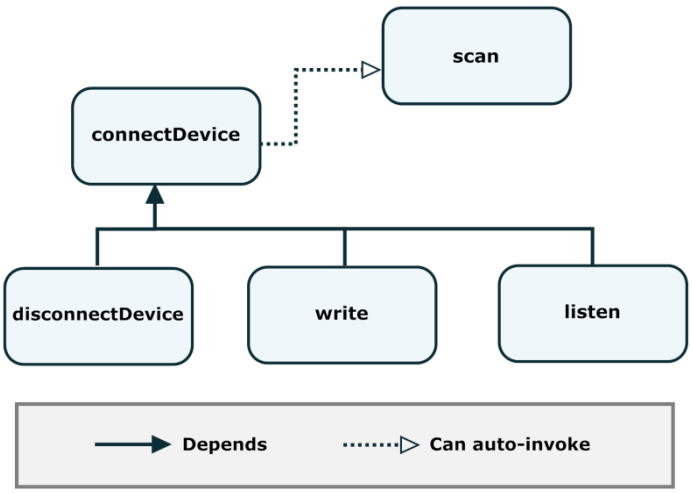

Figura 9.16 - Diagrama de dependencia entre acciones Wi-Fi.

DeviceName

Este atributo se utiliza para identificar el dispositivo objeto de una determinada acción, los dispositivos pueden identificarse de dos formas: (1) utilizando el nombre del dispositivo (2) por medio de un identificador único del interfaz de red válida siguiendo el estándar MAC-48. Estas direcciones tienen seis grupos de dos caracteres hexadecimales separados por ":" o "-". Una dirección en el formato adecuado sería la siguiente 00:43:A8:23:10:F0. Algunos sistemas admiten otras convenciones del MAC como la EUI-64, formado por tres grupos de cuatro dígitos hexadecimales separados por (.) como por ejemplo: 0123.4567.89ab

Valores:

- **Nombre del dispositivo** cadena alfanumérica.
- **Identificador único** MAC-48.

Combinaciones de atributos <wi-fi/>:

- **deviceName** puede combinarse con las acciones connectDevice , disconnectDevice, Write y Listen.

  Si este atributo **action** no se combina con el uso del atributo **deviceName** el usuario deberá seleccionar el dispositivo destino de la acción de forma manual, eligiendo entre uno de los dispositivos que se mostrarán en la lista de conectados o descubiertos según el tipo de acción.
  Si la lista que va a mostrarse al usuario contiene un solo dispositivo la asignación se realizará de manera automática.

Data

Data define la cadena de bytes que será enviara al dispositivo destino de la comunicación.

Valores:

- **Data** cadena de bytes.

Combinaciones de atributos <wi-fi/>:

- **data** debe combinarse con el atributo **action="write";**

Tarea programada asociada a la etiqueta <wi-fi/> : CA_wi-fi

9.4.4.3 NFC <nfc/>

Introducción
Esta etiqueta se utiliza para definir acciones de comunicación que emplean la especificación NFC para establecer vinculaciones y comunicaciones entre el dispositivo objeto y otros dispositivos cercanos.

Restricciones
Para poder utilizar esta etiqueta el dispositivo objeto debe estar equipado con el hardware necesario.

La utilización de algunas configuraciones avanzadas como la escritura de etiquetas NFC podría verse limitada a los dispositivos que contasen elementos hardware y sistemas operativos móviles adecuados a las especificaciones de NFC que contengan dichas funcionalidades.

Atributos

Action

Indica la acción de comunicación que debe ejecutarse en cada momento, existe dependencia entre algunas de las acciones.

Valores:
- **write** envía una cadena de datos (bytes).
- **read** escucha hasta recibir una cadena de datos (bytes).

Dependencias entre acciones:
- **write** previamente debe ejecutarse la acción **connectDevice.**
- **read** previamente debe ejecutarse la acción **connectDevice.**

Retorno distinto de confirmación genérica:
- **read** datos recibidos (Bytes).

Combinaciones de atributos <nfc/>:
- **write** con **data="bytes"**

Data

Define la cadena de bytes que será enviada al dispositivo destino de la comunicación.

Valores:
- **Data** cadena de bytes.

Combinaciones de atributos <nfc/>:
- **data** debe combinarse con el atributo **action="write";**

Tarea programada asociada a la etiqueta <nfc/> : CA_nfc

9.5 NAVEGADOR WEB SENSIBLE AL CONTEXTO

9.5.1 Introducción

El navegador web sensible al contexto es una de las piezas claves de esta arquitectura. La especificación del navegador web sensible al contexto está formada por una arquitectura y una serie de requerimientos funcionales que deben ser aplicados con el fin de realizar una implementación válida de este navegador.

Técnicamente el navegador web sensible al contexto puede ser implementado en la mayoría de plataformas móviles actuales. La funcionalidad del navegador web sensible al contexto puede verse condicionada por las únicas restricciones técnicas de la plataforma móvil o dispositivo destino. El dispositivo móvil destino debe estar equipado con los elementos hardware que permiten capturar la información de contexto y ejecutar las acciones de comunicación. Por otro lado la plataforma móvil del dispositivo debe facilitar mecanismos de gestión para estos mismos elementos hardware.

Las funcionalidades que se describen en posteriores secciones no están ligadas a ninguna plataforma móvil concreta ni a ningún lenguaje de programación. Las funciones principales a las que el navegador web sensible al contexto da soporte son las siguientes:

- Interpretar de la misma forma que los navegadores web comunes (Internet Explorer, Chrome, Safari, etc) las páginas web basadas tanto en lenguajes web estandarizados como en otras tecnologías web de amplia difusión y aceptación.

- Debe notificar a los usuarios los requerimientos provenientes de las etiquetas XML de contexto contenidas en la página web. El navegador web sensible al contexto debe dar la oportunidad al usuario de aceptar o rechazar cada uno de los requerimientos procedentes de las etiquetas XML de contexto. El navegador web sensible al contexto debe comunicarle al usuario en todo momento el estado actual de cada una de las peticiones de información de contexto que contiene la página web, es decir si se ha iniciado el proceso, si se está ejecutando, si el proceso finalizo correctamente, si se ha producido algún error, etc.

- Cuando el usuario da su permiso para que se proceda a satisfacer el requerimiento expresado por la etiqueta XML de contexto el navegador web sensible al contexto ejecutará la tarea programada de contexto correspondiente. Cada una de estas tareas cuenta con un objetivo específico, todas ellas implican la gestión de alguno de los elementos hardware del dispositivo para capturar información de contexto o realizar algún tipo de acción de comunicación. El resultado de la ejecución de la tarea programada de contexto correspondiente es el que se envía de vuelta a la aplicación web, o al destino especificado en la propia etiqueta. Este envío se realiza mediante una petición HTTP POST que cuenta con las cabeceras correspondientes en cada caso, como por ejemplo el DSK, ID, group o member.

9.5.2 Arquitectura

La arquitectura propuesta para el navegador web sensible al contexto está compuesta por tres capas (Fig.9.17).

- **Navegador web**. La primera capa contiene un módulo de funcionalidad muy similar a los navegadores web clásicos.

- **Gestor de etiquetas XML de contexto**. La segunda capa se encarga de gestionar todos los aspectos relativos al procesamiento y representación de las etiquetas XML de contexto.

- **Gestor de tareas programadas de contexto**. La tercera capa se encarga de almacenar, seleccionar y ejecutar las tareas programadas de contexto. Esta capa también se encarga de gestionar todos los aspectos relativos al etiquetado y envío de las respuestas generadas por las tareas programadas de contexto.

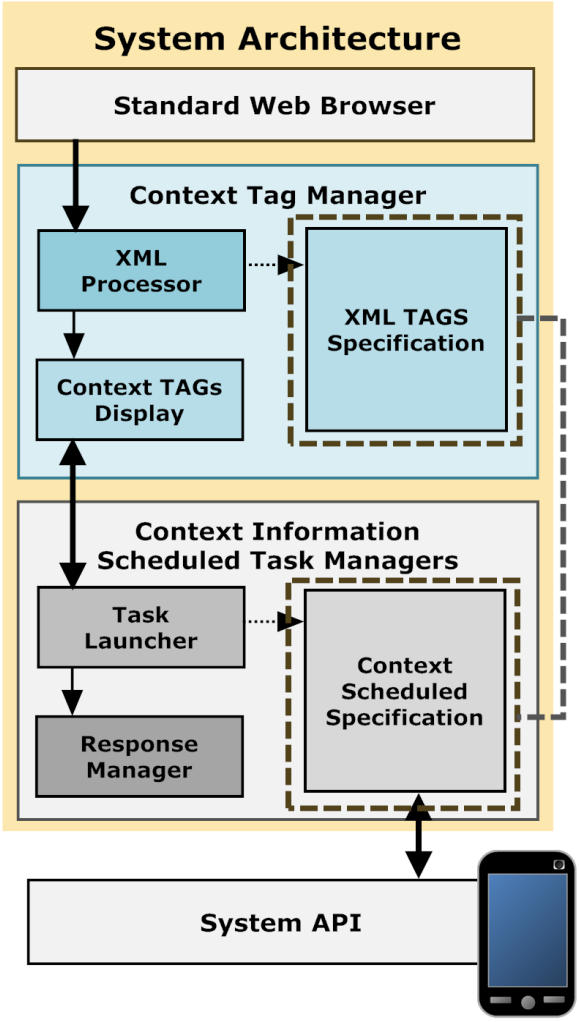

Figura 9.17 - Arquitectura del navegador web sensible al contexto.

9.5.2.1 Capa 1: Navegador Web

Los navegadores web convencionales son aplicaciones que operan a través de Internet accediendo a ficheros "páginas web" que contienen información en determinados formatos estandarizados, como HTML, css y Javascrip. Esta información es interpretada por el navegador y presentada a los usuarios, para que estos sean capaces de interactuar con el contenido de la página. El objetivo de esta capa es permitir al navegador web sensible al contexto interpretar los lenguajes web estándares soportados por los navegadores web convencionales, como por ejemplo: Internet Explorer, Chrome, o Safari.

Esta capa será la encargada de representar exclusivamente aquellos datos que tengan que ver con los lenguajes convencionales de la web y no tendrá ninguna relación con la representación de las etiquetas XML de contexto. Los aspectos relativos a la representación de las etiquetas XML de contexto se encapsulan en el **Gestor de etiquetas XML de contexto**, este componente se encarga de representar gráficamente las etiquetas XML de contexto en una capa propia, independiente del resto de los elementos de la web.

Al igual que los navegadores tradicionales esta capa se encarga de realizar las peticiones HTTP al servidor para solicitar páginas, enviar información, etc. Todas las peticiones HTTP y HTTPs procedentes del navegador web sensible al contexto contendrán una cabecera **DSK** "Clave de sesión de dispositivo". Esta clave debe generarse cada vez que se arranca el navegador web sensible al contexto. El DSK se forma a partir de la concatenación de dos elementos:

- **Identificador de dispositivo:** (3-6) Dígitos aleatorios + IMEI del dispositivo + (3-6) Dígitos aleatorios. Esta cadena de texto se codificará utilizando el algoritmo MD5 Hash.

- **Identificador de sesión del navegador:** Un número aleatorio de 9 caracteres. Esta cadena de texto se codificará utilizando el algoritmo MD5 Hash.

El mecanismo de codificación MD5 hash será sustituido en un futuro por un algoritmo que ofrezca un nivel de seguridad más alto.

Las plataformas móviles más populares en la actualidad: Android, Apple iOS, Windows Mobile, Symbian iOs y Blackberry incluyen componentes predefinidos que encapsula la funcionalidad básica del navegador web del dispositivo, este componente puede resultar útil para el desarrollo de esta capa.

9.5.2.2 Capa 2: Gestor de etiquetas XML de contexto

La segunda capa encapsula la responsabilidad de todas las tareas relacionadas con la gestión de las etiquetas XML de contexto. El gestor de etiquetas XML de contexto está formado por los siguientes módulos:

- **Analizador XML**, un componente analizador que se encarga de procesar el código de la página web en busca de alguna etiqueta XML que corresponda con las etiquetas XML de contexto que el dispositivo está habilitado para procesar. El analizador tiene acceso a una lista en la que están definidas todas las etiquetas XML de contexto, junto a cada una de ellas se encuentra un flag que indica si la plataforma soporta la funcionalidad requerida (Fig.9.18). Adicionalmente, aunque la etiqueta XML de contexto este soportada por la plataforma móvil puede que el dispositivo no esté equipado a nivel de hardware para satisfacer el requisito.

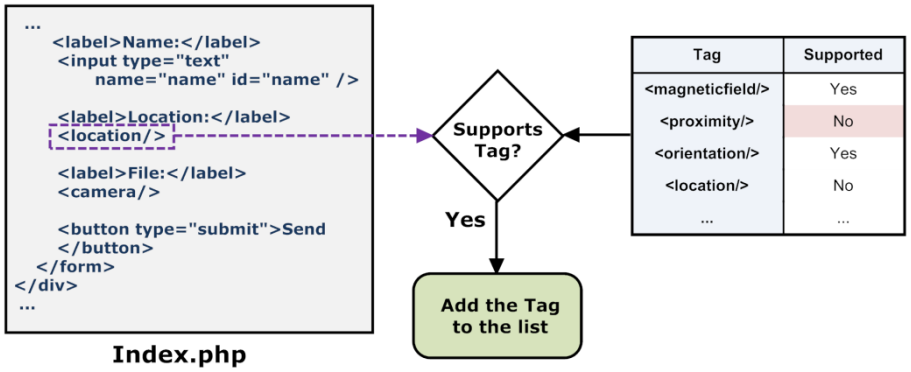

Figura 9.18 – El analizador XML comprueba si las etiquetas XML de contexto esta soportada por la plataforma.

Una vez han sido identificadas todas las etiquetas XML de contexto contenidas en la página web se almacenan en una lista. Cada una de las etiquetas XML de contexto se guarda junto a sus atributos, la lista completa se remite al siguiente módulo del gestor de etiquetas XML de contexto, el **visualizador de etiquetas XML de contexto**. En caso de que la lista contenga varias etiquetas XML de contexto pertenecientes al mismo grupo se incluirá un atributo auto-generado "members" en cada una de ellas, el cual indica el número de miembros que componen el grupo.

- **Visualizador de etiquetas XML de contexto**. Este componente se encarga de procesar la lista de etiquetas XML de contexto recibida. Utiliza el nombre de la etiqueta (<location>, <camera>, etc) y el valor de los atributos de representación gráfica (en caso de que hayan sido definidos: text, image, x e y) para generar su representación gráfica concreta. Esta representación corresponde a un botón que se coloca en una capa independiente a la encargada de la visualización del código web (Fig.9.19 y Fig.9.20). El módulo visualizador genera un botón por cada una de las etiquetas XML de contexto contenidas en la lista. Por lo tanto, cada uno de los

botones que se muestran por pantalla estará asociado a una etiqueta XML de contexto concreta.

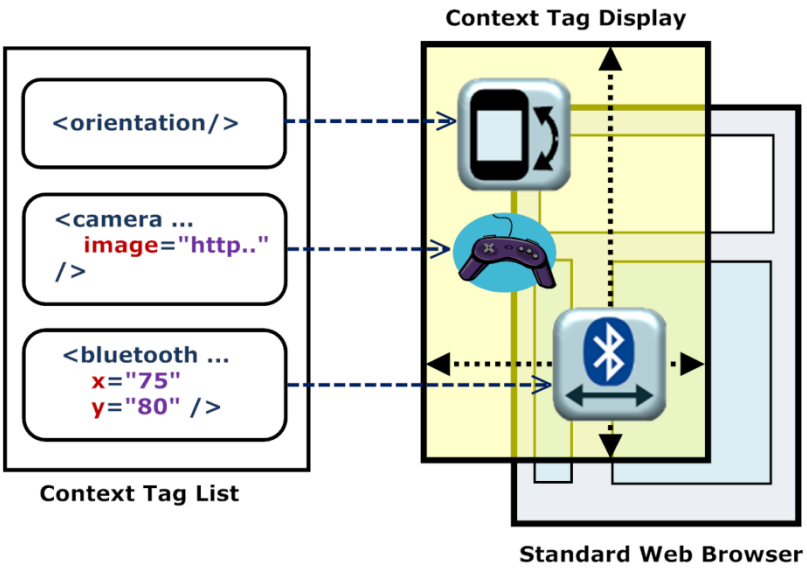

Figura 9.19 - Utiliza el nombre de la etiqueta y el valor de algunos de sus atributos para generar la representación gráfica de la etiqueta XML de contexto. Esta representación se añade a una capa independiente que se superpone a la vista encargada de mostrar los controles web.

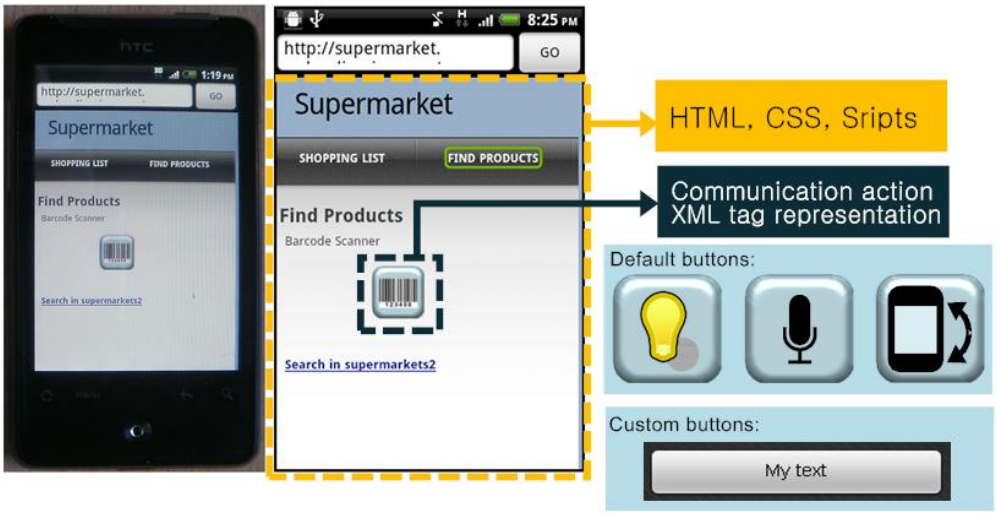

Figura 9.20 - En la captura se muestran los formatos de botones que se asocian a las etiquetas XML de contexto.

El visualizador de etiquetas XML de contexto tiene un canal de comunicación abierto con la capa inferior, el **gestor de tareas programadas de contexto**. Este canal se utiliza para recibir notificaciones que impliquen realizar cambios en la representación gráfica de una etiqueta XML de contexto. Cuando el visualizador de etiquetas XML de contexto recibe una notificación asociada a alguna de las etiquetas XML de contexto, añade a su representación gráfica un icono, este icono ilustra el estado actual de la tarea programada de contexto.

Tipos de notificaciones:

(1) Preparada
(2) En tratamiento
(3) Completada
(4) Error

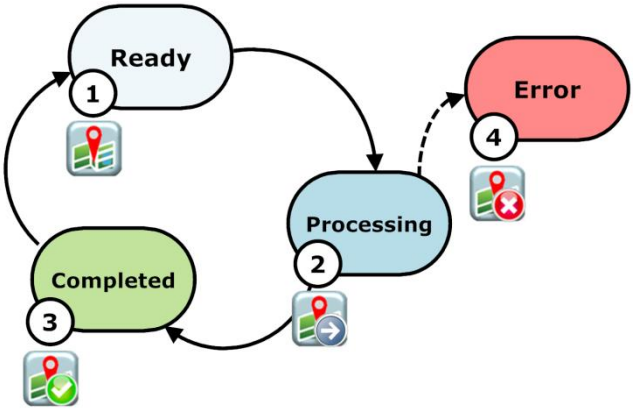

Figura 9.21 – El visualizador de etiquetas XML de contexto modifica la representación gráfica de la tarea en función de las notificaciones recibidas por parte del gestor de tareas programadas de contexto.

Al mantener pulsado uno de los botones asociados a las etiquetas XML de contexto el navegador web sensible al contexto debe ~~de~~ mostrar el contenido de la etiqueta. De esta forma el usuario puede conocer los detalles de la etiqueta asociada antes de aceptar el requerimiento.

Cuando el usuario pulsa sobre uno de los botones asociados a la etiqueta XML de contexto el gestor de etiquetas XML de contexto envía la etiqueta y el conjunto de sus atributos al **gestor de tareas programadas de contexto**, concretamente al componente **lanzador de tareas**.

9.5.2.3 Capa 3: Gestor de tareas programadas de contexto

Esta capa encapsula la funcionalidad relativa a la gestión de las tareas programadas de contexto y los módulos que contienen la implementación de cada una de estas tareas.

Las tareas programadas de contexto son módulos independientes de código ejecutable con una funcionalidad concreta. Cada tarea programada de contexto tiene una funcionalidad bien definida, por norma general ésta requiere de la utilización del API del sistema para permitir la gestión de los elementos hardware del dispositivo. La tarea programa de contexto utilizará los elementos hardware del dispositivo para capturar un tipo concreto de información de contexto o realizar una acción de comunicación. Algunas tareas programadas de contexto se limitan a capturar la información de contexto, otras aplican una serie de algoritmos para procesar la información capturada. Por ejemplo, en el caso de los códigos QR los algoritmos contenidos en la tarea analizarán una imagen con el fin de decodificar la información.

El gestor de tareas programadas de contexto está formado por los siguientes módulos:

- **Implementación de las tareas programadas de contexto**.

- **Lanzador de tareas**: Este módulo recibe las llamadas procedentes de la capa superior, cada una de estas llamadas contienen la etiqueta XML de contexto junto a sus atributos asociados. El lanzador busca en la lista de tareas programadas de contexto la tarea asociada a la etiqueta recibida, una vez encontrada la ejecuta enviando como parámetros de entrada los atributos definidos en la etiqueta XML de contexto.

  La funcionalidad de las tareas programadas de contexto estará condicionada por los parámetros de entrada específicos que admita cada una de ellas. Por ejemplo, la tarea *CI_Location* permite definir la fuente de la ubicación del dispositivo, ésta puede ser el chip GPS o la red Wi-Fi, este parámetro puede ser definido como un atributo en la etiqueta XML de contexto *<location/>*. En caso de que alguno de los atributos necesarios no esté definido en la etiqueta la tarea programada de contexto establecerá un valor por defecto al parámetro.

  Cada tipo de etiqueta XML de contexto está asociado a una tarea programada de contexto específica. Cuando se lanza una tarea programada de contexto esta se encuentra en estado: (1) Preparado, dependiendo del flujo de ejecución la tarea puede cambiar su estado a (2) En tratamiento (3) Completada (4) Error, cada vez que se produce un cambio en el estado de la tarea, este notifica al **Visualizador de etiquetas XML de contexto** . Cuando las tareas programadas de contexto finalizan una acción remiten la información capturada o la respuesta a la acción de comunicación al módulo **gestor de respuestas.** Junto a la respuesta se adjuntan los atributos "id", "group", "members" (auto-generado), en caso de que estos hayan sido definidos en la etiqueta XML de contexto que inicio la ejecución de la tarea.

- **Gestor de respuestas**: Este módulo se encarga de almacenar y enviar las respuestas de las tareas programadas de contexto. Cuando se cumplen las condiciones para realizar el envío, el módulo gestor de respuestas encapsula la

información en una petición POST que envía a la URL que se haya especificado en el atributo "receiver" de la etiqueta XML de contexto. Por norma general este módulo realiza los envíos de datos de manera automática al recibir las respuestas de la tarea programada de contexto. Este proceso se cumple a no ser que la respuesta contenga los atributos "group" y "members"; en este caso el gestor de respuestas esperará a recibir la información de todos los miembros del grupo antes de proceder al envío de la información de forma conjunta.

La estructura de la petición POST enviada es la siguiente:

- Cabecera:
  - **DSK**: Clave de sesión de dispositivo.
  - *** id**: identificador de la etiqueta XML de contexto que inició la tarea programada de contexto.
  - *** group**: grupo de la etiqueta XML de contexto que inició la tarea programada de contexto.
  - *** members**: número de etiquetas XML de contexto pertenecientes al grupo.

  ***** Estas cabeceras solo serán añadidas a la petición si han sido definidas como atributos en la etiqueta XML de contexto que inicia la tarea programada de contexto.
- Cuerpo:
  - **Dato/s**: cadena de texto o archivo.

  * En caso de tratarse de un grupo de etiquetas XML de contexto los datos se separarán como si se tratase de inputs de un formulario HTML. El identificador de cada dato será la id (id : dato).

9.5.2.3.1 Tareas programadas de contexto

9.5.2.3.1.1 Parámetros de entrada

La mayor parte de las tareas programadas de contexto admiten parámetros de entrada de manera opcional. Estos parámetros deben ser definidos en los atributos de las etiquetas XML de contexto. Las tareas de contexto definen valores por defecto para la mayoría de sus parámetros, por lo que no se requiere que las definiciones de las etiquetas XML de contexto contengan todos los atributos disponibles. Algunos de estos parámetros son útiles para configurar el comportamiento de la propia tarea, ya que definen aspectos importantes que deben ser tenidos en cuenta durante el proceso de captura de información de contexto o la ejecución de la acción de comunicación.

Algunos de los atributos que pueden ser definidos de manera opcional en las etiquetas XML de contexto son de carácter general y no tienen que ver directamente con el propio proceso de captura de la información de contexto o acción de comunicación. Estos atributos permiten definir aspectos como la periodicidad en la ejecución de la tarea programada de contexto. Por ejemplo, el atributo "timeinterval" se usaría en el caso de que la tarea programada de contexto asociada a una etiqueta tuviese que ser ejecutada en forma de bucle cada cierto intervalo de tiempo.

9.5.2.3.1.2 Interacción del usuario

La mayoría de las tareas programadas de contexto se ejecutan de forma autónoma sin requerir ningún tipo de interacción por parte del usuario, como por ejemplo obtener el valor del sensor de aceleración. En cambio otras tareas sí que requieren algún tipo de interacción por parte del usuario; realizar una captura de audio puede requerir que el usuario señale el momento inicial y final de la grabación.

La mayoría de tareas programadas de contexto que requieren la interacción de usuario necesitan que este señale momentos puntuales; las señales se utilizan como eventos para lanzar la ejecución de ciertas acciones contenidas en la tarea programada de contexto. Por ejemplo la tarea correspondiente a la etiqueta XML de contexto <camera/> necesita que el usuario enfoque un objetivo y pulse un botón.

9.5.2.3.1.3 Estado de las tareas

Todas las tareas programadas de contexto tienen cuatro estados comunes (1) Preparada, (2) En tratamiento (3) Completada (4) Error. Estos estados son siempre los mismos, independientemente de la funcionalidad de la tarea programada de contexto. Los estados se utilizan para notificar al usuario la fase en la que se encuentra la tarea.

(1) Preparada: cuando una tarea programada de contexto esta lista para ser ejecutada su estado es "preparada"; este estado se manifestará siempre que el teléfono disponga de los elementos hardware necesarios para poder satisfacer los requerimientos señalados.

(2) En tratamiento: este estado indica que la tarea programada de contexto ha empezado a ejecutarse pero que aún no ha finalizado.

(3) Completada: una vez la ejecución de la tarea programa de contexto haya finalizado satisfactoriamente ésta pasará al estado "Completada". En la mayoría de casos este será el estado final de la tarea, con la excepción de aquellas ejecuciones de tareas programadas de contexto que hayan sido lanzadas como bucles. Estas tareas se ejecutarán periódicamente cada un cierto intervalo de tiempo; en estos casos se producirá una transición del estado "Completada" a "en tratamiento", cada vez que se requiera una nueva interacción.

(4) Error: en el caso de que durante la ejecución de una tarea programada de contexto se produzca un error irrecuperable que no permita completar la tarea el estado pasara a ser "Error".

9.6 ESPECIFICACIÓN: TAREAS PROGRAMADAS DE CONTEXTO

9.6.1 CI_magneticfield

Parámetros de entrada específicos:

- **sensitive:** condiciona el envío del campo magnético a que la última medición obtenida sea diferente a la anterior, toma como valor un booleano.
- **accuacy:** especifica el rango de variación mínimo para considerar dos campos magnéticos como diferentes, toma como valor un número decimal positivo.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utiliza el API específico de la plataforma que gestiona el sensor de campo magnético del dispositivo. La tarea ejecutará la acción que permite obtener la medición actual del sensor de campo magnético, a partir de este momento existen dos alternativas posibles:

a. Si el parámetro ***sensitive*** tiene valor true: La primera vez que se obtiene la medición actual, ésta será almacenada en el *registro de último nivel de campo magnético*, el valor de este registro se utilizará como elemento de comparación en futuras invocaciones de la tarea.
   Si el *registro de último nivel de campo magnético* esta vacio , o la diferencia de éste y la medición actual es mayor que la establecida en el parámetro ***accuacy,*** se dará por finalizada la tarea retornando la aceleración actual y almacenando la nueva medición en el *registro de último nivel de campo magnético*. Si la diferencia entre las dos mediciones es menor que la establecida en el parámetro ***accuacy*** la tarea finalizará retornando el valor 0 (sin variación). Para calcular la diferencia entre dos campos magnéticos se compararán los valores de sus tres coordenadas X, Y y Z de manera independiente (x con x, y con y, z con z), si la diferencia absoluta en cualquiera de las tres coordenadas es mayor a la especificada en el parámetro ***accuacy*** se considerarán campos magnéticos distintos.

b. Si el parámetro ***sensitive*** tiene valor false o no tiene valor: se dará por finalizada la tarea retornando el campo magnético actual.

   - Retorno y cambio de estado:
     - **Retorno:** *[Campo magnético: X, Y, y Z . Campo magnético sin cambios: "equal"]*

Estado: 3 (Completada)
Acción completada correctamente.

- **Ret:** 0, **Estado:** 4 (Error)
  Se ha producido un error.

9.6.2 CI_proximity

Parámetros de entrada específicos:

- **sensitive:** condiciona el envío del nivel de proximidad a que la última medición obtenida sea diferente a la anterior, toma como valor un booleano.
- **accuacy:** especifica el rango de variación mínimo para considerar dos mediciones como diferentes, toma como valor un número entero positivo.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utiliza el API específico de la plataforma que gestiona el sensor de proximidad del dispositivo. La tarea ejecutará la acción que permite obtener la medición actual del sensor de proximidad, a partir de este momento existen dos alternativas posibles:

a. Si el parámetro ***sensitive*** tiene valor true: La primera vez que se obtiene la medición actual, ésta será almacenada en el *registro de último nivel de proximidad*, el valor de este registro se utilizará como elemento de comparación en futuras invocaciones de la tarea.
   Si el *registro de último nivel de proximidad* esta vacio o la diferencia de éste y la medición actual es mayor que la establecida en el parámetro ***accuacy*** se dará por finalizada la tarea retornando el nivel de proximidad actual y almacenando la nueva medición en el *registro de último nivel de proximidad*. Si la diferencia entre las dos mediciones es menor que la establecida en el parámetro ***accuacy*** la tarea finalizará retornando el valor 0 (sin variación). Para calcular la diferencia entre dos mediciones se compararán sus valores, si la diferencia absoluta es mayor a la especificada en el parámetro ***accuacy*** se considerarán mediciones distintas.

b. Si el parámetro ***sensitive*** tiene valor false o no tiene valor: se dará por finalizada la tarea retornando el nivel de proximidad actual.

   - Retorno y cambio de estado:
     - **Retorno:** *[Nivel de proximidad (cm) .Nivel de proximidad sin cambios: "equal"]*
       Estado: 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

9.6.3 CI_orientation

Parámetros de entrada específicos:

- **sensitive:** condiciona el envío de la orientación a que la última medición obtenida sea diferente a la anterior, toma como valor un booleano.
- **accuacy:** especifica el rango de variación mínimo para considerar dos orientaciones como diferentes, toma como valor un número decimal positivo.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utiliza el API específico de la plataforma que gestiona el sensor de orientación del dispositivo. La tarea ejecutará la acción que permite obtener la medición actual del sensor de orientación, a partir de este momento existen dos alternativas posibles:

a. Si el parámetro ***sensitive*** tiene valor true: La primera vez que se obtiene la medición actual, ésta será almacenada en el *registro de último nivel de orientación*, el valor de este registro se utilizará como elemento de comparación en futuras invocaciones de la tarea.
   Si el *registro de último nivel de orientación* esta vacio o la diferencia de éste y la medición actual es mayor que la establecida en el parámetro ***accuacy*** se dará por finalizada la tarea retornando la aceleración actual y almacenando la nueva medición en el *registro de último nivel de orientación*. Si la diferencia entre las dos mediciones es menor que la establecida en el parámetro ***accuacy*** la tarea finalizará retornando el valor 0 (sin variación). Para calcular la diferencia entre dos orientaciones se compararán los valores de sus tres coordenadas X, Y y Z de manera independiente (x con x, y con y, z con z), si la diferencia absoluta en cualquiera de las tres coordenadas es mayor a la especificada en el parámetro ***accuacy*** se considerarán aceleraciones distintas.

b. Si el parámetro ***sensitive*** tiene valor false o no tiene valor: se dará por finalizada la tarea retornando el nivel de orientación actual.

   - Retorno y cambio de estado:
     - **Retorno:** *[Orientación: X, Y, y Z . Orientación sin cambios: "equal"]*
       Estado: 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

9.6.4 CI_accelerometer

Parámetros de entrada específicos:

- **sensitive:** condiciona el envío de la aceleración a que la última medición obtenida sea diferente a la anterior, toma como valor un booleano.
- **accuacy:** especifica el rango de variación mínimo para considerar dos aceleraciones como diferentes, toma como valor un número decimal positivo.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utiliza el API específico de la plataforma que gestiona el sensor de aceleración del dispositivo. La tarea ejecutará la acción que permite obtener la medición actual del sensor de aceleración, a partir de este momento existen dos alternativas posibles:

a. Si el parámetro ***sensitive*** tiene valor true: La primera vez que se obtiene la medición actual, ésta será almacenada en el *registro de último nivel de aceleración*, el valor de este registro se utilizará como elemento de comparación en futuras invocaciones de la tarea.
   Si el *registro de último nivel de aceleración* esta vacio o la diferencia de éste y la medición actual es mayor que la establecida en el parámetro ***accuacy*** se dará por finalizada la tarea retornando la aceleración actual y almacenando la nueva medición en el *registro de último nivel de aceleración*. Si la diferencia entre las dos mediciones es menor que la establecida en el parámetro ***accuacy*** la tarea finalizará retornando el valor 0 (sin variación). Para calcular la diferencia entre dos aceleraciones se compararán los valores de sus tres coordenadas X, Y y Z de manera independiente (x con x, y con y, z con z), si la diferencia absoluta en cualquiera de las tres coordenadas es mayor a la especificada en el parámetro ***accuacy*** se considerarán aceleraciones distintas.

b. Si el parámetro ***sensitive*** tiene valor false o no tiene valor: se dará por finalizada la tarea retornando el nivel de aceleración actual.

   - Retorno y cambio de estado:
     - **Retorno:** *[Aceleración: X, Y, y Z . Aceleración: "equal"]*
       Estado: 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

9.6.5 CI_gravity

Parámetros de entrada específicos:

- **sensitive:** condiciona el envío del nivel de gravedad a que la última medición obtenida sea diferente a la anterior, toma como valor un booleano.
- **accuacy:** especifica el rango de variación mínimo para considerar dos mediciones como diferentes, toma como valor un número decimal positivo.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utiliza el API específico de la plataforma que gestiona el sensor de gravedad del dispositivo. La tarea ejecutará la acción que permite obtener la medición actual del sensor de gravedad, a partir de este momento existen dos alternativas posibles:

a. Si el parámetro ***sensitive*** tiene valor true: La primera vez que se obtiene la medición actual, ésta será almacenada en el *registro de último nivel de gravedad*, el valor de este registro se utilizará como elemento de comparación en futuras invocaciones de la tarea.
   Si el *registro de último nivel de gravedad* esta vacio o la diferencia de éste y la medición actual es mayor que la establecida en el parámetro ***accuacy*** se dará por finalizada la tarea retornando el nivel de gravedad actual y almacenando la nueva medición en el *registro de último nivel de gravedad*. Si la diferencia entre las dos mediciones es menor que la establecida en el parámetro ***accuacy*** la tarea finalizará retornando el valor 0 (sin variación). Para calcular la diferencia entre dos mediciones se compararán sus valores, si la diferencia absoluta es mayor a la especificada en el parámetro ***accuacy*** se considerarán mediciones distintas.

b. Si el parámetro ***sensitive*** tiene valor false o no tiene valor: se dará por finalizada la tarea retornando el nivel de gravedad actual.

   - Retorno y cambio de estado:
     - **Retorno:** *[Nivel de gravedad: m/s^2 . Nivel de gravedad: "equal"]*
       Estado: 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

9.6.6 CI_light

Parámetros de entrada específicos:

- **sensitive:** condiciona el envío del nivel de luz a que la última medición obtenida sea diferente a la anterior, toma como valor un booleano.
- **accuacy:** especifica el rango de variación mínimo para considerar dos mediciones como diferentes, toma como valor un número decimal positivo.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utiliza el API específico de la plataforma que gestiona el sensor de luz del dispositivo. La tarea ejecutará la acción que permite obtener la medición actual del sensor de luz, a partir de este momento existen dos alternativas posibles:

a. Si el parámetro ***sensitive*** tiene valor true: La primera vez que se obtiene la medición actual, ésta será almacenada en el *registro de último nivel de luz*, el valor de este registro se utilizará como elemento de comparación en futuras invocaciones de la tarea.
   Si el *registro de último nivel de luz* esta vacio o la diferencia de éste y la medición actual es mayor que la establecida en el parámetro ***accuacy*** se dará por finalizada la tarea retornando el nivel de luz actual y almacenando la nueva medición en el *registro de último nivel de luz*. Si la diferencia entre las dos mediciones es menor que la establecida en el parámetro ***accuacy*** la tarea finalizará retornando el valor 0 (sin variación). Para calcular la diferencia entre dos mediciones se compararán sus valores, si la diferencia absoluta es mayor a la especificada en el parámetro ***accuacy*** se considerarán mediciones distintas.

b. Si el parámetro ***sensitive*** tiene valor false o no tiene valor: se dará por finalizada la tarea retornando el nivel de luz actual.

   - Retorno y cambio de estado:
     - **Retorno:** *[Nivel de luz (0-0.1) .Nivel de luz sin cambios: "equal"]*
       Estado: 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

9.6.7 CI_temperature

Parámetros de entrada específicos:

- **sensitive:** condiciona el envío de la temperatura a que la última medición obtenida sea diferente a la anterior, toma como valor un booleano.
- **accuacy:** especifica el rango de variación mínimo para considerar dos mediciones como diferentes, toma como valor un número entero positivo.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utiliza el API específico de la plataforma que gestiona el sensor de temperatura del dispositivo. La tarea ejecutará la acción que permite obtener la medición actual del sensor de temperatura, a partir de este momento existen dos alternativas posibles:

a. Si el parámetro ***sensitive*** tiene valor true: La primera vez que se obtiene la medición actual, ésta será almacenada en el *registro de última temperatura*, el valor de este registro se utilizará como elemento de comparación en futuras invocaciones de la tarea.
   Si el *registro de la última temperatura* esta vacio o la diferencia de éste y la medición actual es mayor que la establecida en el parámetro ***accuacy*** se dará por finalizada la tarea retornando la temperatura actual y almacenando la nueva temperatura en el *registro de la última temperatura*. Si la diferencia entre las dos mediciones es menor que la establecida en el parámetro ***accuacy*** la tarea finalizará retornando el valor 0 (sin variación). Para calcular la diferencia entre dos mediciones se compararán sus valores, si la diferencia absoluta es mayor a la especificada en el parámetro ***accuacy*** se considerarán mediciones distintas.

b. Si el parámetro ***sensitive*** tiene valor false o no tiene valor: se dará por finalizada la tarea retornando la temperatura actual.

   - Retorno y cambio de estado:
     - **Retorno:** *[Grados (entero) .Temperatura sin cambios: "equal"]*
       Estado: 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

9.6.8 CI_audioCapture.

Parámetros de entrada específicos:

- **format**: define el formato del fichero de audio que se obtiene como respuesta, puede tomar los valores ogg, wav, mp3, mp4, 3gp, aiff, caf.
- **maxTime:** número máximo de segundos de la grabación. En caso de que la grabación actual llegue al máximo de segundos permitido se cortará de manera automática.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utilizará el API de la plataforma para gestionar el micrófono del dispositivo. En primer lugar se ejecutará la acción que permite comenzar una grabación de audio, se utilizará el valor del parámetro ***format*** para seleccionar el formato del fichero de salida, en caso de que este parámetro no haya sido especificado se utilizará el formato por defecto de la plataforma. A partir de este momento existen dos alternativas posibles:

a. Si el parámetro ***maxTime*** tiene valor asignado: la grabación continuará hasta que se cumpla el período de tiempo especificado, momento en el que finalizará la grabación y la tarea.

b. Si el parámetro ***maxTime*** no está definido: el usuario indicará el final de la grabación pulsando uno de los botones que se mostrarán por pantalla, este evento dará por finalizada la tarea.

   - Retorno y cambio de estado:
     - **Retorno:** *[Fichero de audio]* **Estado:** 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

9.6.9 CI_camera.

Parámetros de entrada específicos:

- **format**: define el formato del fichero de tipo imagen que se obtiene como respuesta, puede tomar los valores *png* o *jpg*.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). En primer lugar se utilizará el API de la plataforma para gestionar la cámara de fotos del dispositivo, se ejecutará la acción que permite tomar una fotografía. Se utilizará el valor del parámetro ***format*** para seleccionar el formato de la imagen, en caso de que este parámetro no haya sido especificado se le asignará el valor png. Una vez tomada y guardada la fotografía se dará por finalizada la tarea.

- Retorno y cambio de estado:
  - **Retorno:** *[Fichero fotografía]* **Estado:** 3 (Completada)
    Acción completada correctamente.
  - **Ret:** 0, **Estado:** 4 (Error)
    Se ha producido un error.

9.6.10 CI_location.

Parámetros de entrada específicos:

- **source**: determina el elemento que el dispositivo utilizará para obtener la ubicación actual, puede tomar los valores *GPS* o *Network*.
- **sensitive:** condiciona el envío de la ubicación a que la última ubicación obtenida sea diferente a la anterior, toma como valor un booleano.
- **accuacy:** especifica el rango de variación mínimo para considerar dos ubicaciones como diferentes, toma como valor un número decimal positivo.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utiliza el API específico de la plataforma que permite obtener la ubicación actual del dispositivo. La acción que se emplea para obtener la ubicación suele contar con valores configurables que permiten seleccionar la fuente de la misma, actualmente las fuentes más populares son el GPS y la red. Se utilizará el valor del parámetro ***source*** para seleccionar la fuente, en caso de que el parámetro no tenga un valor asignado se le asignará como valor por defecto *GPS*. Una vez obtenida la ubicación actual existen dos alternativas posibles:

a. Si el parámetro ***sensitive*** tiene valor true: La primera vez que se obtiene la ubicación actual, ésta será almacenada en el *registro de última ubicación*, el valor de este registro se utilizará como elemento de comparación en futuras invocaciones de la tarea.
   Si el *registro de la última ubicación* esta vacio o la diferencia de éste y la ubicación actual es mayor que la establecida en el parámetro ***accuacy*** se dará por finalizada la tarea retornando la ubicación actual y almacenando la nueva ubicación en el *registro de la última ubicación*. Si la diferencia entre las dos ubicaciones es menor que la establecida en el parámetro ***accuacy*** la tarea finalizará retornando el valor 0 (sin variación). Para calcular la diferencia entre dos ubicaciones se compararán los valores de sus coordenadas longitud y latitud de manera independiente (longitudes con longitudes, latitudes con latitudes), si la diferencia absoluta en cualquiera de las dos coordenadas es mayor a la especificada en el parámetro ***accuacy*** se considerarán ubicaciones distintas.

b. Si el parámetro ***sensitive*** tiene valor false o no tiene valor: se dará por finalizada la tarea retornando la ubicación actual.

   - Retorno y cambio de estado:

- **Retorno:** *[Ubicación: lonitud,latitud .Ubicación sin cambios: "equal"]*
  Estado: 3 (Completada)
  Acción completada correctamente.
  - **Ret:** 0, **Estado:** 4 (Error)
    Se ha producido un error.

9.6.11 CIC_audioCaptureSpeech

Parámetros de entrada específicos:

- **language**: idioma para el reconocimiento del discurso, código de idioma ISO 639-1 : EN, ES, FR, EL, IT, JA, etc.

Proceso:

Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). Esta tarea utilizará el API de la plataforma para gestionar el micrófono del dispositivo. En primer lugar se ejecutará la acción que permite comenzar una grabación de audio, cuando el usuario indique el final (pulsando uno de los botones que se mostrarán en pantalla) se detendrá el proceso de grabación. La grabación realizada será procesada aplicando un proceso de reconocimiento de discurso, estos procesos suelen estar incluidos en el API de la mayoría de plataformas móviles, en ocasiones la llamada del API solicita el idioma del discurso (presente en el atributo ***language***), aunque también puede ser inferido. Al identificar el discurso presente en la grabación se dará por finalizada la tarea.

- Retorno y cambio de estado:
  - **Retorno:** *[Discurso – Cadena de texto]*
    Estado: 3 (Completada)
    Acción completada correctamente.
  - **Ret:** 0, **Estado:** 4 (Error)
    Se ha producido un error.

9.6.12 CIC_cameraBarCode

Proceso:

La tarea CIC_cameraBarCode utiliza un algoritmo específico para la decodificación de imágenes que contienen códigos de barras, este algoritmo será aplicado sobre las imágenes capturadas. Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). En primer lugar se utilizará el API de la plataforma para gestionar la cámara de fotos del dispositivo, se ejecutará la acción que permite tomar una fotografía o capturar la visión de la cámara de manera constante. Las imágenes obtenidas serán analizadas con el fin de identificar un código de barras y decodificar la información que éste contiene. Al identificar y decodificar un código de barras se dará por finalizada la tarea.

- Retorno y cambio de estado:
  - **Retorno:** *[Código de barras decodificado – Cadena de texto]*
    Estado: 3 (Completada)
    Acción completada correctamente.
  - **Ret:** 0, **Estado:** 4 (Error)
    Se ha producido un error.

9.6.13 CIC_cameraQRCode

Proceso:

La tarea CIC_cameraQRCode utiliza un algoritmo específico para la decodificación de imágenes que contienen códigos QR, este algoritmo será aplicado sobre las imágenes capturadas. Antes de iniciar el proceso la tarea cambia su estado a 2 (En tratamiento). En primer lugar se utilizará el API de la plataforma para gestionar la cámara de fotos del dispositivo, se ejecutará la acción que permite tomar una fotografía o capturar la visión de la cámara de manera constante. Las imágenes obtenidas serán analizadas con el fin de identificar un código QR y decodificar la información que éste contiene. Al identificar y decodificar un código QR se dará por finalizada la tarea.

- Retorno y cambio de estado:
  - **Retorno:** *[Código QR decodificado – Cadena de texto]*
    Estado: 3 (Completada)
    Acción completada correctamente.
  - **Ret:** 0, **Estado:** 4 (Error)
    Se ha producido un error.

9.6.14 CA_Bluetooth

Parámetros de entrada específicos:

- **action**: de carácter obligatorio, puede tomar los valores ***scan***, ***pairDevice***, ***unPairDevice***, ***connectDevice***, ***write***, ***listen*** y ***disconnectDevice*.**
- **pair:** la conexión requiere un emparejamiento previo.
- **UUID:** identificador universal único del servicio.
- **deviceName:** nombre del dispositivo o identificador único MAC.
- **data**: contiene el valor de la cadena de bytes que será enviada al dispositivo destino.

Controlador de conexiones activas:

Los procesos de comunicación Bluetooth no se basan únicamente en eventos individuales, sino que en muchas ocasiones utilizan procesos dependientes entre sí, por ejemplo antes de poder establecer una conexión con un dispositivo es probable que se requiera un proceso de emparejamiento. La tarea ***CA_Bluetooth*** implementa un controlador de conexiones que almacena cada uno de los emparejamientos realizados y conexiones activas, estos elementos serán utilizados como base para la ejecución de otros procesos.

Procesos:

La tarea ***CA_Bluetooth*** se divide en siete procesos distintos condicionados por el valor del atributo ***action***. Antes de iniciar cualquiera de los siete procesos la tarea cambia su estado a 2 (En tratamiento).

1. **Proceso 1: [action= *scan*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Bluetooth. Se ejecutará la llamada del API que permite descubrir dispositivos Bluetooth cercanos. Se almacenará el nombre y la dirección MAC de cada uno de los dispositivos descubiertos.

   - Retorno y cambio de estado:
     - **Retorno:** *[Lista de dispositivos descubiertos]*
       Estado: 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

2. **Proceso 2: [action= *pairDevice*] [deviceName =*nombre/MAC*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Bluetooth. Se ejecutará la llamada del API que permite establecer el emparejamiento con un dispositivo Bluetooth descubierto, para realizar esta acción se requiere el nombre o la dirección del dispositivo objetivo del proceso.

El emparejamiento requerirá la interacción por parte del usuario, que se encargará de introducir el código requerido en el proceso de emparejamiento.

- Retorno, acciones complementarias y cambio de estado:
  - **Retorno:** 1, **Estado:** 3 (Completada)
    Acción completada correctamente.
  - **Ret:** 0**, Estado:** 4 (Error)
    Se ha producido un error.

3. **Proceso 3: [action= *unPairDevice*] [deviceName =*nombre/MAC*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Bluetooth. Se ejecutará la llamada del API que permite desemparejar un dispositivo Bluetooth previamente emparejado, para realizar esta acción se requiere el nombre o la dirección del dispositivo objetivo del proceso.

   - Retorno, acciones complementarias y cambio de estado:
     - **Retorno:** 1, **Estado:** 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0**, Estado:** 4 (Error)
       Se ha producido un error.

4. **Proceso 4: [action= *connectDevice*] [?pair=*boolean*] [?deviceName =*nombre/MAC*] [UUID=*identificador universal unico*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Bluetooth. El parámetro ***pair*** permite especificar si la conexión requiere un emparejamiento previo, si se requiere y éste no se ha producido con anterioridad el proceso forzará un emparejamiento (Proceso 2). En caso de que no se asigne un valor al parámetro ***pair*** se requerirá el emparejamiento previo a la conexión. La llamada del API que permite establecer la conexión con un dispositivo Bluetooth requiere el identificador del dispositivo objetivo y un UUID que identifica el servicio concreto (en el dispositivo objetivo) al que se va a asociar la conexión.

   - Retorno, acciones complementarias y cambio de estado:
     - **Retorno:** 1, **Estado:** 3 (Completada)
       Acción completada correctamente.
       (1) La nueva conexión activa se almacena en ***controlador de conexiones activas***.
     - **Ret:** 0**, Estado:** 4 (Error)
       Se ha producido un error.

5. **Proceso 5: [action= *write*] [data= *bytes*] [?deviceName =*nombre/MAC*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Bluetooth. Ejecuta la llamada del API que permite enviar una cadena de bytes a un dispositivo Bluetooth (previamente conectado). En este caso la cadena de bytes será la especificada en el parámetro ***data***. El envió de datos requiere una conexión Bluetooth activa como base, para ello este proceso realiza una petición al ***controlador de conexiones activas*** solicitando la conexión que coincida con el identificador del dispositivo recibido como parámetro. En caso de que este parámetro no haya sido especificado se selecciona como base de la comunicación la primera conexión activa almacenada.

   - Retorno y cambio de estado:
     - **Retorno:** 1 **Estado:** 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

6. **Proceso 6: [action= *listen*] [?deviceName =*nombre/MAC*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Bluetooth. Se mantiene a la escucha de una conexión activa hasta recibir una cadena de bytes. Para este objetivo el proceso utiliza la llamada del API que permite recibir información de un canal de conexión. Este proceso requiere una conexión Bluetooth activa como base, para obtenerla se realizará una petición al ***controlador de conexiones activas*** solicitando la conexión que coincida con el identificador del dispositivo recibido como parámetro. En caso de que este parámetro no haya sido especificado se selecciona como base de la comunicación la primera conexión activa almacenada.

   - Retorno y cambio de estado:
     - **Retorno:** [*Cadena de bytes leída],* **Estado:** 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

7. **Proceso 7: [action= *disconnectDevice*] [deviceName =*nombre/MAC*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Bluetooth. En primer lugar realizará una petición al ***controlador de conexiones activas*** solicitando la conexión que coincida con el identificador del dispositivo recibido como parámetro. Posteriormente utilizar el API del

dispositivo para cerrar la conexión activa y la eliminará del ***controlador de conexiones activas***.

- Retorno y cambio de estado:
  - **Retorno:** 1 **Estado:** 3 (Completada)
    Acción completada correctamente.
  - **Ret:** 0**, Estado:** 4 (Error)
    Se ha producido un error.

9.6.15 CA_wi-fi

Parámetros de entrada específicos:

- **action**: de carácter obligatorio, puede tomar los valores ***scan***, ***connectDevice***, ***write***, ***listen*** y ***disconnectDevice*.**
- **deviceName:** nombre del dispositivo o identificador único MAC.
- **data**: contiene el valor de la cadena de bytes que será enviada al dispositivo destino.

Controlador de conexiones activas:

Los procesos de comunicación Wi-Fi no se basan únicamente en eventos individuales, sino que en muchas ocasiones utilizan procesos dependientes entre sí, por ejemplo antes de poder enviarle una cadena de datos a un dispositivo hay que establecer una conexión. La tarea ***CA_wi-fi*** implementa un controlador de conexiones que almacena cada una de las conexiones activas realizadas, estas conexiones serán utilizadas como base para la ejecución de otros procesos.

Procesos:

La tarea ***CA_wi-fi*** se divide en cinco procesos distintos condicionados por el valor del atributo ***action***. Antes de iniciar cualquiera de los cinco procesos la tarea cambia su estado a 2 (En tratamiento).

1. **Proceso 1: [action= *scan*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Wi-Fi. Se ejecutará la llamada del API que permite escanear el radio cercano en busca de dispositivos Wi-Fi Ad hoc. Se almacenará el nombre y la dirección MAC de cada uno de los dispositivos descubiertos.

   - Retorno y cambio de estado:
     - **Retorno:** *[Lista de dispositivos descubiertos]*
       Estado: 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

2. **Proceso 2: [action= *connectDevice*] [?deviceName =*nombre/MAC*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Wi-Fi. Se ejecutará la llamada del API que permite establecer una conexión con un dispositivo Wi-Fi Ad hoc. Si la acción no recibe como parámetro un nombre o dirección de un dispositivo se forzará una acción de búsqueda (Proceso 1), con el fin de establecer el dispositivo destino. El establecimiento de una conexión Wi-Fi puede requerir la interacción por parte

del usuario si es que el dispositivo destino está utilizando una conexión protegida por una clave.

- Retorno, acciones complementarias y cambio de estado:
  - **Retorno:** 1, **Estado:** 3 (Completada)
    Acción completada correctamente.
    (1) La nueva conexión activa se almacena en ***controlador de conexiones activas***.
  - **Ret:** 0, **Estado:** 4 (Error)
    Se ha producido un error.

3. **Proceso 3: [action= *write*] [data= *bytes*] [?deviceName =*nombre/MAC*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Wi-Fi. Ejecuta la llamada del API que permite enviar una cadena de bytes a un dispositivo Wi-Fi Ad Hoc (previamente conectado). En este caso la cadena de bytes será la especificada en el parámetro ***data***. El envió de datos requiere una conexión Wi-Fi activa como base, para ello este proceso realiza una petición al ***controlador de conexiones activas*** solicitando la conexión que coincida con el identificador del dispositivo recibido como parámetro. En caso de que este parámetro no haya sido especificado se selecciona como base de la comunicación la primera conexión activa almacenada.

   - Retorno y cambio de estado:
     - **Retorno:** 1 **Estado:** 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

4. **Proceso 4: [action= *listen*] [?deviceName =*nombre/MAC*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Wi-Fi. Se mantiene a la escucha de una conexión activa hasta recibir una cadena de bytes. Para este objetivo el proceso utiliza la llamada del API que permite recibir información de un canal de conexión. Este proceso requiere una conexión Wi-Fi activa como base, para obtenerla se realizará una petición al ***controlador de conexiones activas*** solicitando la conexión que coincida con el identificador del dispositivo recibido como parámetro. En caso de que este parámetro no haya sido especificado se selecciona como base de la comunicación la primera conexión activa almacenada.

   - Retorno y cambio de estado:
     - **Retorno:** [*Cadena de bytes leída],* **Estado:** 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)

Se ha producido un error.

5. **Proceso 5: [action= *disconnectDevice*] [deviceName =*nombre/MAC*]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación Bluetooth. En primer lugar realizará una petición al ***controlador de conexiones activas*** solicitando la conexión que coincida con el identificador del dispositivo recibido como parámetro. Posteriormente utilizar el API del dispositivo para cerrar la conexión activa y la eliminará del ***controlador de conexiones activas***.

   - Retorno y cambio de estado:
     - **Retorno:** 1 **Estado:** 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

9.6.16 CA_nfc

Parámetros de entrada específicos:

- **action**: puede tomar los valores ***write*** y ***read*.**
- **data**: contiene el valor de la cadena de bytes que será enviada para su escritura al elemento NFC destino.

Procesos:

La tarea ***CA_nfc*** se divide en dos procesos distintos condicionados por el valor del atributo ***action***. Antes de iniciar cualquiera de los dos procesos la tarea cambia su estado a 2 (En tratamiento).

1. **Proceso 1: [action=write] [data=bytes]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación NFC. En primer lugar el dispositivo escanea el radio cercano en busca de un elemento NFC accesible. Una vez detectado ejecuta la llamada del API que permite escribir una cadena de bytes en un elemento NFC. En este caso la cadena de bytes será la especificada en el parámetro ***data***.

   - Retorno y cambio de estado:
     - **Retorno:** 1, **Estado:** 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0, **Estado:** 4 (Error)
       Se ha producido un error.

2. **Proceso 1: [action=read]**
   Utiliza el API específico de la plataforma para gestionar el elemento de comunicación NFC. En primer lugar el dispositivo escanea el radio cercano en busca de un elemento NFC accesible. Una vez detectado ejecuta la llamada del API que permite leer los datos guardados en el elemento NFC.

   - Retorno y cambio de estado:
     - **Retorno:** [*Cadena de bytes leída],* **Estado:** 3 (Completada)
       Acción completada correctamente.
     - **Ret:** 0**, Estado:** 4 (Error)
       Se ha producido un error.

PARTE IV

DESARROLLO DE PROTOTIPOS Y EVALUACIÓN

CAPÍTULO 10

DESARROLLO DE PROTOTIPOS

10.1 INTRODUCCIÓN

En este capítulo se presenta el proceso de desarrollo y posterior uso de tres aplicaciones web; estas aplicaciones han sido desarrolladas combinando la utilización de tecnologías web tradicionales con la especificación propuesta.

El objetivo principal de estas aplicaciones es presentar el proceso de desarrollo a la vez que se ilustra parte de la funcionalidad de la propuesta. Adicionalmente algunos de estos prototipos serán utilizados para realizar posteriores evaluaciones de la propuesta.

Los prototipos desarrollados son los siguientes:

1. Buscador del mejor precio. *Interactúa con un objeto físico.*

2. Navegador GPS. *Interactúa con el contexto.*

3. Control Remoto para centro Multimedia. *Interactúa con un dispositivo electrónico.*

Para interpretar los prototipos listados se ha desarrollado una versión del navegador web sensible al contexto para la plataforma móvil Android. Esta versión del navegador sigue fielmente todas las especificaciones y principios presentados en esta tesis, por lo tanto es capaz de satisfacer los requerimientos expresados por las etiquetas XML de contexto.

10.2 BUSCADOR DEL MEJOR PRECIO

Información requerida:	Etiquetas XML de contexto
Ubicación	<location/>
Lectura del código de barras de un producto.	<camerabarcode/>

Tabla 10.1 – Información requerida por el prototipo "buscador del mejor precio".

Las etiquetas XML de contexto en combinación con el navegador web sensible al contexto otorgan a las aplicaciones web capacidades funcionales que anteriormente eran difíciles de conseguir.

En este primer prototipo se han utilizado las etiquetas XML de contexto para desarrollar una aplicación web capaz de buscar el mejor precio de un determinado producto en los supermercados cercanos al usuario. Este tipo de aplicaciones son bastante comunes en la web, en este caso cambiaremos los mecanismos manuales de entrada de datos por etiquetas XML de contexto. Las etiquetas habilitarán la captura automática de parte de la información que se requiere para ejecutar la lógica de negocio de la aplicación. La aplicación web desarrollada tiene una funcionalidad similar a la aplicación ShopSavvy [11]de Android.

La aplicación web combinará la utilización de tecnologías web tradicionales como HTML, PHP y css con dos etiquetas XML de contexto. (1) La primera etiqueta XML de contexto ***<camerabarcode/>*** se utilizará para permitir al usuario identificar un producto escaneando el código de barras del mismo, en lugar de introducir el nombre del producto en un formulario. (2) Para detectar la ubicación del usuario y tener un punto de referencia que poder utilizar para realizar una búsqueda de supermercados cercanos se utiliza la localización actual del dispositivo, esta ubicación se obtendrá mediante el chip GPS del dispositivo, para ello se utilizará la etiqueta XML de contexto ***<location/>***.

La aplicación web comparadora de precios está alojada en un servidor Apache estándar. En este caso el dispositivo que interactúa con la aplicación es un Smartphone con sistema operativo Android. El dispositivo está ejecutando una versión del navegador web sensible al contexto que ha sido desarrollado siguiendo las especificaciones presentadas en esta tesis.

La página index.php de la aplicación web incluye la etiqueta XML de contexto ***<location receiver='location.php'/>***, que expresa un requerimiento de envío de la posición actual. La información se enviara en una petición POST a la URL definida en el atributo ***receiver*** de la etiqueta XML de contexto. A continuación se muestra un fragmento del código fuente de la página index.php.

[11] https://play.google.com/store/apps/details?id=com.biggu.shopsavvy

```
<body>
...
<location receiver='location.php'/>

<form id="form" name="form" method="post" action="code.php"
enctype="multipart/form-data">

    <label>ZIP Code:</label>
    <input type="text"
           name="location" id="location" value=""/>
    <button type="submit">Accept</button>
</form>
...
```

La página location.php actúa como receptor de las peticiones POST, las cuales contienen las coordenadas de la ubicación actual del dispositivo. Location.php procesa la petición y almacena la ubicación en una cookie en el servidor, para posteriormente aplicar esta ubicación a todas las búsquedas que se realicen dentro de la misma sesión. Para establecer la relación entre el dispositivo y la ubicación se utilizará la cabecera DSK, la cual viaja en todas las peticiones que realiza el dispositivo.

Una vez la aplicación web ya tiene la ubicación actual del usuario ya se pueden efectuar búsquedas de un producto concreto. La página code.php contiene la etiqueta XML de contexto ***<camerabarcode receiver='barcode.php'/>***. Cuando el usuario inicia la tarea asociada a la etiqueta se lanzará una nueva actividad en el teléfono, esta actividad utiliza la cámara de fotos en modo escáner. Cuando la cámara detecta un código de barras finaliza el escaneo y le pide al usuario que pulse un botón para confirmar que la acción se ha realizado correctamente. Inmediatamente después, el navegador web sensible al contexto envía la posición POST que contiene la decodificación del código de barras del producto a la URL 'barcode.php'. A continuación se muestra un fragmento del código fuente de la página index.php:

```
<body>
...
<h2> Find Products </h2>
<p> Barcode Scanner </p>
<camerbarcode receiver='barcode.php'/>
...
```

La página barcode.php es la encargada de recibir y procesar las peticiones POST que contienen el identificador del producto que se pretende buscar. Utilizando la ubicación actual y el identificador único del producto extraído a partir del código de barras, la aplicación web realizará una consulta a un servicio web para obtener los mejores resultados, cercanía/precio.

El navegador web sensible al contexto procesa e interpreta las etiquetas XML de contexto, mostrándolas como botones en la pantalla del dispositivo. Cuando el usuario pulsa sobre uno de los botones, el navegador web sensible al contexto ejecuta la tarea programada de contexto asociada (Fig.10.1).

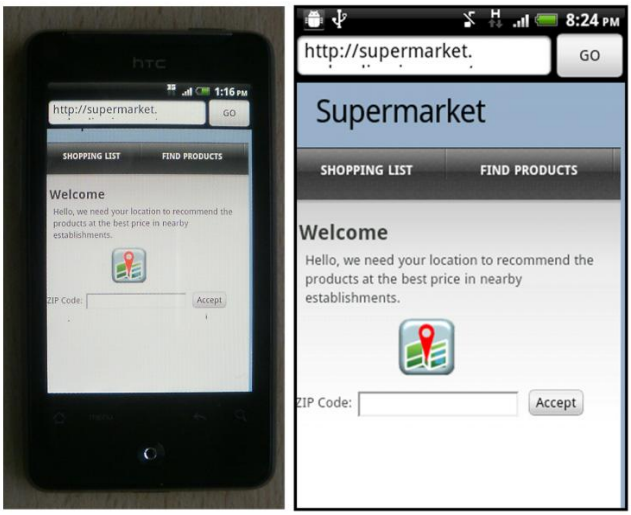

Figura 10.1 - En la captura de pantalla el botón de localización asociado a la etiqueta XML de contexto notifica al usuario la solicitud de su ubicación actual.

La tarea que se encarga de obtener la ubicación actual del dispositivo utiliza el chip GPS para obtener las coordenadas longitud y latitud actuales. Una vez obtenidas, se envían encapsuladas en una petición POST a la URL baseURL/location.php. Durante la ejecución de la tarea, el botón asociado a la etiqueta XML de contexto cambia su icono. Mediante estos cambios, el usuario puede percibir que se está ejecutando una tarea. Cuando finaliza la captura y envío de datos el estilo del botón cambia, para indicar que la tarea se completó con éxito (Fig.10.2).

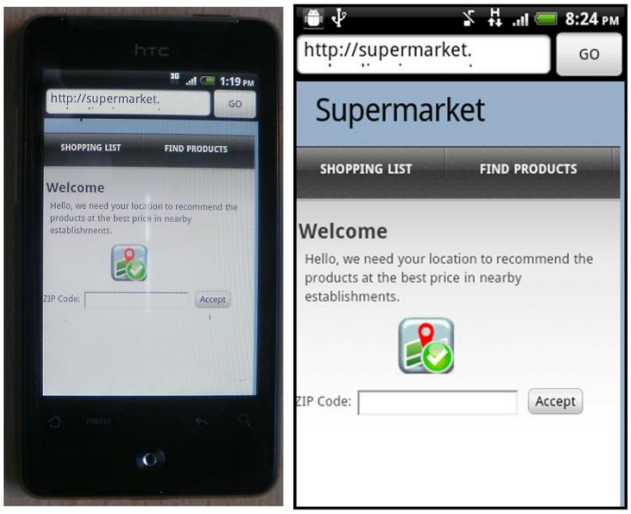

Figura 10.2 - En la captura de pantalla el botón asociado a etiqueta XML de contexto indica que el proceso de obtención y envío de la localización ha finalizado con éxito.

En este caso se han implementado dos páginas que se encargan de procesar y almacenar la información de contexto que el dispositivo envía para satisfacer los requerimientos expresados en las etiquetas XML de contexto. En este prototipo, la aplicación web

almacena los datos recibidos en una cookie del servidor, el identificador único de esta cookie será el DSK (Fig.10.3).

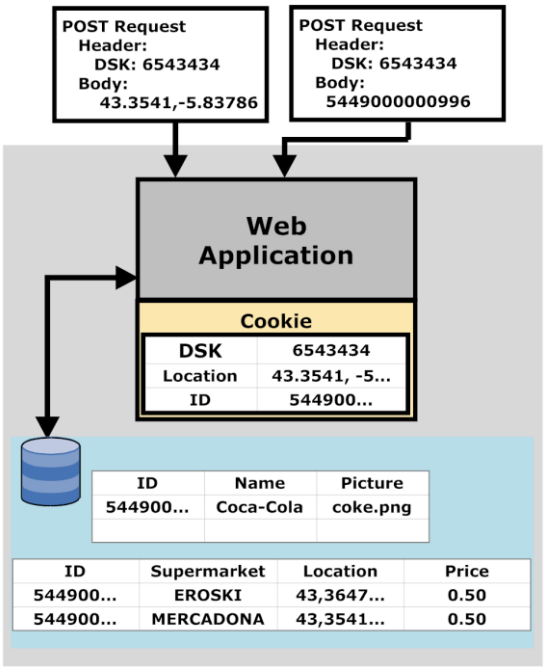

Figura 10.3 - La petición POST contiene la cabecera DSK, la aplicación web utiliza este atributo para identificar el dispositivo del que provienen las peticiones procesadas.

Cuando el usuario pulsa sobre el botón asociado a la etiqueta XML de contexto ***<camerabarcode/>***, el navegador web sensible al contexto inicia la tarea correspondiente (Fig.10.4). La mayoría de las tareas programadas de contexto no requieren de la interacción del usuario, otras como esta sí que requieren algún tipo de acción. La tarea programada tiene como objetivo decodificar un código de barras. Para conseguir su objetivo, la tarea muestra por pantalla la vista de la cámara en modo escáner (Fig.10.5), cuando la cámara detecta un código de barras la vista de la cámara se cierra y se vuelve a la pantalla anterior (la página web). La tarea programada de contexto se encarga de decodificar el código de barras, posteriormente envía la información en una petición POST a la URL baseURL/barcode.php.

En este prototipo se ha implementado la página web barcode.php de una forma muy similar a como se hizo en location.php, utilizando el atributo DSK que viaja en la cabecera de las peticiones POST para identificar al dispositivo y la sesión.

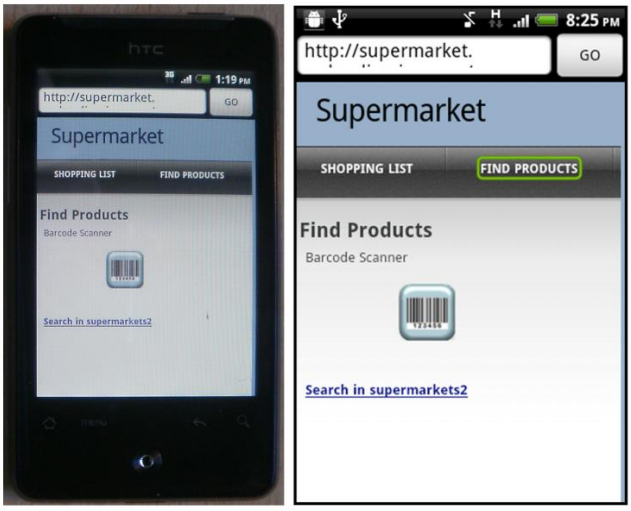

Figura 10.4 - En la captura de pantalla el botón asociado a la etiqueta XML de contexto <camerabarcode/> notifica al usuario el requerimiento de lectura de un código de barras.

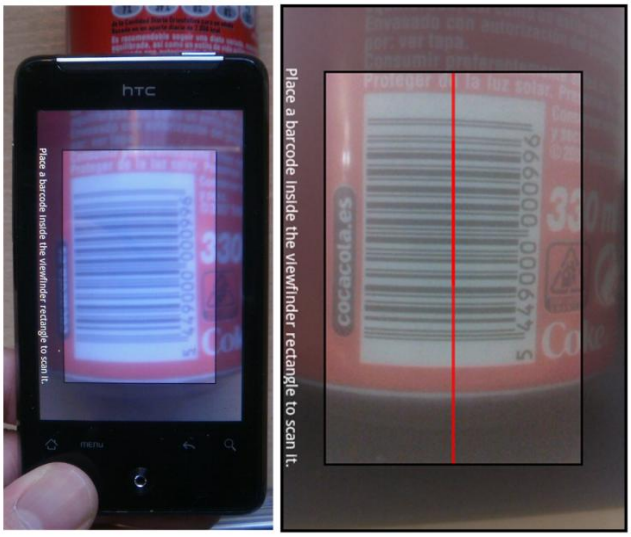

Figura 10.5 - La captura de pantalla muestra la cámara de fotos del teléfono en modo escáner de códigos de barras, esta vista se mostrará durante la ejecución de la tarea programada asociada a la etiqueta XML de contexto <camerabarcode/>.

Una vez que el código de barras ha sido correctamente enviado a la aplicación web el botón asociado a la etiqueta XML de contexto cambia de apariencia, para notificar al usuario que la tarea ha finalizado con éxito (Fig.10.6). Ahora, la aplicación web tiene almacenados en una cookie asociada a la sesión los dos datos que necesita para realizar las búsquedas: la localización actual del usuario y el identificador del producto. El usuario de la aplicación web puede pulsar en el enlace "***buscar en supermercados***". Cuando la aplicación web recibe la petición de búsqueda utiliza la DSK contenida en la petición para seleccionar la cookie que contiene los datos que el usuario envió previamente. Los datos contenidos en la cookie son utilizados por la aplicación web para realizar una consulta a un servicio web, este servicio web retorna una lista de supermercados y precios. Los resultados de la búsqueda se muestran en la página "product.php" (Fig.10.7).

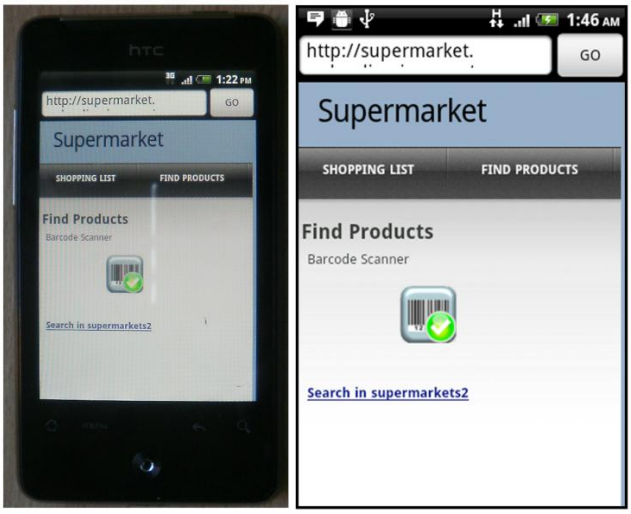

Figura 10.6 - En la captura de pantalla el botón asociado a la etiqueta XML de contexto <camerabarcode/> indica que el proceso de obtención y envío del código de barras ha finalizado con éxito.

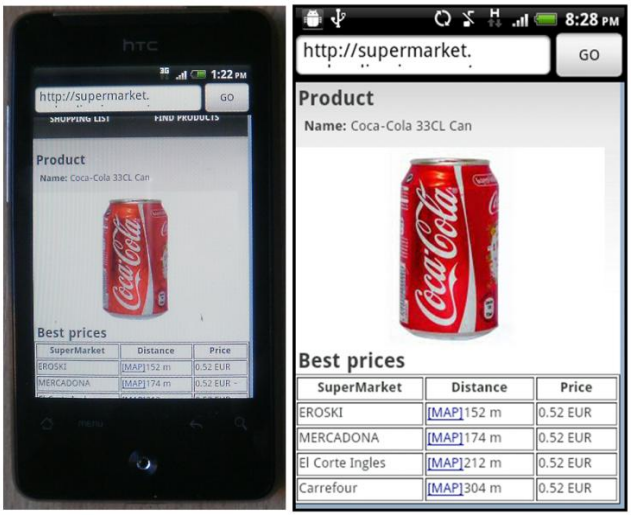

Figura 10.7 - En la captura de pantalla se muestra la información relativa al producto y el precio al que puede comprarse en los supermercados cercanos.

10.3 NAVEGADOR GPS

Información requerida:	Etiquetas XML de contexto
Ubicación	<location/>
Sensor de campo magnético (Brújula)	<magneticfield/>

Tabla 10.2– Información requerida por el prototipo "navegador GPS".

En este segundo prototipo se han utilizado las etiquetas XML de contexto para desarrollar una aplicación web que actúa como un navegador GPS, en esencia la aplicación será similar a Google Navigator o Tomtom[12] aunque este prototipo tendrá una funcionalidad mucho más reducida. La aplicación web navegador GPS se construirá utilizando etiquetas XML de contexto en combinación con las tecnologías web PHP, HTML , javascript y el API Google Maps.

La mayoría de navegadores GPS utilizan dos fuentes de información: el chip GPS y una brújula, aunque este último elemento no resulta imprescindible resulta útil para poder situar el indicador orientado hacia la posición correcta.

La página principal de la aplicación es index.php; esta página contiene la vista del mapa y las dos etiquetas XML de contexto que se utilizarán para solicitar la información. La primera etiqueta XML de contexto solicita la ubicación actual del dispositivo. La segunda hace referencia al valor registrado por el sensor de campo magnético del dispositivo. El sensor de campo magnético puede ser utilizado en algunos escenarios para simular una brújula. Para poder refrescar la ubicación del usuario en el mapa los navegadores GPS deben obtener la ubicación actual del dispositivo cada pocos segundos. Las etiquetas XML de contexto permiten configurar ciertos aspectos de los requerimientos de información de contexto mediante la utilización de atributos. En este caso la etiqueta que se encarga de solicitar la ubicación ***<location/>*** utilizará varios atributos opcionales. Estableceremos que la ubicación actual se envíe cada 3 segundos, para ello se incluye el atributo ***timeinterval*** en la declaración de la etiqueta XML de contexto. La etiqueta XML de contexto ***<location/>*** puede utilizar otros atributos que sirven para optimizar su funcionamiento. En muchas ocasiones no es interesante registrar los pequeños cambios en la ubicación del usuario, puesto que estos cambios no son demasiado relevantes para la funcionalidad del navegador GPS. El atributo ***sensitive="true/false"*** permite configurar la tarea programada de contexto asociada a la etiqueta ***<location/>*** , de tal forma que esta solo envíe la ubicación actual del dispositivo si la última coordenada obtenida (longitud, latitud) experimenta una variación mayor al rango especificado en el atributo ***accuacy="int"***. Mediante la utilización de estos atributos se puede optimizar el comportamiento de la aplicación, reduciendo la cantidad de información que envía el navegador web sensible al contexto al servidor.

En este caso el requerimiento de información de contexto de ubicación ha sido configurado para que actualice la ubicación actual cada 3 segundos mediante el atributo **timeinterval="3"**, pero antes de hacerlo comprobará si la nueva ubicación tiene una

[12] http://www.tomtom.com

diferencia de más de 10 unidades en alguna de sus coordenadas latitud o longitud respecto a la ubicación obtenida anteriormente. En caso de que la variación sea menor a la especificada en el atributo ***accuacy="3" (sensitive="true"),*** la tarea emitirá el envío. La ubicación se enviará a la URL especificada en el atributo ***receiver="/services/location.php"***

La segunda de las etiquetas XML de contexto es ***<magneticfield/>*** **,** esta etiqueta solicita la medición actual del sensor electromagnético. En este caso la etiqueta se configurará de una fórmula muy similar a la etiqueta XML de contexto **<location/>** utilizada anteriormente. Los atributos de la etiqueta especifican que la tarea programada de contexto debe enviar el valor actual del campo magnético a la URL ***receiver="/services/magneticfield.php*** cada tres segundos mediante ***timeinterval="3"***, siempre y cuando este valor tenga una variación de al menos 10 unidades respecto al anteriormente registrado ***sensitive="true" accuacy="10"*****.** A continuación se muestra el fragmento de código de la página index.php que contiene las dos etiquetas XML de contexto.

```
<html>
<body onload="startmap()">
    <div id="map_canvas"
        style="width:100%; height:100%"></div>
    <location
        timeinterval="3" provider="GPS"
        sensitive="true" accuacy="10"
        receiver="/services/location.php" />

    <magneticfield
        timeinterval="3"
        sensitive="true" accuacy="10"
        receiver="/services/magneticfield.php" />
```

El navegador web sensible al contexto interpreta y representa las etiquetas XML de contexto como botones. Cuando el usuario pulsa sobre el botón asociado a la etiqueta XML de contexto ***<magneticfield/>*** comienza la ejecución de la tarea asociada. Esta tarea condiciona su comportamiento en base a los atributos definidos en la declaración de la etiqueta XML de contexto. En este caso la tarea programada de contexto que captura el valor del campo magnético se ejecuta en forma de bucle cada tres segundos. Cuando esta tarea repetitiva comienza, el botón que se muestra por pantalla cambia su estado visual a "en proceso" (Fig.10.8).

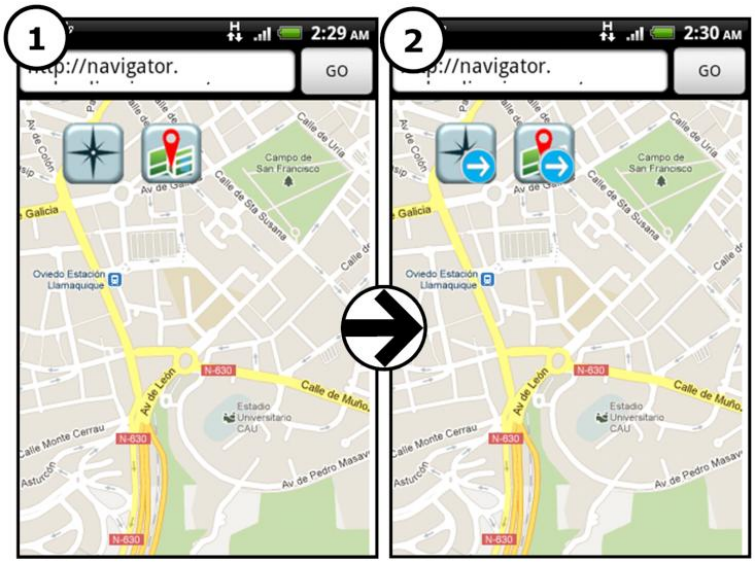

Figura 10.8 - En la primera captura de pantalla el usuario no ha dado su permiso para la captura y envío de la información de contexto. En la segunda captura el usuario ha pulsado sobre los botones asociados a las etiquetas XML de contexto, iniciando así la ejecución de las tareas programadas.

La aplicación web implementa dos páginas web de tipo PHP, que tienen como objetivo recoger y procesar la información de contexto procedente del dispositivo. Tanto la página "/services/location" como "/services/magneticfield" procesan las peticiones POST recibidas, almacenando los valores en una cookie del servidor. Al igual que en el prototipo presentado anteriormente se utiliza el valor de la cabecera DSK para identificar la procedencia de las peticiones.

Las peticiones que el navegador web sensible al contexto realiza a la página index.php también contienen la cabecera DSK, de esta forma la aplicación web establece un nexo de unión entre la información que recibe por ambas vías. La página index.php utiliza el API de Google Maps para implementar un cliente rico implementado principalmente utilizando Javascript. Normalmente la mayoría de los clientes ricos realizan llamadas a servicios web para actualizar sus interfaces de usuario. En este caso se han implementado dos servicios web muy simples, que serán invocados desde la página index.php. Estos servicios web se limitan a enviar la información almacenada en la cookie del servidor para un DSK concreto, el cual reciben como parámetro.

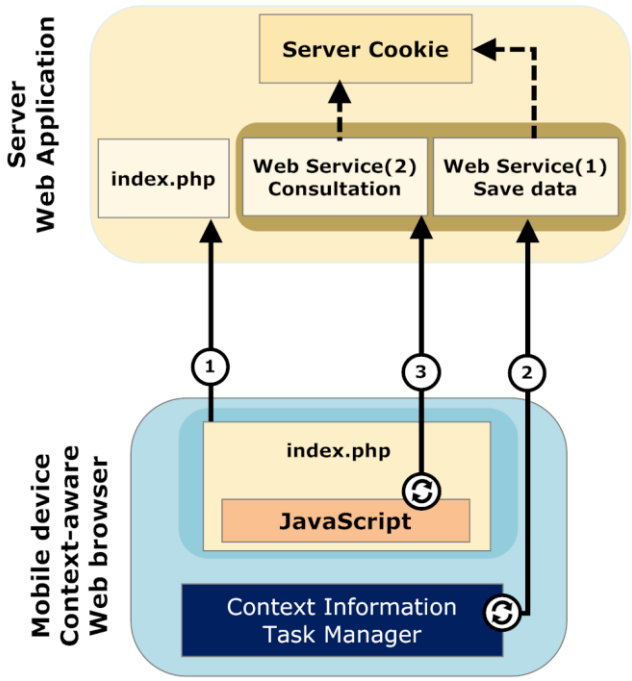

Figura 10.9 - Diagrama de funcionamiento de la aplicación web navegador GPS. (1) El navegador web sensible al contexto carga la página principal. (2) La página contiene etiquetas XML de contexto, la ejecución de las tareas programadas de contexto correspondientes envían la información requerida al servidor, éste almacena la información en una cookie (3) La página principal realiza llamadas a los servicios web de consulta para obtener la ubicación y la orientación.

Cuando el usuario expresa su consentimiento para proceder a la captura y envío de la información de contexto, el navegador GPS comienza a funcionar. La ubicación y la orientación del dispositivo se envían cada tres segundos al servidor para que la posición y la orientación del indicador del mapa se actualicen en consecuencia. Las etiquetas XML de contexto se encuentran en una capa gráfica distinta al HTML, esta división en capas permite tratar ambas de manera independiente. Esta versión del navegador web sensible al contexto permite hacer invisibles los botones asociados a las etiquetas XML de contexto, ya que muchas veces no resulta necesario que estos permanezcan en pantalla una vez cumplido su objetivo (Fig.10.10). La ocultación de los botones no detendrá las tareas programadas de contexto en ejecución, estas sólo finalizarán cuando se cierre el navegador web sensible al contexto (Fig.10.11).

Figura 10.10 - Las capturas de pantalla muestran la posibilidad de ocultar los botones asociados a las etiquetas XML de contexto.

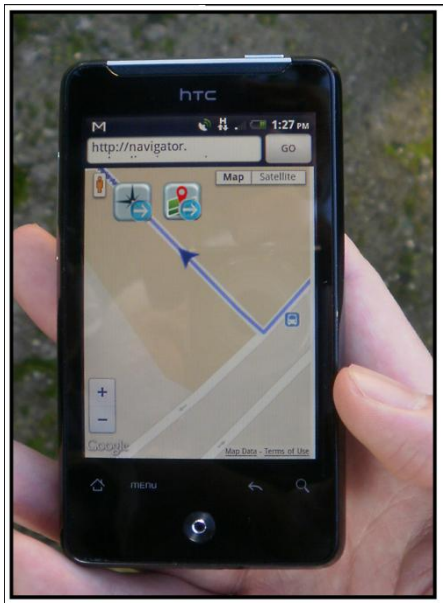

Figura 10.11 - Foto aplicación web Navegador GPS en funcionamiento.

10.4 CONTROL REMOTO PARA CENTRO MULTIMEDIA

Información requerida: Bluetooth	Etiquetas XML de contexto
	<bluetooth/>

Tabla 10.3– Información requerida por el prototipo "control remoto para centro multimedia".

En este tercer prototipo se desarrollará una aplicación web que utiliza etiquetas XML de contexto para habilitar la comunicación inalámbrica entre el dispositivo objeto que ejecuta el navegador web sensible al contexto y un dispositivo electrónico cercano.

La aplicación web desarrollada utiliza la especificación de las etiquetas XML de contexto para modelar un proceso de comunicación basado en la tecnología Bluetooth. En este proceso de comunicación se emplean varias acciones típicas, como el establecimiento de conexiones y el intercambio de datos entre dispositivos. La aplicación web desarrollada será interpretada por una versión del navegador web sensible al contexto.

Uno de los objetivos de este prototipo es ilustrar el proceso de desarrollo de las aplicaciones web que utilizan las etiquetas XML de contexto para implementar nuevas posibilidades de comunicación con dispositivos electrónicos cercanos. La funcionalidad seleccionada para la aplicación web es similar a la de una aplicación de control remoto de un centro multimedia. Normalmente, este tipo de controladores remotos suelen implementarse como aplicaciones nativas, debido a que requieren controlar los mecanismos de comunicación del dispositivo.

La aplicación web consta de una única página index.html, desde ella el usuario gestiona el proceso de conexión y envío de comandos al equipo que está ejecutando el servicio Bluetooth, el servicio implementa la lógica que permite controlar de manera remota el centro multimedia.

Para que la aplicación web defina cómo debe establecerse la conexión con el equipo destino se utiliza la etiqueta ***<bluetooth/>*** combinada con el atributo ***action="connectDevice"***. Este atributo indica que debe realizarse una nueva conexión. Cuando la etiqueta bluetooth define una acción de tipo "***connectDevice***" ésta puede ir acompañada de otros atributos, como por ejemplo: "***deviceName*"**, para definir el nombre o la dirección del dispositivo objeto de la conexión, "***UUID***" para definir el identificador del servicio, o **"*pair*"** que se utiliza para indicar si la conexión requiere un emparejado previo entre dispositivos. En este caso, como el dispositivo destino puede cambiar dependiendo del ordenador destino de la comunicación, no se debería utilizar el atributo **"*deviceName*"**. Ante la omisión de este atributo el navegador web realiza un escaneado de dispositivos Bluetooth cercanos y permite elegir como destino de la conexión a uno de ellos. En este caso el atributo **"*UUID*"** sí será necesario, ya que permite identificar el servicio al que debemos conectarnos, es decir, el servicio que contienen la lógica de negocio para la recepción de los comandos que se aplican sobre el reproductor multimedia. El UUID del servicio será siempre el mismo, independientemente del dispositivo en el que se esté ejecutando el servicio. Por defecto el emparejado de dispositivos que se específica con la etiqueta **"pair"** tiene el valor true, por lo que este atributo **"pair"** puede omitirse si se pretende obligar a realizar un emparejamiento antes de realizar la conexión. En caso de

especificar que no se requiere emparejamiento previo habría que comprobar que el dispositivo destino está preparado para soportar conexiones inseguras.

Además de los atributos que configuran el comportamiento de la acción de comunicación, se añadirán otros atributos a la etiqueta ***<bluetooth/>.*** En primer lugar, se incluirán los atributos **"x"** e **"y"** para especificar la posición de la representación gráfica correspondiente a la etiqueta XML de contexto. La sintaxis final de la etiqueta será la siguiente:

```
<bluetooth  action="connectDevice"
            UUID="5ee01872-4c9e-48d6-9f61-015ff816b278"
            x="50" y="45"/>
```

Una vez se haya establecido la conexión entre los dispositivos, la aplicación web debe permitir al usuario seleccionar los comandos que se van a enviar al centro multimedia, con el objetivo de controlar remotamente su funcionamiento. La aplicación web permitirá realizar seis acciones diferentes:

(1) Avanzar la reproducción.

(2) Iniciar reproducción.

(3) Detener reproducción.

(4) Retroceder la reproducción.

(5) Subir volumen

(6) Bajar volumen.

Cada una de estas acciones está definida utilizando un código numérico, el cual será interpretado por el servicio para ejecutar la funcionalidad oportuna sobre el reproductor multimedia.

Cada acción corresponde a un envío de información, para definir estos procesos se utilizará la etiqueta XML de contexto ***<bluetooth/>*** combinada con el atributo ***action="write"***, este tipo de acción obliga a utilizar el atributo ***data="bytes"***. Utilizando estos atributos se puede especificar el mensaje que se escribirá en el hilo de conexión y, por tanto, los datos que recibirá el dispositivo destino cuando se ejecute la acción. Para que las acciones de comunicación de envío de información puedan realizarse con éxito, el navegador web debe estar conectado al centro multimedia (Fig.10.12). Las tareas asociadas a las etiquetas que definen los envíos de los seis comandos tomarán como referencia la conexión previamente establecida.

Al aplicar varias etiquetas XML de comunicación de un mismo tipo se hace casi imprescindible el uso de atributos específicos que permitan personalizar la representación gráfica. Sin estos atributos todas las etiquetas de un mismo tipo se representarían utilizando la imagen por defecto y el usuario no sería capaz de distinguirlas. Concretamente, en este tipo de aplicación web resulta importante conseguir un aspecto gráfico que favorezca la experiencia del usuario. Por ello se añadirá a las seis etiquetas responsables de transmitir los comandos el atributo de carácter general **"*image*"**, este

atributo permite especificar la URL de la imagen que se mostrará en el botón asociado a la etiqueta XML de contexto. A continuación se muestra el fragmento de código de la página index.html que contiene las etiquetas XML de contexto que definen los envíos de comandos al dispositivo destino:

```
<p>MediaPlayer Remote Control</p>

<div id="stylized" class="myform">
    <h2> HTML + Communication Action XML Tags</h2>
    <p> 1. Connect the device to PC using the button at the top</p>
    <p> 2. Use the buttons above to send commands to the MediaPlayer
</p>
    </br>
    <bluetooth action="connectDevice"
            UUID="5ee01872-4c9e-48d6-9f61-015ff816b278" x="50" y="45"/>
    <bluetooth action="write" data="1" x="20" y="55"
            image="backward.png"/>
    <bluetooth action="write" data="2" x="40" y="55"
            image="play.png"/>
    <bluetooth action="write" data="3" x="60" y="55"
            image="stop.png"/>
    <bluetooth action="write" data="4" x="80" y="55"
            image="forward.png"/>
    <bluetooth action="write" data="5" x="40" y="65"
            image="up_vol.png"/>
    <bluetooth action="write" data="6" x="50" y="65"
            image="down_vol.png"/>
```

En esta aplicación web no resulta imprescindible incluir una etiqueta ***<bluetooth action="disconnectDevice" .../>***, ya que la conexión entre los dos dispositivos se cortará de manera automática cuando el usuario abandone la aplicación web.

El navegador web sensible al contexto representa las etiquetas XML de contexto como botones. La primera etiqueta, define el proceso de conexión y utiliza la imagen por defecto de la tarea bluetooth. El resto de etiquetas utilizan iconos personalizados.

Cuando el usuario pulsa sobre el botón asociado a la etiqueta XML de contexto **<bluetooth action="connectDevice" .../>** da comienzo la ejecución de la tarea programada de contexto correspondiente. Como no se ha especificado un dispositivo destino, el primer paso de la tarea consiste en realizar un escaneo con el fin de encontrar dispositivos bluetooth cercanos (Fig.10.13).

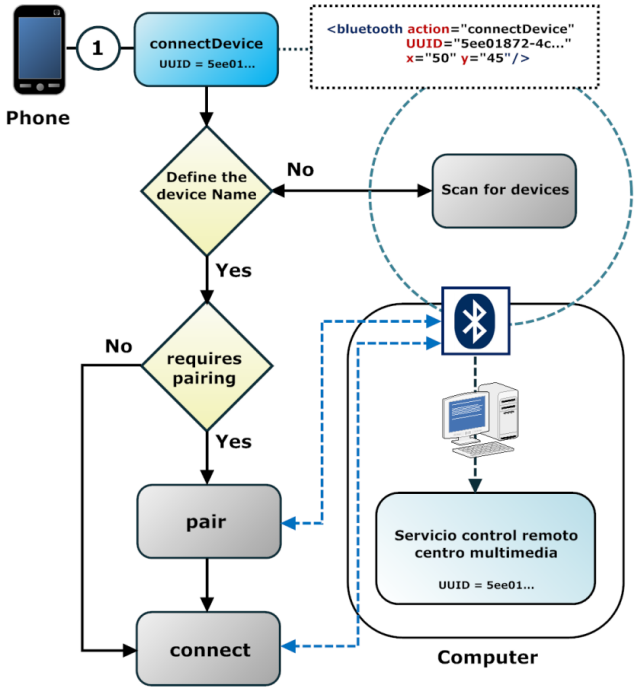

Figura 10.12 - Flujo de las acciones derivadas de la ejecución de la tarea programada de contexto que específica cómo debe establecerse la conexión entre los dispositivos. En función de los atributos utilizados en la etiqueta, el XML de contexto el flujo de ejecución de las tareas puede variar.

Figura 10.13 - En la primera captura de pantalla el usuario no ha dado su permiso para establecer una conexión vía bluetooth con otro dispositivo. En la segunda captura de pantalla el usuario ha pulsado sobre el botón que lanza la tarea de conexión. Dado que la etiqueta XML de contexto no especificaba el dispositivo destino, se realizará un proceso de escaneo, con el fin de descubrir dispositivos cercanos.

Para continuar con la tarea de conexión el usuario debe seleccionar entre uno de los dispositivos descubiertos contenidos en la lista, concretamente el que ejecuta el servicio que permite controlar el centro multimedia. Como en la etiqueta XML de contexto que

define el proceso de conexión no se ha hecho ninguna referencia específica al emparejado, la conexión requerirá el emparejamiento previo de los dispositivos (Fig.10.14).

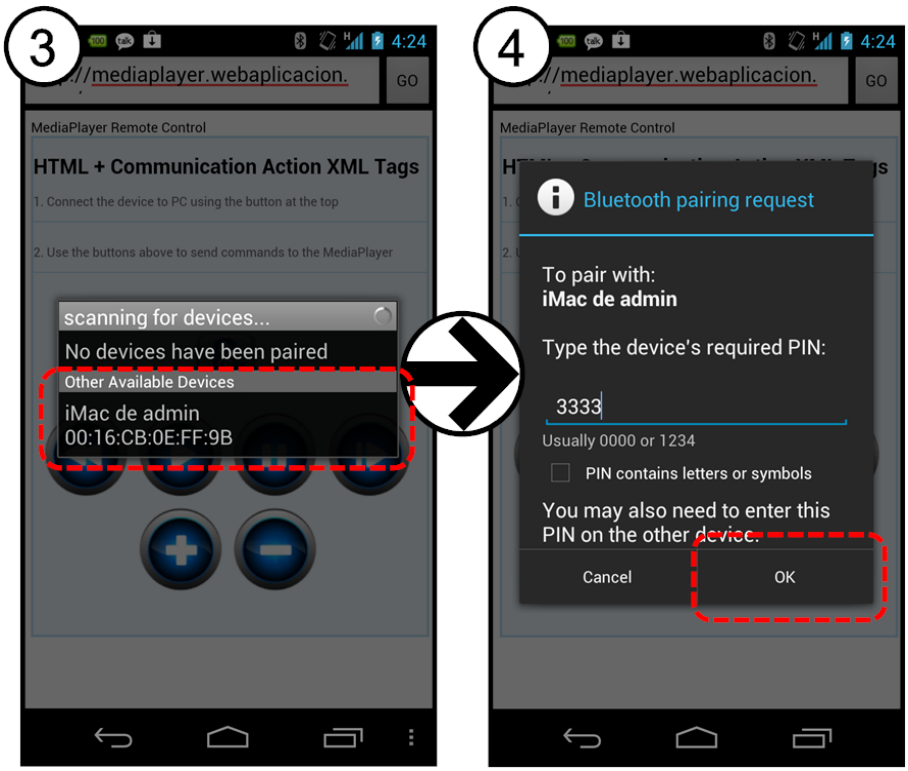

Figura 10.14 - En la primera captura de pantalla el navegador web muestra la lista de dispositivos descubiertos, el usuario debe seleccionar uno de ellos como dispositivo destino de la conexión. La segunda captura de pantalla muestra el formulario de emparejamiento de dispositivos.

Después del emparejado el proceso de conexión continua durante unos segundos, un cambio en el aspecto gráfico en el botón asociado a la etiqueta XML de contexto notifica al usuario que la conexión se establecido con éxito. A partir de este momento, los envíos definidos en las otras seis etiquetas XML de comunicación tomarán como base la conexión establecida. Cuando el usuario pulsa sobre uno de los botones asociados a las etiquetas XML de contexto, el navegador realiza un envío de datos contra el servicio con la UUID especificada. Los datos transmitidos contienen el código identificativo de un comando, estos comandos desencadenan la ejecución de diferentes eventos que permiten gestionar el reproductor multimedia.

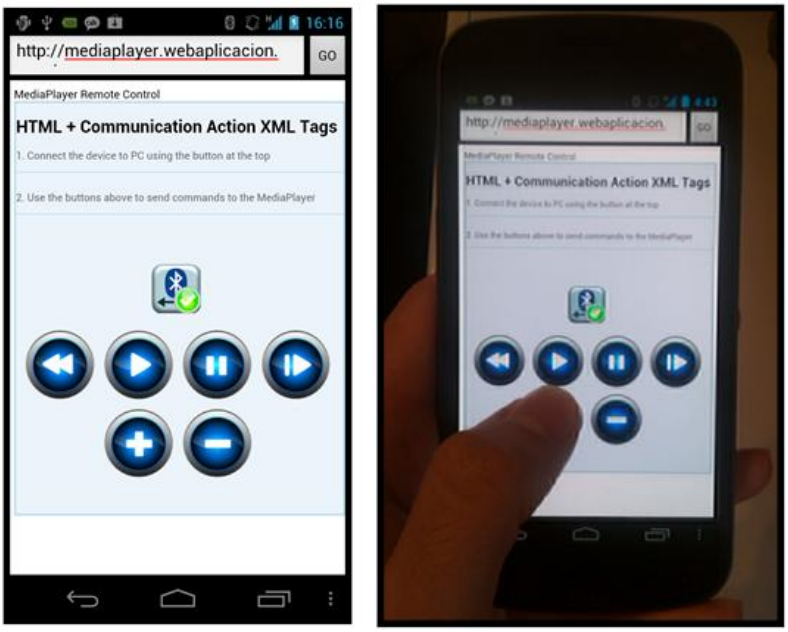

Figura 10.15 - Captura de pantalla aplicación web control remoto para centros multimedia en funcionamiento.

El dispositivo destino ejecuta un servicio que ha sido implementado utilizando java. Para la implementación de las funcionalidades relativas a la conexión Bluetooth, el servicio emplea la librería ***bluecove*** [13]. Inicialmente el servicio está a la espera de conexiones, una vez que se produce una conexión se inicia el proceso de recepción de comandos. Los comandos son cadenas de bytes, estas cadenas son analizadas por la lógica del servicio en busca de coincidencias con los identificadores de los comandos que han sido programados. Cuando se detecta una coincidencia se procede a ejecutar una acción sobre el centro multimedia, subir el volumen, bajarlo, etc (Fig.10.16).

Figura 10.16 - La fotografía muestra el equipo que está siendo controlado remotamente por la aplicación web.

[13] http://bluecove.org/

CAPÍTULO 11

EVALUACIÓN

11.1 INTRODUCCIÓN

En este capítulo se presentan los resultados de las diferentes evaluaciones a las que han sido sometidas varias aplicaciones web que utilizaban la especificación propuesta. Para la realización de las evaluaciones se han empleado las aplicaciones web desarrolladas en el capítulo 10. Adicionalmente, en algunos casos se han desarrollado nuevas aplicaciones web, las cuales se centran específicamente en las características objeto de la evaluación.

Las evaluaciones se han llevado a cabo de un modo comparativo, evaluando de forma paralela aplicaciones desarrolladas con otras tecnologías pero dotadas de una funcionalidad equivalente a la aplicación objeto de la evaluación.

Dada la amplitud y multitud de características del modelo desarrollado se ha dividido el proceso de evaluación en cuatro secciones:

- **Rendimiento**: en este apartado se analiza el rendimiento de las aplicaciones, para ello se monitoriza el consumo de los recursos más significativos que condicionan el rendimiento de los dispositivos móviles.

- **Experiencia de usuario**: en este apartado se analiza el comportamiento y la satisfacción de los usuarios en el uso de las aplicaciones web que utilizan la especificación propuesta.

- **Escenarios críticos**: en este apartado se analizan los aspectos más relevantes que condicionan la idoneidad de las aplicaciones para su uso en entornos basados en la interacción ocasional o puntual con objetos físicos.

- **Implementación**: en este apartado se analizan aspectos relacionados con la complejidad de implementación y configuración de las aplicaciones que requieren la gestión de los elementos hardware del dispositivo.

11.2 RENDIMIENTO

En este apartado se analizarán varios de los aspectos que condicionan el rendimiento de las aplicaciones móviles. Para realizar las evaluaciones, en primer lugar se específica un caso de uso habitual de una aplicación, este caso de uso se repite en varias ocasiones. Paralelamente se monitorizará el consumo de los recursos del dispositivo que condicionan el rendimiento: uso de CPU, uso de memoria y tráfico de red.

Los tipos de aplicaciones que los usuarios ejecutan de manera más habitual en sus dispositivos móviles son las aplicaciones nativas y las aplicaciones web. Para abarcar de manera individual cada uno de estos tipos, esta evaluación del rendimiento se divide en dos partes, donde se comparan el rendimiento de las aplicaciones web que utilizan la especificación propuesta con: **1) Aplicaciones nativas** y **2) Aplicaciones web.**

Para realizar las evaluaciones que se presentan a continuación se ha utilizado una implementación del navegador web sensible al contexto para la plataforma Android, esta versión del navegador ha sido desarrollada siguiendo fielmente la especificación propuesta en esta tesis.

El dispositivo utilizado para realizar las pruebas ha sido un HTC Aria Smartphone con las siguientes características técnicas:

- Sistema operativo: Android OS 2.2
- Chipset: Qualcomm MSM7227
- CPU: 600 MHz ARM 11
- Sensores: Acelerómetro, proximidad y brújula
- Navegador: HTML
- GPS: A-GPS
- Red 3G: HSDPA 850 / 1900, HSDPA 900 / 2100
- Memoria: 512 MB ROM, 384 MB RAM
- WLAN: Wi-Fi 802.11 b/g
- Bluetooth: v2.1 con A2DP

Para monitorizar el consumo de recursos durante la ejecución de las aplicaciones se han empleado tres herramientas de análisis:

- Traffic Monitor
- Advance Task Manager
- TrafficStarts App / Task Manager.

11.2.1 Aplicaciones nativas

Esta parte de la evaluación se centra en la comparación del rendimiento de las aplicaciones web que utilizan la propuesta frente a las aplicaciones nativas Android. Adicionalmente, en esta evaluación se ha incluido una aplicación web simple con el fin de permitir interpretar de un mejor modo los resultados obtenidos.

Las aplicaciones nativas están específicamente concebidas y desarrolladas para su uso en una plataforma móvil específica, por ello en la inmensa mayoría de los casos este tipo de

aplicaciones presentarán el mejor rendimiento posible. Existen plataformas capaces de generar aplicaciones nativas a través de diversos procesos, como por ejemplo PhoneGap. Aunque las aplicaciones generadas por estas plataformas se ejecutan en los teléfonos móviles como aplicaciones nativas, presentan un rendimiento más pobre que las aplicaciones nativas desarrolladas con los entornos de desarrollo del fabricante. Para conseguir dotar del mejor rendimiento posible a la aplicación nativa que se utilizará en las evaluaciones se implementará utilizado el SDK de Android.

Funcionalidad de la aplicación:

Se ha diseñado una aplicación capaz de buscar el mejor precio de un determinado producto en los supermercados cercanos al usuario. La aplicación necesita el identificador o nombre del producto que se desea comprar y la ubicación del usuario. Estos datos son enviados a un servicio web externo para recuperar la lista de precios del producto en los supermercados cercanos.

Las aplicaciones que forman parte del proceso de evaluación son:

1. **Aplicación nativa Android:** desarrollada utilizando el SDK de Android y Java.

2. **Aplicación web + etiquetas XML de contexto:** desarrollada utilizando PHP, HTML y etiquetas XML de contexto.

3. **Aplicación web:** desarrollada utilizando PHP y HTML.

La funcionalidad de las tres aplicaciones es equivalente, aunque en el caso de la aplicación web clásica (3) no se incluyen los elementos que habilitan la captura de información de contexto, ésta información debe ser introducida de manera manual.

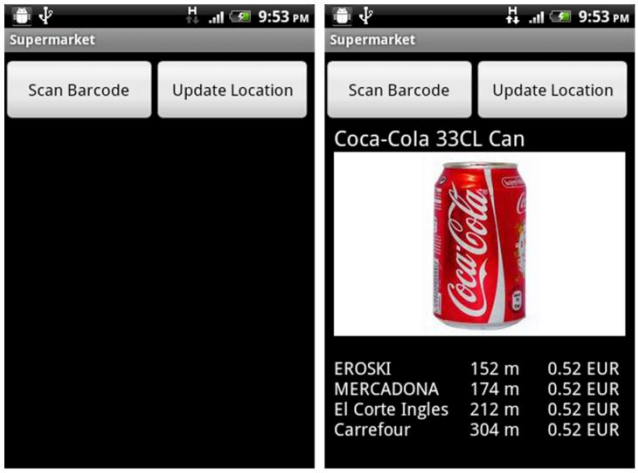

Figura 11.1 – Aplicación nativa Android (1) Antes de escanear el código de barras (2) después de obtener la ubicación y escanear el código de barras.

Caso de uso evaluado:

En primer lugar el usuario debe establecer su ubicación, dependiendo de la aplicación obtendrá su ubicación de manera automática o la introducirá manualmente. Una vez la posición del usuario está determinada, se realizará una búsqueda de un producto. Para

buscar un producto se necesita su identificador o nombre. En el caso de las aplicaciones (1) **Aplicación nativa Android** y (2) **Aplicación web + etiquetas XML de contexto,** se utilizará la cámara de fotos como lector de código de barras para identificar el producto. En la aplicación (3) **Aplicación web** el nombre del producto debe escribirse en un campo de entrada de datos.

El caso de uso se ha repetido un total de 25 veces para cada una de las tres aplicaciones. Las medias de los consumos de recursos obtenidos en las monitorizaciones se presentan en el siguiente conjunto de gráficas.

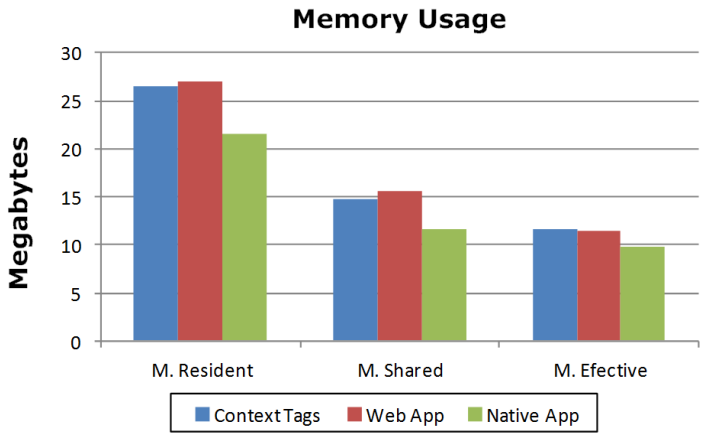

Figura 11.2 – Representación gráfica de la memoria utilizada por el sistema durante la ejecución de las aplicaciones evaluadas.

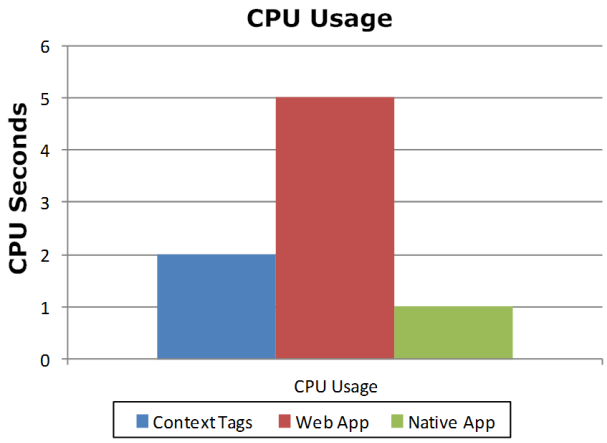

Figura 11.2 – Representación gráfica del consumo de segundos de CPU durante la ejecución de las aplicaciones evaluadas.

A partir del uso de memoria y de los segundos de CPU consumidos es posible evaluar el rendimiento del dispositivo durante la ejecución de la aplicación. Los valores obtenidos muestran que el rendimiento de la aplicación nativa es significativamente mejor. Esta diferencia se debe en parte a que la aplicación nativa ha sido desarrollada de la manera más simple posible, sin apenas interfaces de usuarios y aplicando unos procesos relativamente simples. Seguramente la mejora del interfaz de usuario de la aplicación o de

la lógica de negocio de alguno de sus procesos haría aumentar el gasto de memoria y segundos de CPU. La (2) **Aplicación web + etiquetas XML de contexto** es interpretada por el navegador web sensible al contexto, esta aplicación es relativamente compleja, por eso presenta un consumo de recursos significativamente mayor al de una aplicación nativa simple. La mayor diferencia se encuentra en los tiempos de uso de la CPU, donde la (2) **Aplicación web + etiquetas XML de contexto** consume el doble de los recursos que la aplicación nativa. Sin embargo, aunque el consumo de recursos de la (2) **Aplicación web + etiquetas XML de contexto** pueda parecer alto no lo es en comparación con la aplicación (3) **Aplicación web**.

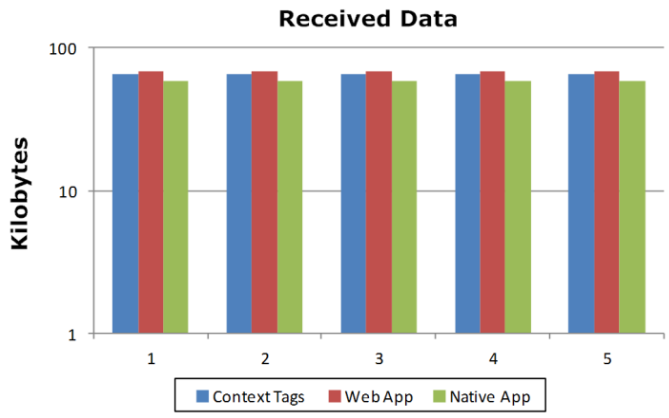

Figura 11.3 – Representación gráfica del tráfico de red recibido durante la ejecución de las aplicaciones evaluadas.

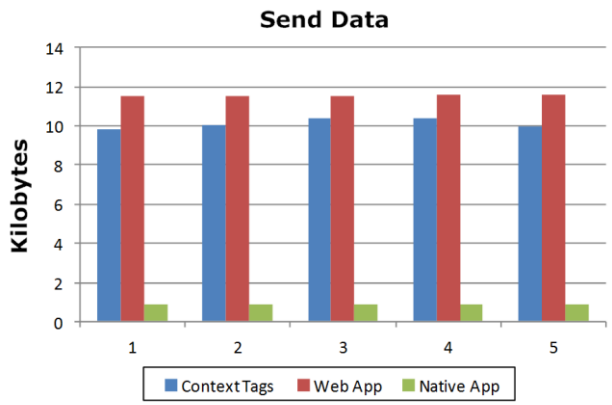

Figura 11.4 – Representación gráfica del tráfico de red saliente durante la ejecución de las aplicaciones evaluadas.

El análisis del tráfico de datos durante los casos de uso confirma que el volumen de datos intercambiado por la aplicación nativa es mucho menor que el de las aplicaciones web; este resultado era esperado. Las aplicaciones web están alojadas en un servidor, como consecuencia se requiere un intercambio de ficheros HTML e imágenes. En cambio, la aplicación nativa únicamente utiliza la red para realizar una consulta a un servidor web y para obtener la información y la imagen del producto.

Comparando los resultados obtenidos en las simulaciones, se puede concluir que resultaría muy difícil que cualquier tipo de aplicación web pudiera ofrecer un rendimiento

similar al de una aplicación nativa. La (2) **Aplicación web + etiquetas XML de contexto** puede obtener rendimientos similares o incluso mejores que los de las aplicaciones web clásicas. Esto se debe a que la interpretación y gestión de los controles HTML que realiza el WebKit es menos eficiente que la de los controles nativos, justamente el tipo de controles que utiliza el navegador web sensible al contexto para representar los requerimientos de información de contexto. Adicionalmente, en muchos casos, el uso de los servicios web para el intercambio de información consigue reducir el tráfico de datos generado por ciertos controles HTML, como formularios.

11.2.2 Aplicaciones web

Esta parte de la evaluación del rendimiento se centra en la comparación de las aplicaciones web que utilizan la especificación propuesta frente a las aplicaciones web tradicionales. El rendimiento de las aplicaciones web está condicionado en gran parte por sus interfaces de usuario, cuando se aumenta el número de controles y clientes ricos suele aumentar el consumo de recursos del dispositivo. Con el fin de obtener una evaluación objetiva y representativa de la diversidad existente, se seleccionarán como elementos comparativos: una aplicación web clásica con un interfaz de usuario muy simple y otra con uno relativamente complejo.

Funcionalidad de la aplicación:

Se ha desarrollado una aplicación web que permite introducir en una galería fotografías geolocalizadas. La aplicación requiere una fotografía y las coordenadas de la ubicación actual para añadir una fotografía a la galería.

1. **Aplicación Web HTML:** desarrollada utilizando PHP y HTML.

2. **Aplicación web + etiquetas XML de contexto:** desarrollada utilizando PHP, HTML y etiquetas XML de contexto.

3. **Aplicación Web HTML + Gmail:** desarrollada utilizando PHP, HTML, Javascript y el API de Google Maps.

La funcionalidad de las tres aplicaciones es equivalente, aunque en el caso de la (2) **Aplicación web + etiquetas XML de contexto** (3) se aplicarán las etiquetas XML de contexto para realizar la captura de la fotografía y obtener la ubicación actual. En las otras dos aplicaciones el envío de la fotografía y el establecimiento de la ubicación actual se realizarán de manera manual, mediante la utilización de formularios HTML. Además, la (3) **Aplicación Web HTML + Gmail** permitirá establecer la ubicación exacta de la fotografía seleccionando la posición en un mapa Google Maps.

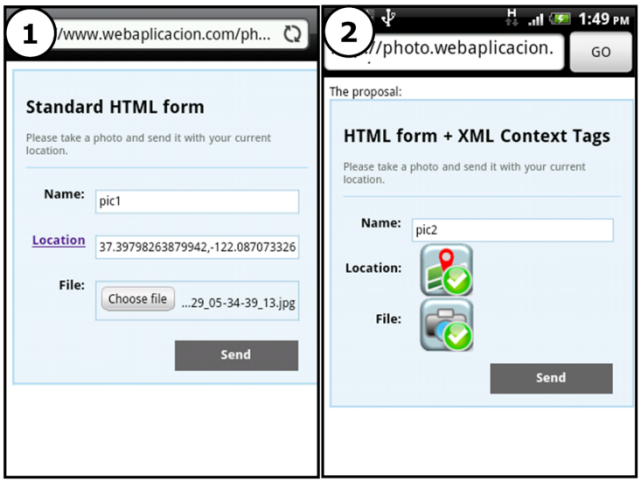

Figura 11.5 – (1) La aplicación web que utiliza únicamente HTML. (2) La aplicación web que incluye etiquetas XML de contexto.

Caso de uso evaluado:

En primer lugar el usuario debe tomar una fotografía, asignarle un nombre y añadir las coordenadas longitud y latitud de su ubicación actual. Cuando el formulario esté completo el usuario debe pulsar el botón enviar para que los datos se transmitan al servidor y puedan ser almacenados en la galería.

El caso de uso se ha repetido un total de 25 veces para cada una de las tres aplicaciones. La media de los consumos de recursos obtenidos en las monitorizaciones se presenta en el siguiente conjunto de gráficas.

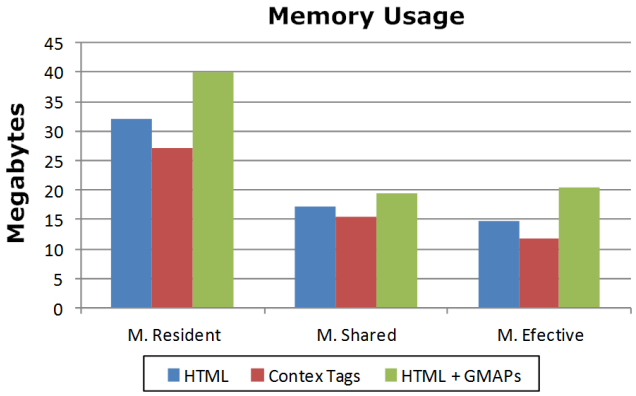

Figura 11.6 – Representación gráfica del consumo de memoria del sistema durante la ejecución de las aplicaciones evaluadas.

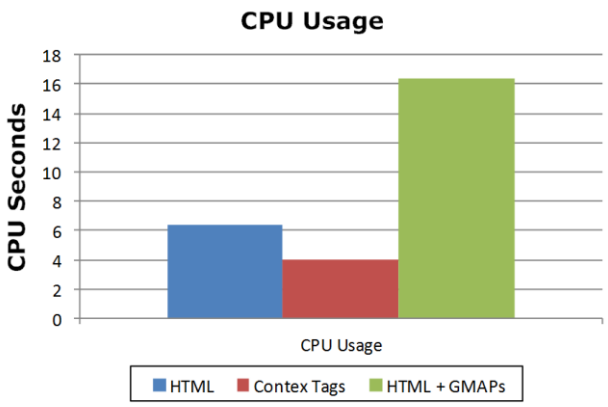

Figura 11.7 – Representación gráfica del consumo de CPU durante la ejecución de las aplicaciones evaluadas.

La evaluación del rendimiento expresada en consumo de memoria y segundos de CPU presenta unos resultados muy similares en la (1) **Aplicación Web HTML y** (2) **Aplicación web + etiquetas XML de contexto,** aunque en todos los casos la aplicación web que utiliza las etiquetas XML de contexto ha obtenido un rendimiento ligeramente mejor. Como ya se explicó en la evaluación anterior, esta diferencia se debe en parte a que la interpretación y gestión de los controles HTML que realiza el WebKit es menos eficiente que la de los controles nativos que realiza el navegador web sensible al contexto. Como era de esperar, la utilización de los interfaces ricos dispara el consumo de recursos en el dispositivo.

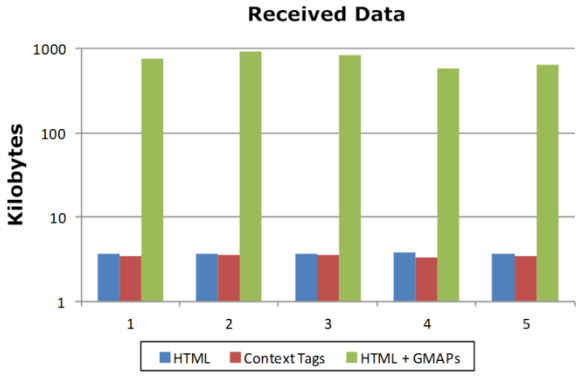

Figura 11.8 – Representación gráfica del tráfico de red recibido durante la ejecución de las aplicaciones evaluadas.

Figura 11.9 – Representación gráfica del tráfico de red recibido durante la ejecución de las aplicaciones (1) HTML y (2) HTML + Etiquetas XML de contexto.

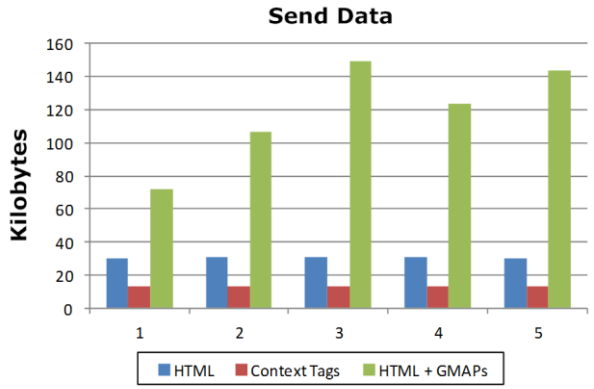

Figura 11.10 – Representación gráfica del tráfico de red saliente durante la ejecución de las aplicaciones evaluadas.

El análisis del tráfico de red durante la ejecución de los casos de uso ofrece valores similares para las aplicaciones: (1) **Aplicación Web HTML y** (2) **Aplicación web + etiquetas XML de contexto,** aunque en todos los casos la aplicación que utilizaba las etiquetas XML de contexto ha requerido de un intercambio de datos menor. Esta diferencia se ha acentuado especialmente en el envío de datos, donde la (2) **Aplicación web + etiquetas XML de contexto** consiguió reducir el tráfico de datos en más de un 50%. Este comportamiento se debe a que en muchos casos el uso de los servicios web para el intercambio de información consigue reducir el tráfico de datos generado por ciertos controles HTML, como formularios. La segunda parte de esta evaluación se centrara en el análisis del proceso de envío de datos.

Proceso de envío de datos

En esta segunda evaluación se analizarán dos aspectos de suma importancia:

1) El tráfico de red generado exclusivamente por el envío de información de contexto.

2) El tiempo que transcurre desde que se inicia el envío de la información hasta que el usuario percibe la respuesta.

Para realizar esta evaluación se han utilizado veinticinco aplicaciones divididas en tres grupos. Cada una de las aplicaciones web envía datos a un servidor utilizando 1) un formulario HTML, 2) varias etiquetas XML de contexto pertenecientes a un mismo grupo, 3) varias etiquetas XML de contexto individuales. El número de campos enviado se ha ido incrementando progresivamente desde 1 hasta 5.

Por ejemplo para n = 3:

1) Es una aplicación web con un formulario HTML que contiene 3 campos de entrada.

2) Es una aplicación web con 3 etiquetas XML de contexto "<audiocapture speech='true'/>" individuales. Las etiquetas XML de contexto individuales realizan el envío de datos en el mismo momento que finaliza su tarea programada de contexto. Si la página utiliza 3 etiquetas individuales, cada una realizará su propio envío de datos.

3) Es una aplicación web con 3 etiquetas XML de contexto "<audiocapture speech='true' group='info'/>" que forman parte de un mismo grupo. La agrupación de las etiquetas permite modelar un comportamiento similar al de un formulario HTML. Hasta que la información de todas las etiquetas no haya sido capturada no se realiza el envío de datos, en este caso el envío será único y contendrá la información correspondiente a todas las etiquetas.

Cada etiqueta ***<audioCaptureSpeech>*** se utilizará para enviar exactamente la misma cadena de texto que se introducirá en los campos de texto del formulario. A continuación se muestra de forma gráfica los resultados de la evaluación realizada.

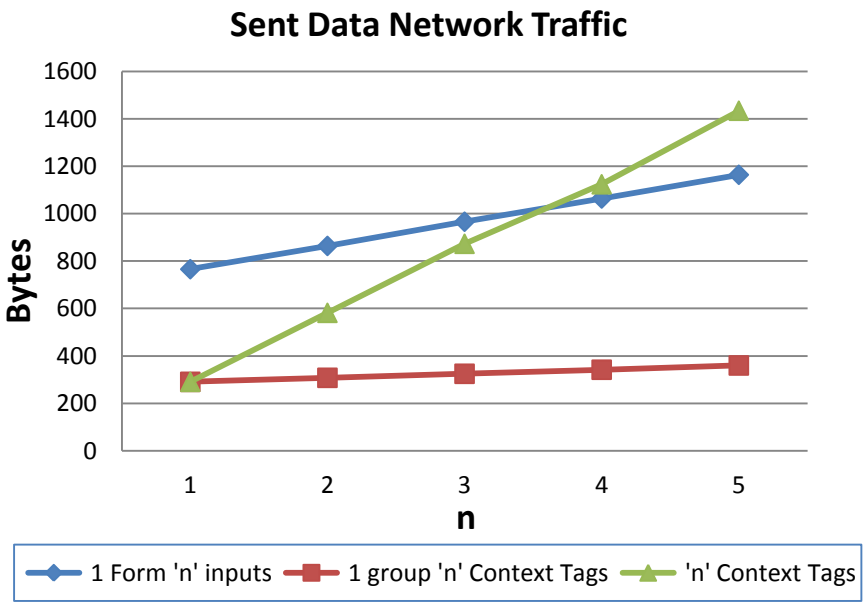

Figura 11.11 – La gráfica muestra el volumen de datos transferido para tres enfoques diferentes.

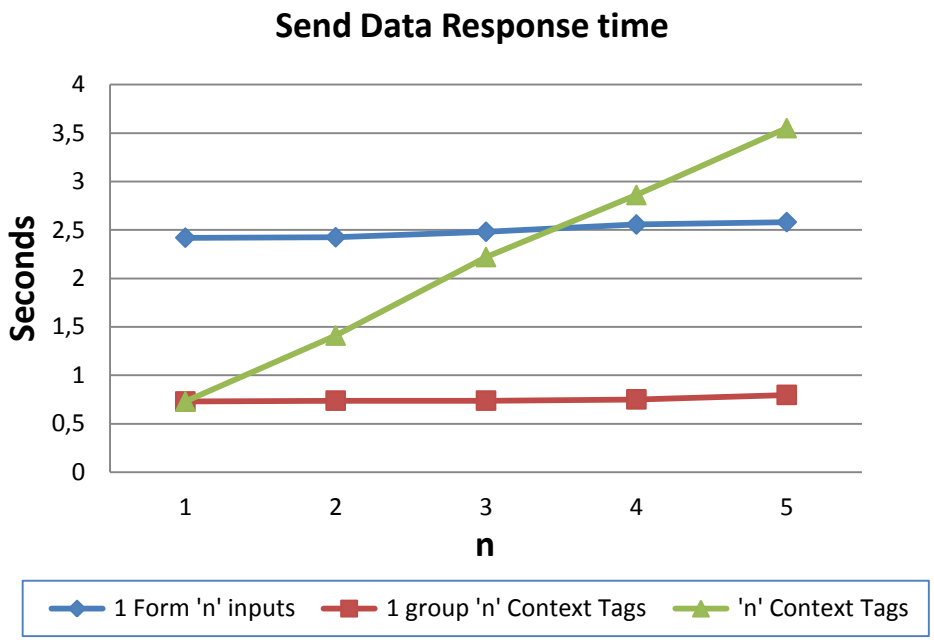

Figura 11.12 – El gráfico muestra los tiempos de espera para tres enfoques diferentes, estos tiempos comprenden desde que el usuario envía la información hasta que éste percibe la respuesta de la aplicación.

Comparando los resultados obtenidos se puede llegar a la siguiente conclusión: en algunos escenarios el uso de las etiquetas XML de contexto puede ayudar a reducir significativamente tanto el tráfico de red como los tiempos de respuesta. La utilización de un grupo de etiquetas XML de contexto resulta significativamente más eficiente que la de un formulario HTML tradicional, incluso la utilización de dos grupos de etiquetas continuaría siendo más eficiente que la utilización de un formulario HTML. Cabe señalar que en algunas ocasiones una etiqueta XML de contexto podría reemplazar a varios campos de entrada de un formulario, por ejemplo la lectura de un código QR podría facilitar de manera automática el envío de varias cadenas de datos que de otro modo deberían ser introducidas individualmente en los campos de un formulario HTML.

11.3 EXPERIENCIA DE USUARIO

Para la siguiente evaluación se ha contado con la participación de 20 usuarios habituales de Smartphones. Los usuarios participantes poseen teléfonos móviles inteligentes y están acostumbrados tanto a manejar aplicaciones nativas como aplicaciones web desde sus dispositivos.

Para realizar esta evaluación se han empleado dos aplicaciones que habían sido anteriormente utilizadas en el apartado **11.2.1 Aplicaciones nativas**. Las aplicaciones permiten buscar el mejor precio de un determinado producto en los supermercados cercanos al usuario. Para esta labor la aplicación requiere el identificador o nombre del producto y la ubicación actual del usuario. Ambos datos se envían a un servicio web externo que retorna la lista de supermercados cercanos que venden ese producto.

Las aplicaciones que forman parte del proceso de evaluación son:

1. **Aplicación web + etiquetas XML de contexto:** desarrollada utilizando PHP, HTML y etiquetas XML de contexto.

2. **Aplicación web:** desarrollada utilizando PHP y HTML.

La funcionalidad de las dos aplicaciones es equivalente, aunque en el caso de la (2) **Aplicación web** no se incluyen los elementos que habilitan la captura de información de contexto, esta información debe ser introducida de manera manual.

Caso de uso

La tarea asignada a los usuarios fue la búsqueda de un determinado producto en ambas aplicaciones. Mientras los usuarios realizaban la tarea se realizaron las siguientes mediciones:

- Tiempo total consumido.

- Número de pulsaciones en pantalla.

- Número de errores.

Después de finalizar las dos tareas, se les pidió a los usuarios que valorasen con un número comprendido entre 0 y 10, su grado de satisfacción para ambas aplicaciones (0 – Nada satisfecho, 10 – Totalmente satisfecho).

La media de los resultados y valoraciones obtenidas se muestran de manera simplificada en las siguientes gráficas.

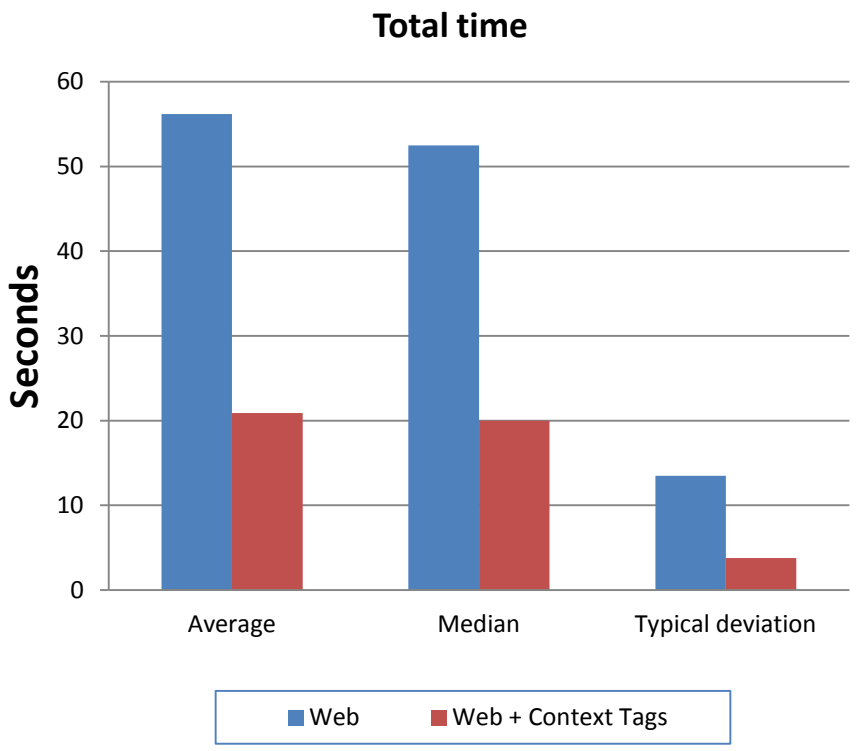

Figura 11.13 – Representación gráfica de los tiempos consumidos por los usuarios en la realización de las acciones correspondientes al caso de uso.

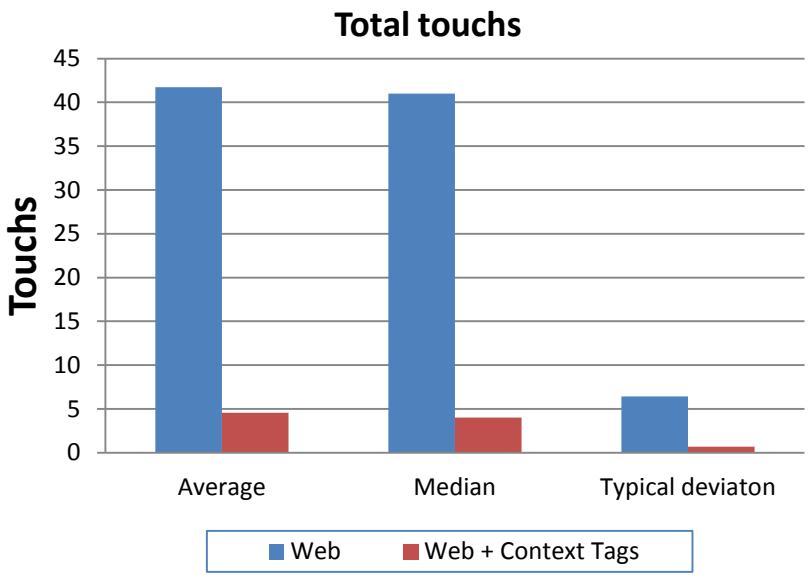

Figura 11.14 – Representación gráfica del número de pulsaciones de pantalla realizadas por los usuarios en la ejecución de las acciones correspondientes al caso de uso.

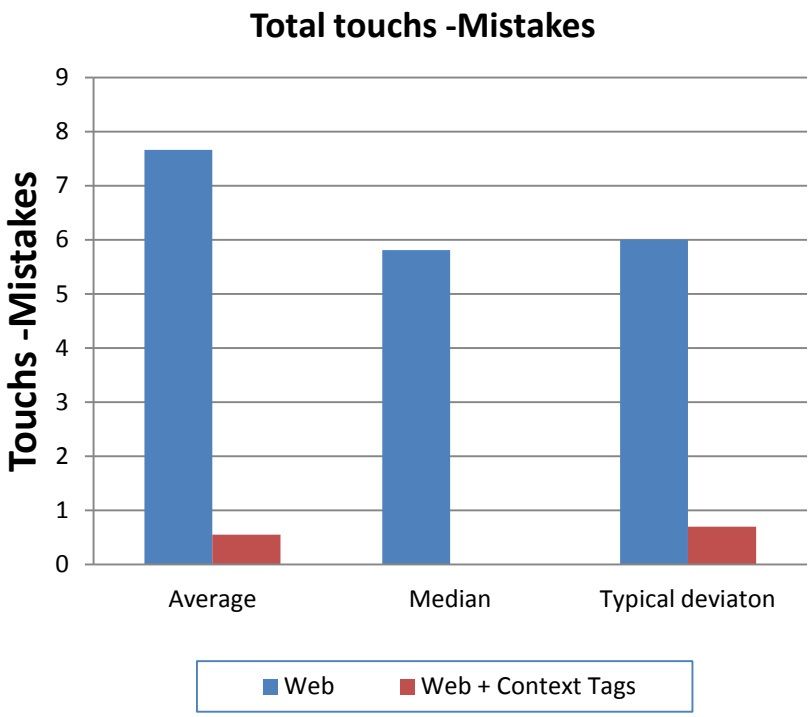

Figura 11.15 – Representación gráfica del número de errores cometidos en las pulsaciones de pantalla realizadas por los usuarios en la ejecución de las acciones correspondientes al caso de uso.

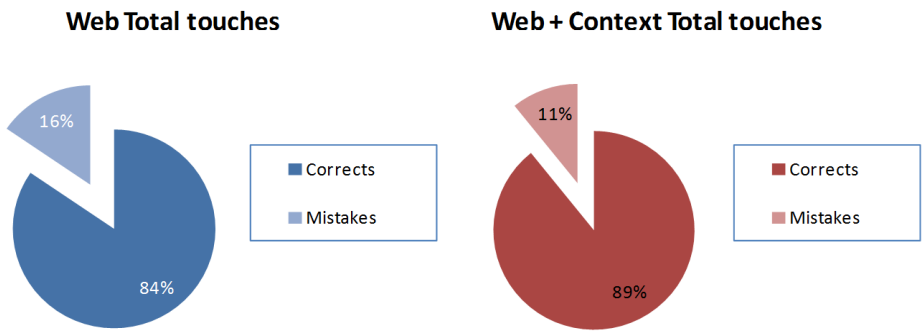

Figura 11.16 – Comparación estadística entre el porcentaje de pulsaciones de pantalla erróneas y correctas realizadas por los usuarios en la ejecución de las acciones correspondientes al caso de uso.

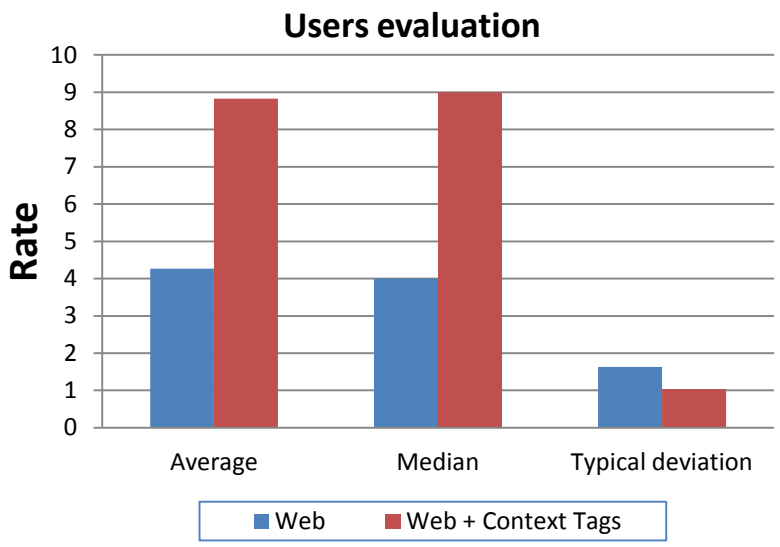

Figura 11.17 – Nota asignada por los usuarios a las aplicaciones empleadas en la evaluación.

Aunque los resultados de esta evaluación se centran en una aplicación concreta, se pueden extraer algunas conclusiones. En varios escenarios la utilización de las etiquetas XML de contexto en las aplicaciones web pueden aportar beneficios muy significativos:

- **Reducir el tiempo empleado en la entrada de datos**, en este caso concreto el tiempo se reduce en más de un 60% y el proceso de interacción requiere 10 veces menos pulsaciones de pantalla que antes.

- **Disminuye la tendencia a cometer errores**. Ya que el número de pulsaciones de pantalla se reduce considerablemente con la utilización de las etiquetas XML de contexto, es lógico que también se reduzca el número total de errores. Según el análisis el número de errores no solo descendió de manera absoluta, sino también de manera relativa en un 5%.

- **Mejorar la satisfacción del usuario**. En este caso, el 100% de los usuarios han otorgado una mejor valoración a la aplicación web que utilizaba etiquetas XML de contexto, consiguiendo una valoración media de 8,83 frente al 4,27 de la aplicación web tradicional.

Aunque esta evaluación no abarca la totalidad de los casos existentes, ya que la posible utilización de las etiquetas XML de contexto depende en gran parte del objetivo y los datos que maneja la aplicación, se puede confirmar que en determinadas aplicaciones web la tecnología propuesta es capaz de simplificar notablemente los procesos de entrada de datos y de mejorar significativamente el grado de satisfacción de los usuarios.

11.4 ESCENARIOS CRÍTICOS

Uno de los objetivos de esta propuesta es su utilización en entornos que se basan en la interacción ocasional o puntual con objetos físicos. Para que las aplicaciones se puedan adaptar de forma óptima a este tipo de entornos, estas deben consumir el mínimo tiempo posible en las tareas auxiliares distintas al propio proceso de interacción con el objeto físico o dispositivo electrónico. Dentro de estas tareas auxiliares se encontrarían los procesos de instalación, acceso a la aplicación y otros tiempos de carga.

En esta evaluación se va a obtener una estimación de la reducción de tiempos que puede lograrse con la utilización de las aplicaciones web respecto al uso de las aplicaciones nativas. Para ello se compararán los tiempos consumidos y el tráfico de datos generado por aplicaciones web y aplicaciones nativas en varios escenarios que implican el acceso a una aplicación por primera vez.

Los escenarios que forman parte de la evaluación son los siguientes:

1. **Aplicación nativa, Búsqueda por nombre, descarga e instalación desde Google Play**. Este escenario representa un caso típico donde un usuario debe introducir el nombre de la aplicación en la pestaña de búsqueda de la tienda de aplicaciones Google Play.

2. **Aplicación web, Introducción manual de una URL y carga de la aplicación desde el navegador web sensible al contexto**. Este escenario representa un caso típico donde un usuario debe introducir la URL de la aplicación web en el navegador.

3. **Aplicación nativa, Acceso directo a la página de descarga de la aplicación, descarga e instalación desde Google Play**. Este escenario representa un caso típico en el que al usuario se le facilita el enlace a la aplicación en la tienda de aplicaciones Google Play. El enlace puede ser facilitado a través de un mensaje de texto, un email, un código QR, etiqueta NFC, etc.

4. **Aplicación nativa, Acceso directo a una URL con el instalable de la aplicación, descarga e instalación desde el gestor de aplicaciones**. Este escenario representa un caso típico en el que al usuario se le facilita la URL donde se aloja un fichero ejecutable de una aplicación nativa. La URL puede estar en la pestaña de favoritos, haber sido enviada a través de un mensaje de texto, un email, un código QR, etiqueta NFC, etc.

5. **Aplicación web, Introducción manual de una URL y carga de la aplicación desde el navegador web sensible al contexto**. Este escenario representa un caso típico en el que al usuario se le facilita la URL de la aplicación web. La URL puede estar en la pestaña de favoritos, haber sido enviada a través de un mensaje de texto, un email, un código QR, etiqueta NFC, etc.

El tiempo medio obtenido en cada uno de los diferentes escenarios se muestra en la siguiente representación gráfica:

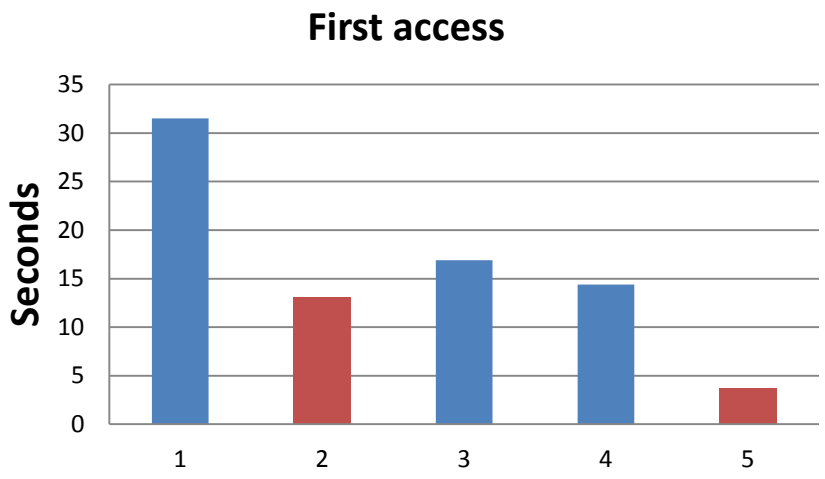

Figura 11.18 – Tiempo medio consumido en el primer uso de las aplicaciones en los 5 escenarios evaluados.

El consumo de tráfico de datos medio obtenido en los diferentes escenarios se muestra en la siguiente representación gráfica:

Figura 11.19 – Tráfico de datos medio consumido en el primer uso de las aplicaciones en los 5 escenarios evaluados.

Cabe destacar que en los escenarios seleccionados se está evaluando una de las situaciones más beneficiosas para las aplicaciones nativas. La aplicación a descargar e instalar tiene un tamaño mínimo, significativamente menor que el de la inmensa mayoría de las aplicaciones distribuidas, tan solo 149kb; con la correspondiente reducción en los tiempos de descarga e instalación que ello implica. La aplicación utilizada en las evaluaciones ha sido desarrollada utilizando el SDK de Android. Por ello, se trata de una aplicación nativa más rápida y eficiente que las que puedan desarrollarse en el resto de plataformas alternativas.

Según los escenarios evaluados, si un usuario desea interactuar con un objeto físico por primera vez, la utilización de una aplicación web es un 68% - 75% más rápida que la de

una aplicación nativa. Como es lógico el tráfico de datos también se redujo considerablemente 89% - 95%, ya que en el caso de las aplicaciones web la descarga de datos se limita a la parte de la aplicación necesaria en lugar del global de la aplicación.

Los resultados obtenidos en la evaluación muestran que las aplicaciones web que utilizan etiquetas XML de contexto pueden adaptarse de una forma mucho más eficiente a los entornos que se basan en la interacción ocasional con objetos físicos. Incluso en un caso muy favorable para la utilización de las aplicaciones nativas, donde el tamaño del ejecutable tiene un tamaño muy reducido, la aplicación web ha conseguido tiempos notablemente más bajos.

11.5 IMPLEMENTACIÓN

En esta sección se evaluarán los mecanismos que ofrecen las plataformas móviles para gestionar los elementos hardware que permiten capturar la información de contexto y realizar las labores de comunicación con otros dispositivos cercanos. La evaluación realizada consiste en medir entre otras cosas el número de líneas de código que se requieren por término medio para realizar una misma acción en distintas plataformas. Aunque el criterio de medición seleccionado presenta deficiencias, los resultados de la evaluación pueden servir para sacar a la luz diferencias significativas entre las plataformas. Además de la cantidad de líneas de código, existen muchos otros factores que pueden utilizarse para estimar el coste o complejidad del desarrollo.

En esta evaluación se va a comparar la implementación de los segmentos de la aplicación que tienen como objetivo acceder a elementos hardware del dispositivo. Se han evaluado aplicaciones desarrolladas con las siguientes tecnologías:

- **Java-Android**: utilizando las librerías de Android se puede gestionar la mayor parte de los elementos hardware del dispositivo móvil, dependiendo del tipo de componente hardware, esta gestión se suele realizar con un nivel más bajo o más alto de abstracción.

- **Javascript - DeviceOrientation Event *Specification:*** esta especificación ofrece un API Javascript de alto nivel de abstracción que permite obtener los valores registrados por los sensores: acelerómetro, giroscopio y compás del dispositivo cliente.
  Javascript – GeolocationAPI: esta especificación ofrece un API Javascript de alto nivel de abstracción que permite obtener la ubicación actual del dispositivo cliente.
  Ambas especificaciones han sido incluidas en varios navegadores móviles durante el año 2011.

- **Javascript - PhoneGap**: es una de las plataformas de aplicaciones centradas en la web más populares y maduras, permite desarrollar aplicaciones móviles nativas empleando tecnologías web como: HTML5, CSS y Javascript. PhoneGap utiliza un conjunto de librerías propias que soportan de manera parcial o total el acceso a gran parte de los componentes hardware del dispositivo móvil.

- **Objective-C iOS**: los dispositivos móviles de Apple se sirven de un conjunto de frameworks propios y del lenguaje de programación orientado a objetos Objetive-C para habilitar la gestión de los elementos hardware del dispositivo. La gestión de los elementos hardware se lleva a cabo con un gran nivel de detalle. En varios aspectos presenta muchas similitudes con la plataforma móvil Android.

- **PHP + Etiquetas de contexto y comunicación**. La especificación propuesta permite combinar las tecnologías web tradicionales con un conjunto de etiquetas XML especiales. Estas etiquetas XML permiten especificar de forma simple requerimientos que implican la gestión de los elementos hardware del dispositivo.

El dispositivo cliente envía las respuestas correspondientes a los requerimientos al servidor utilizando servicios web.

Para realizar esta evaluación se han seleccionado 5 tareas básicas que emplean elementos hardware del dispositivo. El objetivo es conseguir un escenario de comparación que no otorgue ventaja a la especificación propuesta, debido a que la implementación de funcionalidades similares a las presentadas en las tareas complejas puede requerir procesos de implementación muy costosos en varias de las plataformas seleccionadas para este análisis.

Las tareas seleccionadas son las siguientes:

- Sensor de aceleración: obtener aceleración actual.
- Chip GPS: obtener ubicación actual.
- Cámara de fotos: tomar fotografía.
- Bluetooth: emparejamiento con otro dispositivo más envío de un dato.
- NFC: lectura de etiqueta NFC.

Por cada una de las cinco tecnologías seleccionadas [(1) Java-Android, (2) Javascript - DevOrientation – GeolocationAPI, (3) Javascript – PhoneGap, (4) Objective-C iOS, (5) PHP Etiquetas de contexto] se han implementado varias aplicaciones, cada aplicación contiene una funcionalidad básica que implica la gestión de un elemento hardware del dispositivo [(1) Sensor de aceleración, (2) Chip GPS, (3) Cámara de fotos, (4) Bluetooth y (5) NFC]. No se han implementado aplicaciones para los cinco tipos de elementos hardware seleccionados en todas las tecnologías, ya que algunas de ellas presentaban incompatibilidades técnicas.

Las aplicaciones desarrolladas cuentan con una implementación mínima que hace posible el cumplimiento de la tarea encomendada en cada caso. Los aspectos medidos en el código fuente y estructura de aplicaciones son los siguientes:

- L = número de líneas de código añadidas a la aplicación para dotarla de la funcionalidad específica requerida en cada caso.
- T = Tamaño: número de caracteres, con espacios incluidos.
- E = número de elementos específicos (clases, etiquetas, objetos, etc) que se requieren para la gestión de ese elemento hardware.
- F = número de funciones de configuración requeridas adicionales a la implementación, como puede ser: la asignación de permisos, adjuntar librerías externas, etc.

Los resultados obtenidos se resumen en la siguiente tabla:

	Obtener Aceleración	Obtener ubicación	Tomar Fotografía	Pair + Envío Bluetooth	Lectura NFC
Java-Android	14L, 468T 4E, 0F	51L, 1453T 3E, 1F	75L, 1946T 1E, 1F	204L, 6711T 7E, 1F	40L, 1044T 6E, 1F
Javascript - DevOrientation	11L, 381T 2E, 0F	---	---	---	---
Javascript – GeolocationAPI	---	15L, 357T 3E, 0F	---	---	---
Javascript – PhoneGap	9L, 210T 2E, 0F	16L, 401T 2E, 1F	6L, 212T 1E, 1F	*	* Plugin 22L, 482T 2E, 2F
Objective-C iOS	13L, 446T 4E, 0F	22L, 819T 2E, 0F	14L, 633T 2E, 0F	76L, 1788T 3E, 0F	---
PHP Etiquetas XML de contexto	11L, 202T 1E, 0F	11L, 208T 1E, 0F	13L, 228T 1E, 0F	2L, 133T 1E, 0F	11L, 205T 1E, 0F

Tabla 11.1 – Comparación parcial entre implementaciones.

Con el fin de poder establecer un mecanismo de comparación global que sirva para estimar la complejidad que implica el desarrollo se le ha asignado un peso a cada uno de los aspectos medidos.

- L = 3
- T = 1
- E = 10
- F = 20

La siguiente tabla muestra el resultado global de los aspectos medidos según los pesos asignados:

	Obtener Aceleración	Obtener ubicación	Tomar Fotografía	Pair + Envío Bluetooth	Lectura NFC
Java-Android	**550**	**1656**	**2201**	**7413**	**1244**
Javascript - DevOrientation	**434**	**---**	**---**	**---**	**---**
Javascript – GeolocationAPI	**---**	**432**	**---**	**---**	**---**
Javascript – PhoneGap	**257**	**489**	**260**	**---**	**608**
Objective-C iOS	**525**	**905**	**695**	**2046**	**---**
PHP Etiquetas XML de contexto	**245**	**251**	**277**	**149**	**248**

Tabla 21.2 – Comparación global entre implementaciones.

Basándonos exclusivamente en los aspectos analizados y los criterios tenidos en cuenta para realizar este análisis, podemos establecer una serie de conclusiones:

- La gestión de los elementos hardware del dispositivo mediante la utilización de etiquetas XML de contexto puede ser significativamente más ágil que el de las aplicaciones nativas **Java-Android** y **Objetice-C iOs**.

- Los procesos de desarrollo de aplicaciones basadas en tecnologías web (o centradas en la web) como **Javascript – DevOrientation, Javascript GeolocationAPI** y **Javascript – PhoneGap** presentan una complejidad relativamente similar al proceso de desarrollo de las aplicaciones web que utilizan etiquetas XML de contexto.

Aunque las evaluaciones realizadas no son completamente precisas, debido a que en ellas intervienen factores variables y dependientes de la codificación, se puede concluir que las aplicaciones web que utilizan etiquetas XML de contexto presentan un proceso de desarrollo relativamente rápido y posiblemente más sencillo que el del resto de enfoques analizados.

PARTE V

CONCLUSIONES

CAPÍTULO 12

CONCLUSIONES Y TRABAJO FUTURO

12.1 SÍNTESIS DEL TRABAJO DESARROLLADO

Esta tesis propone un modelo para el desarrollo de aplicaciones móviles multiplataforma, capaces de gestionar los elementos hardware de los dispositivos que permiten la comunicación con otros dispositivos cercanos y la captura de diferentes tipos de información de contexto. Las características de este modelo lo hacen óptimo para el desarrollo de objetos virtuales basados en tecnologías web. Además de este propósito general, el modelo propuesto ha de cumplir otra serie de objetivos concretos que han sido detallados en el capítulo 1.

A lo largo de los capítulos 3 a 6 se ha realizado una descripción detallada de todas las actividades llevadas a cabo durante la realización de esta tesis doctoral. Se comenzó exponiendo y analizando el ámbito de trabajo en el que se enmarca esta tesis y, posteriormente, se realizó un análisis de los sistemas y contribuciones más relevantes relacionadas con los objetivos fijados al inicio de esta tesis doctoral.

El modelo propuesto partió de una arquitectura inicial que se presentó en el capítulo 7 "Objetos virtuales v1.0 STDD". La propuesta definía una estructura específica que permitía desarrollar aplicaciones móviles multiplataforma con ciertas limitaciones. Estas aplicaciones encapsulaban la lógica de negocio y presentación de la aplicación. Las aplicaciones basadas en la propuesta debían ser ejecutadas por un sistema gestor instalado en el propio teléfono móvil, este sistema permitía la ejecución local o remota de las aplicaciones. Con el fin de cumplir de una forma más óptima los objetivos de esta tesis y adaptarse a las recomendaciones realizadas por los revisores científicos, se realizó una profunda revisión del modelo propuesto. La segunda versión del modelo presentada en el capítulo 8 "Objetos virtuales v2.0 distribuidos" avanzaba en el cumplimiento de los objetivos de esta tesis, a la vez que eliminaba las potenciales deficiencias de seguridad detectadas en el modelo anterior. Este modelo transformaba la estructura propuesta inicialmente en un modelo de aplicaciones distribuidas, las cuales eran ejecutadas en un servidor de manera remota. El dispositivo móvil cliente era el encargado de gestionar y presentar al usuario las aplicaciones que se ejecutaban en el servidor. Adicionalmente, el dispositivo cliente debía estar preparado para satisfacer los requerimientos que implicaban el uso de los elementos hardware propios, dado que estos requerimientos no podían completarse en la parte del servidor. Para este propósito, la versión 2 de la propuesta introduce el uso de una especificación de etiquetas XML, estas etiquetas se incluyen en los interfaces de la aplicación para expresar al cliente los requerimientos concretos que implican la utilización de alguno de sus elementos hardware.

Finalmente, la versión 3 del modelo propuesto presentado en el capítulo 9 "Objetos virtuales v3.0 En la web" introduce un nuevo modelo que integra las tecnologías web existentes con una nueva especificación concebida a partir de la versión anterior de la propuesta. Esta nueva especificación está especialmente diseñada para permitir la gestión ágil y simple de los elementos hardware del dispositivo cliente (Fig.12.1). De esta forma, el alcance del nuevo modelo se centra exclusivamente en el foco de la deficiencia de las aplicaciones web: la dificultad para gestionar los elementos hardware del dispositivo. Al contrario que la versión anterior, esta nueva especificación evita definir elementos típicos de las aplicaciones tales como las interfaces de usuario, estructura, ficheros y todos aquellos elementos que pueden ser construidos a partir de las tecnologías web existentes. Ofreciendo la posibilidad de desarrollar objetos virtuales combinando el uso tecnologías web tradicionales con parte de la especificación propuesta. Esta especificación se divide en dos ámbitos:

- El primer ámbito se centra en la parte del servidor, se ha especificado un conjunto abierto de etiquetas XML que sirven para expresar requerimientos de comunicación o de información de contexto, estas etiquetas XML se introducen en las interfaces de las aplicaciones web.

- El segundo ámbito se centra en la parte del cliente, se ha presentado una especificación para la construcción de un navegador web sensible al contexto. Este navegador es capaz de satisfacer los requerimientos de comunicación y de información de contexto expresados en las etiquetas XML. El navegador web sensible al contexto interpreta las etiquetas XML específicas y ejecuta las tareas programadas asociadas a las mismas. En la mayoría de ocasiones estas tareas gestionan los elementos hardware del dispositivo, con el fin de realizar tareas de comunicación o capturar información de contexto, el resultado de la tarea se remite al servidor, para que éste pueda incluir la respuesta en sus procesos de negocio.

La versión final de la especificación propuesta se ha utilizado para implementar un navegador web sensible al contexto y varias aplicaciones web con diferentes funcionalidades. El proceso de desarrollo de estas aplicaciones web se detalla en el capítulo 10. Posteriormente, parte de las aplicaciones mostradas en el capítulo fueron sometidas a diversos tipos de evaluación. Estas evaluaciones incluían estudios sobre el rendimiento, la experiencia de usuario, la capacidad de adaptación a entornos específicos y la complejidad de implementación.

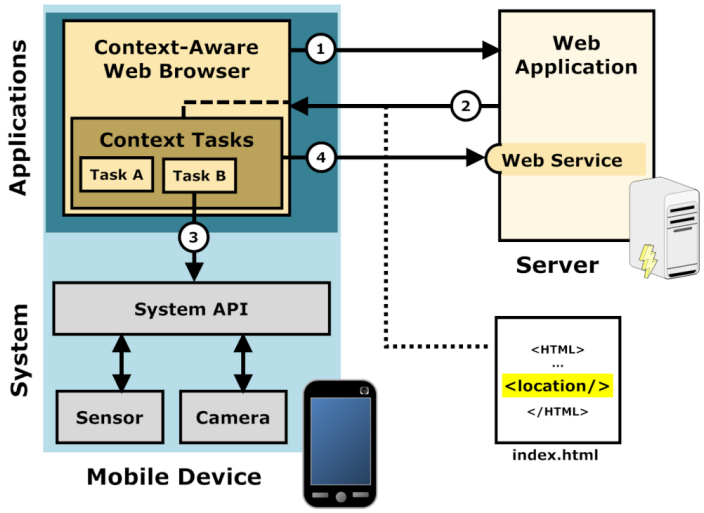

Figura 12.1 – Modelo: aplicaciones web con etiquetas XML de contexto y navegador web sensible al contexto.

12.2 VERIFICACIÓN Y EVALUACIÓN DE LOS OBJETIVOS

En el capítulo 1 de esta memoria hemos planteado un grupo de objetivos específicos, los cuales han sido cumplidos durante el desarrollo de esta tesis doctoral. La consecución de estos objetivos específicos implica el cumplimiento de los objetivos generales, enumerados de igual manera en dicho capítulo, dado que se derivan directamente de estos. A continuación se enumeran dichos objetivos y se enuncia la forma en la que ha sido conseguido.

- **Realizar un análisis global de la situación actual de Internet de las cosas, abarcando el estudio de los distintos tipos de interacción entre objetos, especialmente de aquellos aspectos relacionados con la utilización de los teléfonos móviles inteligentes**. En este primer análisis se ha contextualizado el marco de trabajo de esta tesis, identificando las características y aspectos más relevantes de dos de los pilares que definen el marco de trabajo de esta tesis, Internet de las cosas y los teléfonos móviles inteligentes. Este análisis se ha dividido en dos partes, la primera se ha presentado en el capítulo 3 y abarca de forma general diversos aspectos del ámbito de Internet de las cosas. La segunda parte del análisis se presenta en el capítulo 4 y está centrada en el estudio de los teléfonos inteligentes. En este estudio se analizan varios aspectos relevantes relacionados con los teléfonos inteligentes, como los servicios, las aplicaciones y las comunicaciones.

- **Realizar un análisis de los sistemas y propuestas que permitan desarrollar aplicaciones móviles ligeras, las cuales puedan acceder a los distintos elementos hardware del teléfono**. Se ha realizado un análisis de las diferentes propuestas y alternativas que basan una parte importante de su funcionalidad en la gestión de los elementos hardware del dispositivo móvil, se han evaluado los aspectos positivos y negativos de cada propuesta en función de los objetivos de esta tesis. Este análisis se ha dividido en dos partes, la primera se centra en las

alternativas capaces de gestionar los elementos de comunicación del dispositivo móvil, con el fin de establecer comunicaciones con otros dispositivos electrónicos. El resultado de este primer análisis se ha presentado en el capítulo 5. La segunda parte se centra en las alternativas que habilitan la gestión de los elementos hardware capaces de capturar información de contexto, este estudio se ha presentado en el capítulo 6.

- **Desarrollar una especificación que permita crear aplicaciones móviles aptas para ser usadas en distintas plataformas móviles**. La especificación presentada en el capítulo 9 de esta tesis "Objetos virtuales v3.0 En la web" permite desarrollar aplicaciones basadas en tecnologías web capaces de gestionar los elementos hardware del dispositivo cliente. Esta especificación no contiene elementos ligados a plataformas móviles específicas, sino que introduce conceptos generales que pueden ser aplicados en las distintas plataformas móviles. La especificación se apoya en las tecnologías web como base para el desarrollo de las aplicaciones, esta característica permite que las aplicaciones generadas puedan ser utilizadas en las plataformas móviles más populares.

- **Dotar a la especificación de mecanismos de acceso a los principales elementos hardware del dispositivo, como sensores, módulos de comunicaciones, etc.** La especificación presentada en el capítulo 9 de esta tesis "Objetos virtuales v3.0 En la web" introduce el uso de etiquetas XML que habilitan la gestión de la mayoría de los elementos hardware del dispositivo cliente. Entre estos elementos hardware se encuentran los que habilitan las comunicaciones con dispositivos cercanos y los que capturan diferentes tipos de información de contexto.

  El conjunto de etiquetas XML propuesto permite especificar con alto grado de detalle y capacidad de configuración, las acciones que los elementos hardware del dispositivo pueden realizar.

- **Dotar a la especificación de una arquitectura bien estructurada y modular que permita modificar o añadir mecanismos que habiliten la gestión de nuevos elementos hardware.** La especificación presentada en el capítulo 9 de esta tesis "Objetos virtuales v3.0 En la web", presenta un conjunto de etiquetas XML capaces de describir de manera detallada un requerimiento de comunicación o de información de contexto. Por otro lado, la propuesta introduce las tareas programadas, que son los componentes encargados de satisfacer los requerimientos. Cada etiqueta XML está asociada a una tarea programada, la arquitectura del sistema otorga una total independencia a cada uno de los pares etiqueta-tarea, de forma que uno de estos pares puede ser modificado sin afectar al resto. Del mismo modo, la especificación permite incluir nuevos pares etiqueta-tarea de forma metódica y simple.

- **Diseñar e incluir en la propuesta mecanismos simplificados de acceso a los elementos hardware del dispositivo**. Las etiquetas XML se utilizan para expresar los requerimientos de comunicación o de información de contexto, estos

requerimientos se satisfacen en el dispositivo cliente por medio de un proceso que puede implicar una o varias llamadas al API encargado de gestionar los elementos hardware del dispositivo. Se evita que el desarrollador tenga que conocer detalles concretos de la plataforma y se consigue ejecutar la acción deseada sobre el elemento hardware del dispositivo por medio de un conjunto de etiquetas y atributos que tienen un gran nivel de abstracción. Las evaluaciones realizadas en el capítulo 11 verifican que en múltiples escenarios, los mecanismo de acceso a los elementos hardware propuestos resultan menos complejos que los del resto de soluciones evaluadas.

- **Dotar a la especificación de las características necesarias para poder generar aplicaciones bajo una arquitectura que favorezca la reducción del tiempo empleado en procesos secundarios, como configuraciones e instalaciones. Estas aplicaciones mejoraran la experiencia del usuario en entornos que se basan en la interacción ocasional o puntual con objetos físicos.** La especificación presentada en el capítulo 9 de esta tesis "Objetos virtuales v3.0 En la web" describe un sistema que puede ser integrado con gran variedad de tecnologías web actuales, con el fin de desarrollar aplicaciones web capaces de gestionar los elementos hardware del dispositivo cliente. Las aplicaciones web no requieren de un proceso de instalación y en la mayoría de los casos sus tiempos de carga son relativamente rápidos. Este tipo de aplicaciones web resultan muy adecuadas en escenarios concebidos para interacciones ocasionales que quizás el usuario nunca más vuelva a repetir. Tal y como se verifica en el capítulo 11, cuando se utiliza por primera vez una aplicación, el proceso de acceso a la aplicación web es significativamente más rápido que el del resto de soluciones evaluadas.

- **Utilizar la especificación para desarrollar un conjunto de aplicaciones móviles, las cuales deben servir para ilustrar las características más destacadas de la propuesta**. En el capítulo 10 se ha detallado el proceso de desarrollo de varios prototipos funcionales, los cuales emplean diferentes partes de la especificación propuesta en sus procesos de negocio. Estos prototipos han sido desarrollados combinando el uso de tecnologías web tradicionales como HTML, Javascript y PHP con la especificación propuesta en el capítulo 9 "Objetos virtuales v3.0 En la Web".

- **Comparar las aplicaciones desarrolladas siguiendo la especificación propuesta con otras aplicaciones de similares funcionalidades desarrolladas con otras tecnologías, con el fin de analizar varias de sus características como el rendimiento técnico y la usabilidad**. En el capítulo 11 se han realizado diversas evaluaciones que implicaban el uso de multitud de aplicaciones web, desarrolladas bajo parte de la especificación propuesta. Se han realizado evaluaciones de varios de los factores críticos que condicionan: el rendimiento de las aplicaciones móviles, la experiencia de usuario y los procesos de desarrollo de aplicaciones. Además en varias de estas evaluaciones las aplicaciones que seguían la especificación propuesta han sido comparadas con aplicaciones similares desarrolladas bajo otras alternativas.

12.3 ÁMBITOS DE APLICACIÓN

La especificación propuesta en esta tesis tiene diversos ámbitos de aplicación, a continuación se listan algunos de los más significativos:

1. **Interacción ocasional con objetos**. Cada vez resulta más frecuente la existencia de procesos que implican la interacción con objetos físicos o dispositivos virtuales, algunos de estos procesos están diseñados para que se realicen de forma ocasional. En la mayoría de los casos los teléfonos inteligentes que interactúan con otros elementos se sirven de una aplicación específica que se encarga de gestionar la interacción (realizar envíos de información, interpretar datos, transmitir las acciones del usuario, etc). La naturaleza de las aplicaciones nativas no se adecua a los sistemas que se basan en interacciones ocasionales con objetos físicos, interacciones que quizá el usuario nunca vuelva a repetir. Los procesos de descarga y configuración requeridos por las aplicaciones nativas pueden llegar a consumir más tiempo que la propia lógica de la interacción. La especificación propuesta elimina ese problema puesto que la carga de las aplicaciones web es un proceso significativamente más rápido.

2. **Ahorro de costes de desarrollo y mantenimiento de aplicaciones multiplataforma**. Esta especificación habilita el desarrollo de aplicaciones web multiplataforma capaces de utilizar información de contexto o establecer comunicaciones con otros dispositivos electrónicos cercanos. El desarrollo único, evita que los desarrolladores deban conocer las tecnologías y lenguajes propios de cada plataforma y simplifica los procesos de actualización.

3. **Prototipos y desarrollos rápidos**. El desarrollo de aplicaciones móviles bajo la mayoría de plataformas requiere dotar a los desarrolladores de multitud de conocimientos técnicos. Estos conocimientos abarcan aspectos genéricos de la plataforma y también otros más específicos, como por ejemplo las APIs que gestionan los elementos hardware del dispositivo. La especificación propuesta permite utilizar tecnologías web clásicas para la mayor parte del desarrollo, permitiendo aprovechar multitud de herramientas y gestores. La utilización de las etiquetas XML de contexto permite al desarrollador abstraerse de los detalles técnicos de la plataforma y expresar de forma genérica con unas pocas palabras procesos equivalentes a decenas de líneas de código.

4. **Interacción habitual con multitud de objetos**. Es previsible que en el futuro el número de objetos físicos y dispositivos electrónicos con los que las personas interactuarán a través de sus teléfonos móviles aumente drásticamente. Actualmente, las aplicaciones nativas contenidas localmente en los teléfonos rigen los procesos de interacción con otros objetos físicos y dispositivos electrónicos. Si el número de objetos físicos y dispositivos electrónicos con los que las personas interactúan de manera habitual aumenta, seguramente también aumentará la cantidad de aplicaciones que los usuarios deberán gestionar en sus teléfonos móviles, con el consecuente consumo de tiempo y recursos que esto implica.

Las aplicaciones web que utilizan la especificación propuesta son capaces de interactuar con objetos físicos y dispositivos electrónicos de manera remota, sin requerir las labores de gestión de las aplicaciones locales (instalar, desinstalar, actualizar, configurar…). Es posible asociar las URLs de las aplicaciones web a códigos QR (código de barras, NFC, RFID, etc) colocados en los propios objetos o dispositivos, de esta forma cuando el teléfono lee el código, puede cargar de forma automática la aplicación web indicada. La utilización de las aplicaciones web para estas labores puede liberar al usuario la gestión de un gran número de aplicaciones nativas y también contribuir a optimizar los recursos del teléfono móvil.

5. **Mejora de la experiencia de usuario y optimización de los procesos de negocio en aplicaciones web existentes**. Las etiquetas XML de contexto presentadas en esta tesis pueden ser incorporadas de manera sencilla y no restrictiva a aplicaciones web en funcionamiento, con el fin de mejorar la experiencia del usuario u optimizar algún proceso de negocio. La información de contexto procede del entorno. Este tipo de información puede incorporarse a determinados procesos con el fin de sustituir las entradas de datos manuales, limitando así las posibilidades de que los usuarios cometan errores y reduciendo el tiempo empleado en los procesos de introducción de datos. Esta característica es especialmente útil en dispositivos móviles, donde las limitaciones de sus interfaces, como teclados reducidos y pantallas táctiles provocan que la entrada de datos sea un proceso relativamente costoso para una parte importante de los usuarios.

Los ámbitos de aplicación citados no deberían verse como elementos excluyentes, al contrario, los mayores beneficios se ponen de manifiesto cuando se combinan varios de los ámbitos citados. Por ejemplo, al diseñar un sistema para la **Interacción ocasional con objetos** (ámbito 1) seguramente también se pretenda que el sistema pueda ser utilizado en el máximo número de plataformas móviles, **ahorro de costes de desarrollo y mantenimiento de aplicaciones multiplataforma** (ámbito 2) y que además el proceso de desarrollo de las aplicaciones implicadas en el sistema sea lo más rápido y sencillo posible, **prototipos y desarrollos rápidos** (ámbito 3).

12.4 PRINCIPALES APORTACIONES

A continuación, se enumeran los principales beneficios y aportaciones que surgen de esta tesis:

- ✓ Una especificación que permite desarrollar aplicaciones web móviles multiplataforma capaces de gestionar los elementos hardware del dispositivo cliente. Además, la especificación propuesta no está ligada a ninguna tecnología concreta por lo que su uso puede extenderse a la mayoría de tecnologías web utilizadas en la actualidad en el desarrollo de aplicaciones web.

- ✓ Un lenguaje XML capaz de describir de manera concreta y detallada diversos requerimientos de comunicación e información de contexto. El conjunto de etiquetas propuesto permite expresar la mayoría de los requerimientos de información de contexto y acciones de comunicaciones necesarios en las aplicaciones móviles actuales. La independencia del conjunto de etiquetas respecto al resto de la especificación, permite añadir nuevos elementos o modificar la estructura de los que ya existen.

  Las etiquetas XML de contexto son, a su vez, un mecanismo simplificado que permite a los desarrolladores expresar acciones o requerimientos con un alto grado de abstracción, sin conocer detalles de la plataforma ni del propio elemento hardware que satisface el requerimiento.

- ✓ Una especificación detallada e independiente de la plataforma para la construcción de un navegador web que permita satisfacer los requerimientos de información de contexto y comunicación expresados por las etiquetas XML de contexto.

- ✓ Un navegador web sensible al contexto implementado en la plataforma móvil Android, siguiendo la especificación propuesta en el capítulo 9 "Objetos virtuales v3.0 En la Web".

12.5 PRINCIPALES RESULTADOS DE LA INVESTIGACIÓN

Los siguientes trabajos han sido presentados en diferentes medios para difundir y contrastar ante la comunidad científica los resultados que se reflejan en esta tesis.

- Pascual, J., García-Díaz. V., Sanjuán, O., Pelayo, B.C. & Cueva. J.M. (In review). Mobile Web-based system for remote control electronic devices and smart objects. IEEE Internet Computing **JCR: 2.514.**

- Pascual, J., Sanjuán, O., Pelayo, B.C. & Cueva. J.M. (In review). A simple model based on web services to exchange context information between web browsers and web applications. Journal of Universal Computer Science **JCR: 0.669**.

- Pascual, J., González, R., Sanjuán, O., Pelayo, B.C. & Cueva. J.M. (In review). A lightweight approach to managing real world information in mobile web applications. IEEE Pervasive Computing **JCR: 2.189.**

- Pascual, J., Sanjuán, O., Pelayo, B.C. & Cueva. J.M. (In review). Using collaborative virtual objects based on services to communicate smart objects. IET Software **JCR: 0.671**.

- Pascual, J., Sanjuán, O., Pelayo, B.C. & Cueva. J.M. (Accepted, in press). Using mobile web applications as a multiplatform mechanism to communicate users with embedded devices. Embedded Systems: Theory, Applications and Role in Quantum Mechanics.

- Pascual, J., González, R., Sanjuán, O., Pelayo, B.C. & Cueva. J.M. (2012). Extensible architecture for context-aware mobile web applications. Expert Systems with Applications **JCR: 1.924** , 39(10), 9686-9694.

- Pascual, J. (2011, 15-16 December 2011). Collaborative virtual objects based on services within the Internet of Things. Poster presented at I Jornadas Doctorales de la Universidad de Oviedo.

- Cueva. J.M., Pascual, J., Sanjuán, O. & Pelayo, B.C. (2011). Internet de los objetos. Netbiblio .Editorial: Netbiblo, ISBN**:** 9788497454766.

- Pascual, J., González, R., Sanjuán, O., Pelayo, B.C. & Cueva. J.M. (2011). Virtual objects on the Internet of things. International Journal of Interactive Multimedia and Artificial Intelligence. 1(4), 22-29.

- Pascual, J. (2011). Internet de las Cosas Cuando los objetos hablen entre sí. Certamen FECYT de Comunicación Científica.

- Pascual, J., Sanjuán, O., Cueva, J.M., Pelayo, B.C., Álvarez, M. & González, A. (2011) Modeling architecture for collaborative virtual objects based on services. Journal of Network and Computer Applications **JCR: 0.660**. 34(5), 1634-47.

- Pascual, J., González, R., Sanjuán, O., Pelayo, B.C., Cueva. J.M. & Ordoñez, P. (2011). Standardization of Virtual Objects. Semantic Web Personalization And Context Awareness. 7-21.

12.6 LÍNEAS DE INVESTIGACIÓN Y TRABAJO FUTURO

Si bien el objetivo de este trabajo (establecido al inicio de esta tesis en el capítulo 1) ha sido cumplido, todavía queda trabajo por hacer. El modelo puede ser refinado y extendido para soportar características adicionales u optimizar alguna de las funcionalidades existentes. En la lista que se muestra a continuación se enumeran algunas de las principales líneas de investigación y trabajo futuro:

- **Soporte automatizado a la creación de servicios web**. Desarrollo de una o varias herramientas específicas que permitan automatizar la generación de servicios web responsables de recibir y procesar la información de contexto y las respuestas de las acciones de comunicación.

- **Gestión de los elementos conectados al dispositivo**. La propuesta contempla soporte para la gestión de los elementos hardware del propio dispositivo, en un futuro se pretende extender este soporte a otros dispositivos o elementos hardware externos, que puedan estar ocasionalmente vinculados al teléfono. Entre los cuales podemos encontrar: un dispositivo manos libres bluetooth, elementos capaces de monitorizar parámetros corporales con fines clínicos, lectores externos de RFID, etc.

- **Diseño y construcción de una herramienta visual que ofrezca un soporte automatizado a la generación de etiquetas XML de contexto que se incluyen en aplicaciones web**. Algunas etiquetas pueden contener varios atributos que sólo admiten valores que estén dentro de un rango, debido a estas restricciones existe la posibilidad de que el desarrollador asigne valores no válidos a los atributos. Se han especificado varios XML Schema para que los desarrolladores puedan validar las etiquetas XML, pero en un futuro se espera desarrollar una herramienta especialmente diseñada para este propósito.

- **Evolución de la especificación y arquitectura del navegador web sensible al contexto**. Se realizará una revisión de los navegadores web desarrollados en diferentes plataformas siguiendo la especificación propuesta, con el fin de incluir nuevas recomendaciones que permitan optimizar el rendimiento y reducir el tráfico de datos.

- **Integración con herramientas de desarrollo**. Crear plugins para los entornos de desarrollo Eclipse y Visual Studio, estos plugins deben facilitar a los desarrolladores la integración de los elementos propios de esta especificación en sus aplicaciones web.

- **Análisis del comportamiento de los usuarios frente a los requerimientos de información de contexto y acciones de comunicación**. De este análisis se pretenden extraer posibles errores de presentación y patrones de comportamiento de los usuarios. Este análisis tiene como objetivo conseguir datos contrastados que permitan realizar mejoras en la capa de presentación del navegador sensible al contexto.

- **Nuevas implementaciones del navegador sensible al contexto**. Implementar una versión del navegador web sensible al contexto para cada una de las plataformas móviles más utilizadas en la actualidad, Blackberry Os, Symbian, etc.

- **Evaluación de la experiencia de usuario en personas no familiarizadas con el uso de los Smartphones**. Las pruebas realizadas durante esta tesis se centraron en usuarios familiarizados con el uso de los Smartphones. Se desarrollará una nueva batería de pruebas especialmente orientadas a usuarios inexpertos, con el fin de evaluar el impacto de la propuesta en este tipo de usuarios.

- **Desarrollo de una métrica que permita estimar los costes derivados de la implementación de aquellos fragmentos de código que hacen uso de los elementos hardware del dispositivo**. Esta métrica debe ser aplicable indistintamente a diversas plataformas y lenguajes de programación para permitir realizar comparaciones objetivas y fiables.

- **Revisión de algunas etiquetas XML de contexto y tareas programadas de contexto con el fin de ofrecer mayores posibilidades de configuración**. Sobre todo centrando la labor de rediseño en las etiquetas XML de contexto basadas en acciones de comunicación, las cuales no aprovechan todas las posibilidades ofrecidas por las tecnologías de comunicación en las que se basan.

- **Incluir mecanismos que permitan la ocultación de la sintaxis para las etiquetas XML incluidas, presentes en las páginas web**. Algunas aplicaciones pueden no desear que los usuarios sean capaces de visualizar la sintaxis de las etiquetas XML de contexto.

- **Tutoriales y material de aprendizaje para la difusión de la propuesta**.

- **Nuevos mecanismos de intercambio de datos entre el cliente móvil y el servidor web**. La especificación utiliza peticiones POST como medio para intercambiar la información de contexto y comunicación entre el cliente móvil y el servidor web; en el futuro se investigará la integración con otros mecanismos de comunicación más seguros y eficientes.

BIBLIOGRAFÍA Y REFERENCIAS

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999). Towards a Better Understanding of Context and Context-Awareness. Paper presented at the Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing.

ActiveX Controls. (2012). <http://msdn.microsoft.com/en-us/library/aa751968(v=vs.85).aspx/>.

Adelmann, R., Langheinrich, M. & Floerkemeier, C. (2006). A toolkit for bar-coderecognition and -resolving on camera phones – jump starting the Internet of things. Workshop Mobile and Embedded Interactive Systems.

Ahn, C., & Nah, Y. (2010, 7-9 June 2010). Design of Location-Based Web Service Framework for Context-Aware Applications in Ubiquitous Environments. Paper presented at the Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on.

Applets Java Sun. (2011). <http://java.sun.com/applets/>.

Armenio, F., Barthel, H., Burstein, L., Dietrich, P., Duker, J., Garrett, J., Hogan, B., Ryaboy, O., Sarma, S., Schmidt, J., Suen, K., Traub, K. & Williams, J. (2007). The EPCglobal Architecture Framework.

Atukorala, K., Wijekoon, D., Tharugasini, M., Perera, I. & Silva, C. (2009, 15-18 Sept. 2009). SmartEye Integrated Solution to Home Automation, Security and Monitoring through Mobile Phones. Paper presented at the Next Generation Mobile Applications, Services and Technologies, 2009. NGMAST '09. Third International Conference on.

Bhatti, R., Bertino, E., & Ghafoor, A. (2004, 6-9 July 2004). A trust-based context-aware access control model for Web-services. Paper presented at the Web Services, 2004. Proceedings. IEEE International Conference on.

Biegel, G., & Cahill, V. (2004). A Framework for Developing Mobile, Context-aware Applications. Paper presented at the Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04).

Bluetooth. (2009). Bluetooth Core Specification v3.0 + HS. <http://www.Bluetooth.com/Bluetooth/Technology/Building/Specifications/>.

Brock, D.L. (2001). The electronic product code (EPC)-a naming scheme. Technical Report MIT-AUTOID-WH-002, MIT Auto ID Center.

Carabelea, C. & Boissier, O. (2003). Multi-agent platforms on smart devices: Dream or reality?. Proceedings of the Smart Objects Conference 03.

Chakravorty, R. (2006, 13-17 March 2006). A programmable service architecture for mobile medical care. Paper presented at the Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on.

Challiol, C., Fortier, A., Gordillo, S., & Rossi, G. (2007, 3-7 Sept. 2007). A Flexible Architecture for Context-Aware Physical Hypermedia. Paper presented at the Database and Expert Systems Applications, 2007. DEXA '07. 18th International Workshop on.

Chang, Y. F., Chen, C. S., & Zhou, H. (2009). Smartphone for mobile commerce. Computer Standards and Interfaces, 31(4), 740-747.

Chen, G., and Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. Technical Report. Dartmouth College, Hanover, NH, USA.

Chen, H., Finin, T., & Joshi, A. (2003). Using OWL in a Pervasive Computing Broker. In Proceedings of the Workshop on Ontologies in Agent Systems (OAS 2003), Melbourne, Australia.

Chen, H., Finin, T., & Joshi, A. (2004, 14-17 March 2004). Semantic Web in the context broker architecture. Paper presented at the Pervasive Computing and Communications, 2004. PerCom 2004. Proceedings of the Second IEEE Annual Conference on.

Cheverst, K., Mitchell, K., & Davies, N. (1999). Design of an object model for a context sensitive tourist GUIDE. Computers & Graphics, 23(6), 883-891.

Chua, A. Y. K., Balkunje, R. S. & Dion Hoe-Lian, G. (2011, 22-25 March 2011). The Influence of User-Context on Mobile Information Needs. Paper presented at the Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on.

Collaborative Business Items Funded By The European Comission. (2007). <http://www.cobis-online.de/>.

Coppola, P., Della Mea, V., Di Gaspero, L., Menegon, D., Mischis, D., Mizzaro, S., et al. (2010). The Context-Aware Browser. Intelligent Systems, IEEE, 25(1), 38-47.

Coppola, P., Della Mea, V., Di Gaspero, L., Mizzaro, S., Scagnetto, I., Selva, A., Vassena, L., et al. (2005). Information Filtering and Retrieving of Context-Aware Applications within the MoBe Framework. Security. In proceedings of Proceedings of the Workshop on Context-Based Information Retrieval.

De, S. & Moessner, K. (2009). A framework for mobile, context-aware applications. Paper presented at the Proceedings of the 16th international conference on Telecommunications.

Decker, C., Spiess, P., Moreira sa de Souza, L., Beigl, M. & Nochta, Z. (2006).Coupling Enterprise Systems with Wireless Sensor Nodes: Analysis, Implementation, Experiences and Guidelines. Pervasive Technology Applied.

DeviceOrientation Event Specification. (2011). <http://dev.w3.org/geo/api/spec-source-orientation.html/>.

Dey, A. K., Abowd, G. D. & Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Hum.-Comput. Interact., 16(2), 97-166.

Ecma. (2005). Near Field Communication White paper. <http://www.ecma-international.org/activities/Communications />.

Eissele, M., Weiskopf, D. & Ertl, T. (2009). Interactive Context-Aware Visualization for Mobile Devices. Paper presented at the Proceedings of the 10th International Symposium on Smart Graphics.

Ennai, A. & Bose, S. (2008, 16-16 Sept. 2008). MobileSOA: A Service Oriented Web 2.0 Framework for Context-Aware, Lightweight and Flexible Mobile Applications. Paper presented at the Enterprise Distributed Object Computing Conference Workshops, 2008 12th.

EPCglobal . (2008). EPCglobal Object Name Service (ONS) 1.0.1. <http://www.gs1.org/gsmp/kc/epcglobal/ons/ons_1_0_1-standard-20080529.pdf />.

Fajardo, R., Miramá, V., Caicedo, 0. Mesa, J. & Martínez, F.O. (2008) Descubrimiento e interacción de servicios móviles ubicuos utilizando Bluetooth y Wi-Fi. Sistemas y Telemática. 6(11), 55-73.

Fasbender, A., Hoferer, S., Gerdes, M., Matsumura, T., Häber, A. & Reichert, F. (2008, 16-19 Sept. 2008). Phone-Controlled Delivery of NGN Services into Residential Environments. Paper presented at the Next Generation Mobile Applications, Services and Technologies, 2008. NGMAST '08. The Second International Conference on.

Ferreira, J. C. & Afonso, J. L. (2011, 27-30 June 2011). Mobi_System: A personal travel assistance for electrical vehicles in smart cities. Paper presented at the Industrial Electronics (ISIE), 2011 IEEE International Symposium on.

Filibeli, M. C., Ozkasap, O. & Civanlar, M. R. (2007). Embedded web server-based home appliance networks. J. Netw. Comput. Appl., 30(2), 499-514.

Fleisch, E. (2010) .What is the Internet of Things? An Economic Perspective. Auto-ID Labs White Paper WP-BIZAPP-053.

Floerkemeier, C., Roduner, C. & Lampe, M. (2007). RFID Application Development With the Accada Middleware Platform. Systems Journal, IEEE, 1(2), 82-94.

Främling, K., Holmström, J., Ala-Risku, T. & Kärkkäinen, M. (2003). Product agents for handling information about physical objects. Technical report, Helsinki University of Technology.

Fuhrmann, T. & Harbaum, T. (2003). Using Bluetooth for informationally enhanced environments. Proceedings of the IADIS International Conference e-Society.

Gasimov, A., Magagna, F. & Sutanto, J. (2010). CAMB: context-aware mobile browser. Paper presented at the Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia.

Geolocation API specification. (2012). <http://dev.w3.org/geo/api/>.

Global Trends 2025: A transformed world. (2008). Appendix F: The Internet of Things (Background). SRI Consulting Business Intelligence.

Gossweiler, R., McDonough, C., Lin, J. & Want, R. (2011). Argos: Building a Web-Centric Application Platform on Top of Android. Pervasive Computing, IEEE, 10(4), 10-14.

Grønbæk, I. (2008). Connecting objects in the Internet of Things (IoT). Telenor ASA.

GSM Association. (2007). Mobile NFC Services. <http://www.gsmworld.com/documents/nfc_services_0207.pdf/>.

Hardgrave, B., Waller, M. & Miller, R. (2005). Does RFID Reduce Out of Stocks? A Preliminary Analysis. RFID Research center Walton College, University of Arkansas.

Hattori, S., Tezuka, T. & Tanaka, K. (2007, 15-19 Jan. 2007). Context-Aware Query Refinement for Mobile Web Search. Paper presented at the Applications and the Internet Workshops, 2007. SAINT Workshops 2007. International Symposium on.

Hernandez, E. A. (2009). War of the Mobile Browsers. Pervasive Computing, IEEE, 8(1), 82-85.

Hickson I. (2011). HTML 5 W3C working draft. <http://www.w3.org/TR/html5/>.

Hyunho, P., Byoungoh, K., Yangwoo, K. & Dongman, L. (2011, 21-25 March 2011). InterX: A service interoperability gateway for heterogeneous smart objects. Paper presented at the Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on.

ITU Internet Reports. (2005). The Internet of things - Executive sumario.

Java Web Start. (2011). <http://www.oracle.com/technetwork/java/javase/javawebstart/>.

Johnson, S. (2007). A framework for mobile context-aware applications. BT Technology Journal, 25(2), 106-111.

Juan, W., Weiliang, L. & Sining, M. (2011, 28-30 Sept. 2011). A convergence service oriented architecture in smart homes. Paper presented at the ICT Convergence (ICTC), 2011 International Conference on.

Kanma, H., Wakabayashi, N., Kanazawa, R. & Ito, H. (2003). Home appliance control system over Bluetooth with a cellular phone. Consumer Electronics, IEEE Transactions on, 49(4), 1049-1053.

Kanma, H., Wakabayashi, N., Kanazawa, R. & Ito, H. (2003). Home appliance control system over Bluetooth with a cellular phone. Consumer Electronics, IEEE Transactions on, 49(4), 1049-1053.

Kapitsaki, G. M., Kateros, D. A., & Venieris, I. S. (2008, 15-18 Sept. 2008). Architecture for provision of context-aware web applications based on web services. Paper presented at the Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on.

Kärkkäinen, M., Holmström, J., Främling, K. & Artto, K. (2003). Intelligent products: a step towards a more effective project delivery chain. Comput. Ind. 50(2), 141-151.

Karnouskos, S. (2007). Lead market potential: The promise of Networking Embedded Devices. Consultation Meeting of the European Commission on "Sensing, Monitoring and Control".

Karnouskos, S., & Tariq, M. M. J. (2009, 23-25 March 2009). Using multi-agent systems to simulate dynamic infrastructures populated with large numbers of web service enabled devices. Paper presented at the Autonomous Decentralized Systems, 2009. ISADS '09. International Symposium on.

Kim, N., Lee, H. S., Oh, K. J. & Choi, J. Y. (2009). Context-aware mobile service for routing the fastest subway path. Expert Systems with Applications, 36(2, Part 2), 3319-3326.

Kortuem, G., Kawsar, F., Sundramoorthy, V. & Fitton, D. (2010). Smart objects as building blocks for the Internet of things. IEEE Internet Computing. 14(1), 44-51.

Kranz, M., Holleis, P. & Schmidt, A. (2010). Embedded Interaction: Interacting with the Internet of Things. IEEE Internet Computing, 14(2), 46-53.

Lahti, J., Palola, M., Korva, J., Westermann, U., Pentikousis, K. & Pietarila, P. (2006). A mobile phone-based context-aware video management application. Proc. SPIE-IS&T Electronic Imaging (Multimedia on Mobile Devices II*)*, SPIE Vol. 6074, San Jose, California, USA. 6074, 183-194.

Lemlouma, T. & Layaida, N. (2004). Context-aware adaptation for mobile devices. Paper presented at the Mobile Data Management, 2004. Proceedings. 2004 IEEE International Conference on.

Yan, L., Zhang, Y. & Yang, L. T. (2008). The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems: Auerbach Publications.

Meng, W., Chunxiao, F., Zhigang, W. Shan, L. & Jie, L. (2011, 23-25 Sept. 2011). Implementation of Internet of Things Oriented Data Sharing Platform Based on RESTful Web Service. Paper presented at the Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on.

Meyer, G, Främling, K. & Holmström, J. (2009). Intelligent Products: A survey. Computers in Industry. 60(3), 137-148.

Meyer, G., Främling, K. & Holmström, J. (2009). Intelligent Products: A survey. Computers in Industry. 60(3), 137-148.

Myers, B. A., Nichols, J., Wobbrock, J. O. & Miller, R. C. (2004). Taking handheld devices to the next level. Computer, 37(12), 36-43.

Myers, B. A., Stiel, H. & Gargiulo, R. (1998). Collaboration using multiple PDAs connected to a PC. Paper presented at the Proceedings of the 1998 ACM conference on Computer supported cooperative work.

Naphade, M., Banavar, G., Harrison, C., Paraszczak, J. & Morris, R. (2011). Smarter Cities and Their Innovation Challenges. Computer, 44(6), 32-39.

Nathanail, S., Tsetsos, V., & Hadjiefthymiades, S. (2007). Sensor-driven adaptation of web document presentation. Paper presented at the Proceedings of the 4th international conference on Universal access in human-computer interaction: applications and services.

NFC Forum. (2012). <http://www.nfcforum.org/>.

Nichols, J. & Myers, B. A. (2006). Controlling Home and Office Appliances with Smartphones. Pervasive Computing, IEEE, 5(3), 60-67.

Noltes, J. (2008). An Architecture For Context-aware Mobile Web Browsing. 9TH TSCONIT, university of twente.

Ok, K., Coskun, V., Aydin, M. N. & Ozdenizci, B. (2010, 2-4 Nov. 2010). Current benefits and future directions of NFC services. Paper presented at the Education and Management Technology (ICEMT), 2010 International Conference on.

Pantsar-Syvaniemi, S., Simula, K. & Ovaska, E. (2010, 22-25 June 2010). Context-awareness in smart spaces. Paper presented at the Computers and Communications (ISCC), 2010 IEEE Symposium on.

Pascual, J., González, R., Sanjuán, O., Pelayo, B.C. & Cueva. J.M. (2012). Extensible architecture for context-aware mobile web applications. Expert Systems with Applications, 39(10), 9686-9694.

Pascual, J., Sanjuán, O., Cueva, J.M., Pelayo, B.C., Álvarez, M. & González, A. (2011a) Modeling architecture for collaborative virtual objects based on services. Journal of Network and Computer Applications. 34(5), 1634-47.

Pascual, J., Sanjuán, O., Pelayo, B.C., Cueva. J.M. & Ordoñez, P. (2011b). Standardization of Virtual Objects. Semantic Web Personalization And Context Awareness. 7-21.

Pascual, J., Sanjuán, O., Pelayo, B.C. & Cueva. J.M. (2011c). Virtual objects on the Internet of things. International Journal of Interactive Multimedia and Artificial Intelligence. 1(4), 22-29.

Perumal, T., Ramli, A. R., Chui Yew, L., Mansor, S. & Samsudin, K. (2008, Nov. 30 2008-Dec. 3 2008). Interoperability among Heterogeneous Systems in Smart Home Environment. Paper presented at the Signal Image Technology and Internet Based Systems, 2008. SITIS '08. IEEE International Conference on.

PhoneGap (2011) . <http://phonegap.com/>.

Piyare, R. & Tazil, M. (2011, 14-17 June 2011). Bluetooth based home automation system using cell phone. Paper presented at the Consumer Electronics (ISCE), 2011 IEEE 15th International Symposium on.

Quack, T., Bay, H, & Gool, L. V. (2008). Object recognition for the Internet of things. Paper presented at the Proceedings of the 1st international conference on The Internet of things.

Ramparany, F. & Boissier, O. (2002). Smart devices embedding multi-agent technologies for a pro-active world. In Proc. of the Ubiquitous Computing Workshop.

RHUB. (2012). <http://www.rhubcom.com/>.

Riva, O. & Toivonen, S. (2006, 26-29 June 2006). A Hybrid Model of Context-aware Service Provisioning Implemented on Smartphones. Paper presented at the Pervasive Services, 2006 ACS/IEEE International Conference on.

Rohs, M. & Gfeller, B. (2004).Using camera-equipped mobile phones for interacting with real-world objects. Advances in Pervasive Computing, Austrian Computer Society (OCG).

Román, M.,Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. H., et al. (2002). A Middleware Infrastructure for Active Spaces. IEEE Pervasive Computing, 1(4), 74-83.

Rose, B. (2001). Home networks: a standards perspective. Communications Magazine, IEEE, 39(12), 78-85.

Salber, D., Dey, A. K., & Abowd, G. D. (1999). The context toolkit: aiding the development of context-enabled applications. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit.

Samsung Remote. (2011).
<https://play.google.com/store/apps/details?id=com.samsung.remoteTV/>

Sanchez, L., Galache, J. A., Gutierrez, V., Hernandez, J. M., Bernat, J., Gluhak, A. & Garcia, T. (2011, 15-17 June 2011). SmartSantander: The meeting point between Future Internet research and experimentation and the smart cities. Paper presented at the Future Network & Mobile Summit (FutureNetw), 2011.

Schilit, B. & Theimer, M. (1994). Disseminating active map information to mobile hosts. Network, IEEE, 8(5), 22-32.

Schilit, B., Adams, N. & Want, R. (1994, 8-9 Dec. 1994). Context-Aware Computing Applications. Paper presented at the Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on.

Schmidt, A. (1999). There is more to context than location. Computers & Graphics, 23(6), 893-901.

Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Laerhoven, K. V. & Velde, W. V. d. (1999). Advanced Interaction in Context. Paper presented at the Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing.

Sheng, Q. Z., Pohlenz, S., Jian, Y., Wong, H. S., Ngu, A. H. H. & Maamar, Z. (2009, 16-24 May 2009). ContextServ: A platform for rapid and flexible development of context-aware Web services. Paper presented at the Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on.

Shirazi, A. S., Winkler, C. & Schmidt, A. (2010, Nov. 29 2010-Dec. 1 2010). SENSE-SATION: An extensible platform for integration of phones into the Web. Paper presented at the Internet of Things (IOT), 2010.

Sundmaeker, H., Patrick Guillemin, P., Friess, P. & Woelfflém, S. (2010). Vision and Challenges for Realising the Internet of Things. CERP-IoT.

Thiesse, F., Floerkemeier, C., Harrison, M., Michahelles, F. & Roduner, C. (2009). Technology, Standards, and Real-World Deployments of the EPC Network. Internet Computing, IEEE, 13(2), 36-43.

Thompson, C. W. (2005). Smart devices and soft controllers. Internet Computing, IEEE, 9(1), 82-85.

Toye, E., Sharp, R., Anil, M. & Scott, D. (2005). Using Smartphones to access site-specific services. Pervasive Computing, IEEE, 4(2), 60-66.

Vale, S. & Hammoudi, S. (2009, 24-28 May 2009). COMODE: A Framework for the Development of Context-Aware Applications in the Context of MDE. Paper presented at the Internet and Web Applications and Services, 2009. ICIW '09. Fourth International Conference on.

WCL. (2012). Wireless Communication Library . <http://www.btframework.com/>.

Webkit . (2011). <http://www.webkit.org/>.

Weerawarana, S., Curbera, F., Leymann, F., Storey, T. & Ferguson, D. F. (2005). Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More: Prentice Hall PTR.

Wei-Dar, C., Mayes, K. E., Yuan-Hung, L. & Jung-Hui, C. (2011, 21-23 June 2011). NFC mobile payment with Citizen Digital Certificate. Paper presented at the Next Generation Information Technology (ICNIT), 2011 The 2nd International Conference on.

Wiechert, T., Thiesse, F., Michahelles, F., Schmitt, P. & Fleisch, E. (2009). Connecting Mobile Phones to the Internet Of Things: A Discussion of compatibility issues between EPC technology and NFC Technology. Auto ID Center.

Wi-Fi Alliance. (2010). <http://www.Wi-Fi.org/>.

Xu, H., Su, R., Hou, X., & Ni, Q. (2009, 12-14 Aug. 2009). Remote Control System Design Based on Web Server for Digital Home. Paper presented at the Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on.

ZigBee Alliance . (2012). <http://www.zigbee.org/>.

Z-Wave . (2012). <http://www.z-wave.com/>.