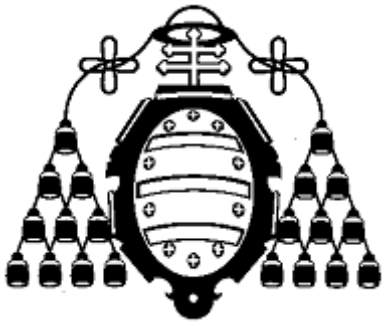


UNIVERSIDAD DE OVIEDO

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

EVALUACIÓN DE LA HERRAMIENTA TITANIUM STUDIO PARA EL DESARROLLO DE APLICACIONES MULTIDISPOSITIVO

TUTORES: CLAUDIO DE LA RIVA Y JAVIER RODRÍGUEZ

AUTOR: DAVID RODRÍGUEZ GONZÁLEZ

JULIO DE 2013

Tabla de contenido

Tabla de figuras

CAPÍTULO 1. INTRODUCCIÓN

1.1. IDENTIFICACIÓN DEL PROBLEMA

El objetivo principal de este proyecto es la evaluación de una herramienta de desarrollo de aplicaciones para dispositivos móviles llamada Titanium.

Para ello, se desarrollará un proyecto lo más parecido posible a una dupla de aplicaciones que el Centro Tecnológico de la Información y Comunicación de Gijón ha desarrollado para dispositivos móviles. El CTIC ha creado una aplicación codificada completamente en el entorno nativo de Android y otra prácticamente igual codificada en el entorno nativo de iOS. Se dice prácticamente igual, porque existen componentes y funcionalidades diferentes dependiendo del sistema operativo dónde se vaya a ejecutar la aplicación final. De las aplicaciones nativas facilitadas por el CTIC para el desarrollo del presente proyecto no se harán menciones ni a su nombre ni a su funcionalidad por temas de confidencialidad del centro.

A lo largo de la creación del proyecto se irán recogiendo impresiones del autor acerca de la plataforma Titanium y del entorno de desarrollo que se utilizará para dicho fin, Titanium Studio. Se alternarán las impresiones del autor con la ejecución de algunas comprobaciones empíricas como tomas de tiempo de ejecución y compilación de las aplicaciones, tiempos de desarrollo, funcionalidades que existen en los entorno nativos que no pueden ser ejecutadas en Titanium, etc.

La aplicación que se creará en Titanium para dicho fin ha sido llamada MobiMarket. La aplicación consiste en un carrito de la compra para un supermercado que pueda correr bajo el sistema operativo Android y el sistema operativo iOS y que permita a sus clientes la realización de transacciones desde un terminal sin necesidad de ir al supermercado o a un computador con conexión a internet.

1.2. JUSTIFICACIÓN DEL PROYECTO

El proyecto fue iniciado a petición del CTIC, que había detectado grandes problemáticas en cuanto al desarrollo de aplicaciones multidispositivo. Algunos de los inconvenientes se enumeran a continuación:

- Múltiples dispositivos móviles: a día de hoy, cuando se desarrolla una aplicación para dispositivos móviles, no sirve con realizarla en un único entorno. Una empresa que se dedica a la programación de software para móviles maneja clientes que utilizan dispositivos con diferentes necesidades. El mercado de los terminales móviles está muy marcado por los usuarios de Android e iOS (estos últimos en menos medida). Pero también se encuentran usuarios de Blackberry OS, Windows Phone. Crear una aplicación que llegue a todo el mundo, es crear una aplicación por cada plataforma.

- Personal: a raíz del problema anterior es obvio que se necesitan equipos de trabajo especializados en cada uno de los sistemas operativos anteriormente nombrados para la creación de aplicaciones para una población. Esto es muy costoso y en algunos casos inviable para la mayoría de las empresas. De hecho, a día de hoy es muy difícil encontrar aplicaciones que estén disponibles para todos los sistemas del mercado. Suelen desarrollarse aplicaciones para Android e iOS que son los entornos que poseen una mayor cota de mercado. Para ello las empresas suelen utilizar dos equipos de trabajo con el incremento de coste que eso supone (el doble normalmente).

- Programación multiplataforma: otro problema común a los dos anteriores es la dificultad de encontrar personal cualificado en todos los lenguajes necesarios para el desarrollo de aplicaciones móviles. Bien es cierto que la creación de aplicaciones para Android y Blackberry OS es similar (Java), pero poco o nada tienen que ver con el lenguaje utilizado para generar aplicaciones para iOS (Objetive-C).

- Mantenimiento: hay que tener en cuenta además, que un proyecto software no sólo es la codificación de una solución, también hay que mantenerla. El mantenimiento de una misma aplicación en distintos lenguajes y corriendo en distintos sistemas operativos, también supone para la una empresa desarrolladora de software un coste mucho mayor al que le supone mantenerla en un único lenguaje y sistema operativo.

La plataforma Titanium busca solventar todos estos problemas, proponiendo un entorno de programación con un lenguaje común (JavaScript), capaz de interpretar el código fuente dándole el formato necesario para que los compiladores de iOS, Android, Windows, BlackberryOS, etc., puedan comprenderlo y generar los instaladores para que la aplicación pueda ser desplegada en cada uno de los sistemas operativos.

Existen otras herramientas como PhoneGap en el mercado, que realizan labores similares. El CTIC ya ha evaluado otras herramientas y el interés se centra únicamente en Titanium.

1.2.1. POSIBILIDADES COMERCIALES DEL PROYECTO

Hay que tener en cuenta que el medio para la evaluación de la plataforma Titanium es la creación de una aplicación que puede ser adquirida por un determinado número de supermercados.

Con este sistema se pretende facilitar la compra a los clientes, desde cualquier lugar dónde se encuentren. Sin necesidad de desplazarse al supermercado. Sin necesidad de estar en casa y tener que realizar la compra a través del ordenador. De esta forma, un supermercado determinado podría ampliar el número de ventas a través de un mayor número de compras de sus clientes actuales y ampliar su número de clientes de forma viral.

1.2.2. ACERCA DEL CTIC

La información que aparece a continuación está directamente sacada de la página web de la fundación CTIC.

--

Centro Tecnológico

La misión de CTIC es la de actuar de agente tractor y transformador en el ámbito de las TIC y sus aplicaciones, potenciando la innovación tecnológica, cultivando el desarrollo de talento y atrayendo recursos financieros. De esta manera se genera el reconocimiento, al apoyar y promover la cooperación, el desarrollo y competitividad de nuestras empresas y la calidad del servicio de las administraciones.

Todos y cada uno de los empleados de CTIC debemos conocer y asumir la misión de la organización, comprendiendo que todo lo que CTIC realiza está orientado hacia su cumplimiento.

Sociedad de la Información

Fundación CTIC Sociedad de la Información tiene la misión de ser un agente para el desarrollo de la Sociedad de la Información en Asturias, ayudando al despliegue de políticas de S.I. del Gobierno Regional así como de otros organismos públicos y privados, para lo que focalizará sus esfuerzos en el desarrollo de acciones dentro de la región.

Como parte de su compromiso social, tiene como misión la colaboración con otros organismos nacionales e internacionales con objetivos similares a los suyos, a través de la transferencia de estrategias, modelos e instrumentos basados en la propia experiencia y el conocimiento adquirido en nuestro territorio, con especial énfasis en los países en vías de desarrollo como destinatarios de la transferencia.

Todo el personal de CTIC Sociedad de la Información debe conocer, compartir y asumir la misión de la organización y comprender que todo lo que realiza está orientado al cumplimiento de dicha misión.

Fundación CTIC Sociedad de la Información debe aportar a la Sociedad todo su conocimiento y esfuerzo para lograr la plena e-Inclusión en todos sus ámbitos (ciudadanía, empresas, instituciones, administraciones y territorios), redundando este esfuerzo en una mejor calidad de vida y en la disminución de las desigualdades inherentes a la propia Sociedad de la Información u otras desigualdades contra las que se puede luchar a través de la apropiación de las TIC o de la innovación social en el ámbito tecnológico.

CTIC Centro Tecnológico despliega su actividad investigadora en el ámbito regional, nacional e internacional (UE).

Colabora para ello con empresas, organismos de investigación y universidades. El objetivo es contribuir a la mejora de la competitividad empresarial a través de la Investigación, el Desarrollo y la innovación tecnológica.

El área de I+D+i organiza sus capacidades tecnológicas en cuatro Unidades:

- Unidad de Tecnologías Semánticas

- Unidad de Movilidad e Independencia de Dispositivo
- Unidad de Tecnologías Emergentes
- Unidad 4U

Unidad de Tecnologías Semánticas

En el entorno de las Tecnologías Semánticas facilitamos el enriquecimiento semántico de la información (basado en lenguajes como RDF, OWL y RIF), con objeto de proporcionar valor añadido a los datos y contribuir al desarrollo hacia nuevos sistemas de información y conocimiento.

En CTIC trabajamos en el diseño y construcción de soluciones a problemas complejos de gestión de la información. Estas soluciones se adaptan de forma no intrusiva a las infraestructuras y sistemas ya existentes, facilitando así la adopción de la tecnología por parte de las entidades destinatarias.

Entre las capacidades de investigación y experiencia actual de esta unidad se encuentran las siguientes temáticas y campos de actuación:

- Soluciones basadas en Linked Data.
- Personalización y recomendación.
- Gestión Documental y búsqueda semántica.

Unidad de Movilidad e Independencia de Dispositivo

En el campo de la Independencia de Dispositivo se investiga en el diseño de arquitecturas software que permitan la creación de aplicaciones preparadas para interactuar con múltiples dispositivos, desde PCs con diversos sistemas operativos hasta las plataformas iOS propias de dispositivos móviles como teléfonos o tablets, pasando por dispositivos con capacidades restringidas como reproductores multimedia o computadores embebidos.

Se mantiene especial atención a la creación de aplicaciones para teléfonos móviles y tablets, por la casuística especial de la movilidad del usuario y del contexto en que éste interactúa con el dispositivo. Se atiende, en especial, a aspectos como el acceso intermitente a la red, la variabilidad en la calidad del acceso a la misma y la adaptación a las distintas capacidades software y hardware de los distintos dispositivos en el mercado.

Entre las capacidades de investigación y experiencia actual de esta unidad se encuentran las siguientes temáticas y campos de actuación:

- Tecnologías móviles/multidispositivo.
- Servicios multimedia interactivos.

Unidad de Tecnologías Emergentes

El trabajo de esta Unidad se orienta hacia la búsqueda de la mejor solución tecnológica posible en el marco de diversos escenarios dentro del ámbito industrial, ya sea desde el punto de vista de la eficiencia como del coste del sistema en sí. Para ello, se trabaja en el diseño de prototipos experimentales que incorporan protocolos de comunicación y equipamiento hardware diverso de bajo coste, con objeto de evaluarlos en campo.

En términos generales, se persigue el diseño de sistemas embebidos o agentes computacionales, sistemas eficientes de comunicaciones entre los mismos y el diseño de algoritmos optimizados para su ejecución en tiempo real, con el objetivo final de dar respuesta a problemáticas industriales concretas de una manera simple y económica.

Entre las capacidades de investigación y experiencia actual de esta unidad se encuentran las siguientes temáticas y campos de actuación:

- Sistemas de identificación, localización y posicionamiento.
- Diseño de sistemas embebidos aplicados al control avanzado.
- Protocolos eficientes de comunicación.

Unidad 4U

La misión de esta Unidad de Investigación es lograr el acceso universal a la información mediante el uso de tecnologías y herramientas útiles, usables y claramente centradas en el usuario y su entorno. El trabajo de esta Unidad se centra en campos y áreas temáticas que conllevan mejoras en la vida cotidiana de las personas mediante la utilización de las tecnologías de la información y la comunicación (TICs).

Entre las capacidades de investigación y experiencia actual de esta unidad se encuentran las siguientes temáticas y campos de actuación:

- Tecnologías de interacción
- eSalud y bienestar
- Entorno inteligentes
- Diseño y evaluación de interfaces de usuario

--

El desarrollo del presente proyecto pertenece al área de I+D+i, concretamente a la unidad de movilidad e independencia del dispositivo para la sección de Tecnologías móviles/multidispositivo.

En dicha sección se trabaja en herramientas que faciliten la creación de aplicaciones cliente-servidor para interactuar con múltiples dispositivos y diversos sistemas operativos, centrando el foco en las tecnologías móviles más populares.

1.3. OBJETIVOS DEL PROYECTO

El objetivo principal del proyecto es la evaluación de la plataforma Titanium. Para ello, se establecerá una serie de criterios de evaluación que constarán de una descripción y una evaluación. La evaluación se realizará en comparación a las aplicaciones desarrolladas en las tecnologías nativas iOS y Android obteniendo así una valoración de su funcionamiento. Se irán anotando diferencias encontradas a lo largo del desarrollo de la aplicación.

Luego para alcanzar el objetivo principal, podemos dividir el proyecto en los siguientes sub-objetivos:

- Descripción de cómo se realizará la evaluación a la que se someterá la herramienta de desarrollo Titanium Studio.
- Criterios a utilizar para la realización de la evaluación.
- Realización de la evaluación y muestra de resultados.
- Conclusiones respecto a las espectativas.

Para llegar a tales conclusiones se desarrollará una aplicación en Titanium que cumplirá con los siguientes sub-objetivos:

- La aplicación permitirá a los usuarios localizar supermercado.
- La aplicación ofrecerá la posibilidad de buscar y añadir productos y promociones existentes en el supermercado a un carro de la compra.
- La aplicación ofrecerá la posibilidad de realizar pedidos, modificarlos y consultar su estado en un momento dado.

CAPÍTULO 2. PLANIFICACIÓN DEL PROYECTO Y RESUMEN DE PRESUPUESTOS.

2.1. PLANIFICACIÓN INICIAL

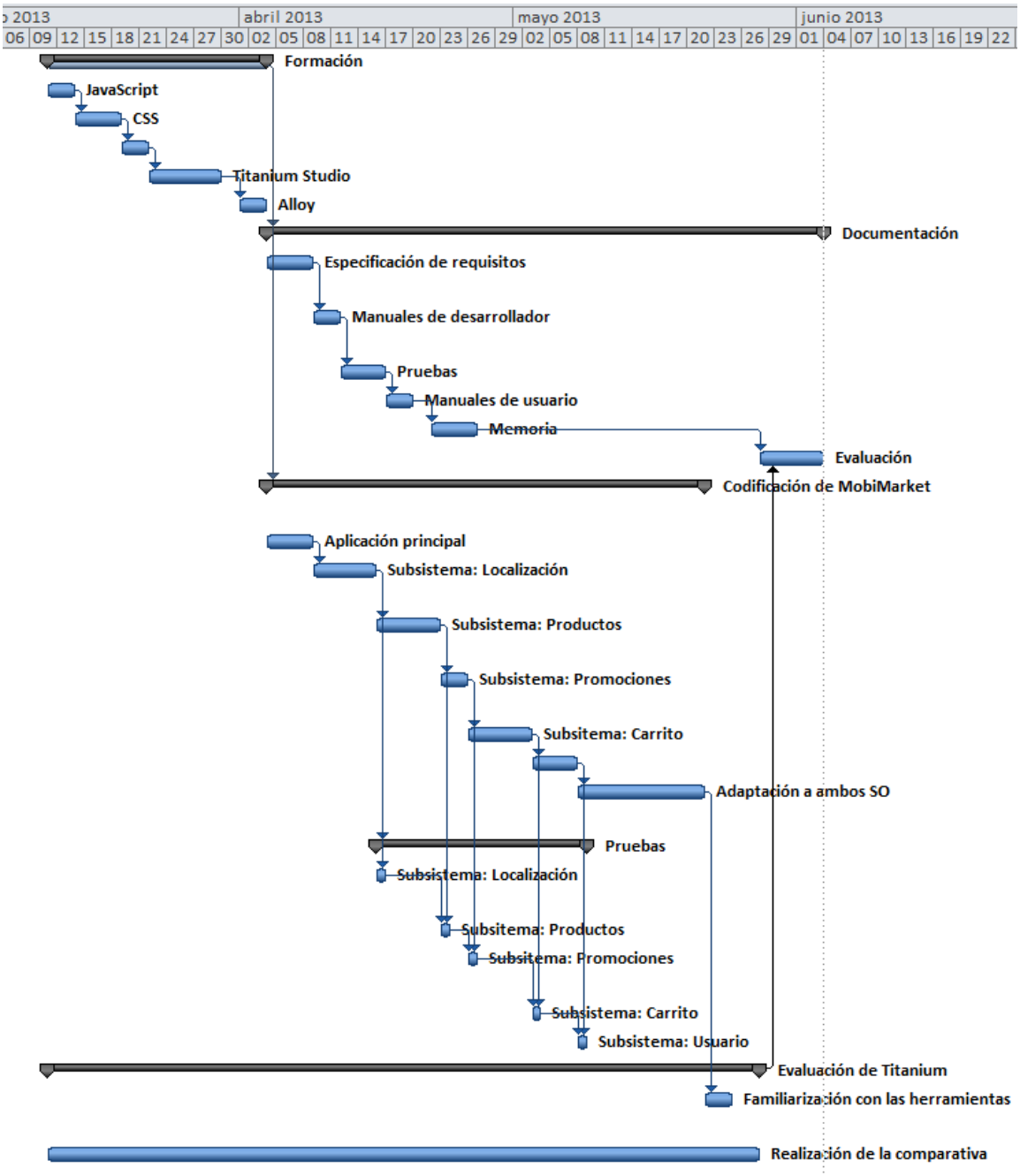


Figura 1

2.2. CALENDARIO REAL EN HORAS POR MES

En esta sección se mostrará una tabla con el número de horas dedicado a la realización del proyecto. A continuación se describe brevemente en que consiste cada una de las fases en las que se han separado los tiempos dedicados.

- Documentación: tiempo dedicado (en horas) al desarrollo del presente documento.
- Codificación: tiempo dedicado (en horas) a generar el código fuente de la aplicación necesaria para la evaluación del entorno de desarrollo.
- Evaluación: tiempo dedicado (en horas) a la toma de notas, comparación de entornos de desarrollo, investigación y generación del código fuente relativo a las pruebas de rendimiento y consumo de recursos.
- Pruebas: tiempo (en horas) dedicado al diseño y ejecución de pruebas.
- Formación: tiempo (en horas) dedicado al aprendizaje del lenguje de programación, el entorno de dearrollo y todo lo necesario para la ejecución de la evaluación de la plataforma Titanium.

Tareas / Meses	Mar-13	Abri-13	May-13	Jun-13	Jul-13	Totales
Documentación	8	11	26	26	44	115
Codificación	29	70	80	0	4	183
Evaluación	6	11	40	12	7	76
Pruebas	0	10	4	3	0	17
Formación	76	18	0	0	0	94
Total	119	126	151	41	55	**485**

Figura 2

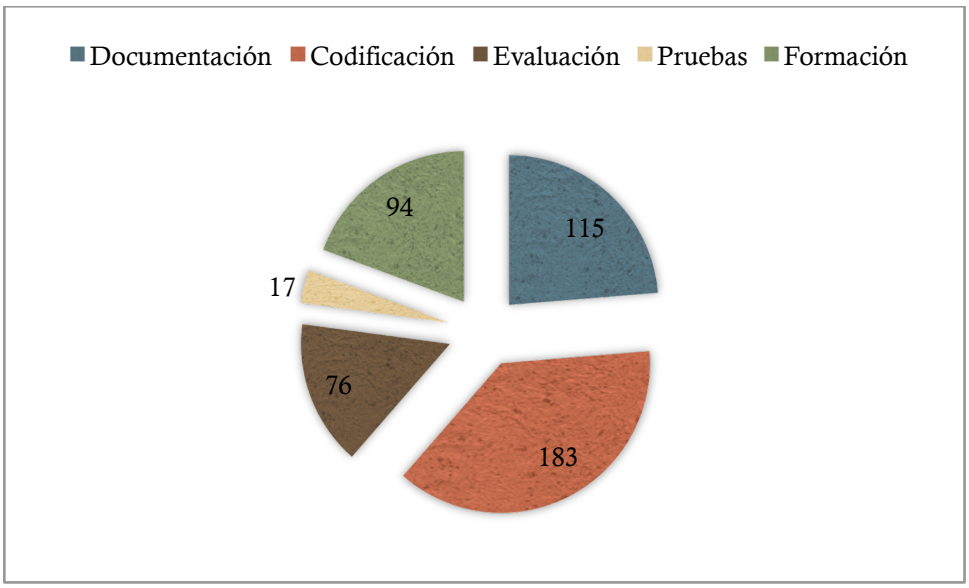


Figura 3

2.3. RESUMEN DEL PRESUPUESTO

En esta sección se añadirá una tabla que describe el coste total que se ha estimado para el desarrollo completo del presente proyecto.

El presupuesto está elaborado con el tiempo invertido y la valoración de la retribución por hora para las distintas actividades establecidas según información obtenida de diferentes fuentes, principalmente bolsas de trabajo online.

Tareas	Recursos en horas	Precio por hora	TOTAL
Pruebas	17 horas	20€	340 €
Formación	94 horas	20€	1.880 €
Documentación	115 horas	15€	1.725 €
Evaluación	76 horas	35€	2.660 €
Codificación	183 horas	20€	3.660 €
TOTAL	**485 horas**		**10.265 €**

Figura 4

CAPÍTULO 3. EVALUACIÓN DE LA HERRAMIENTA TITANIUM.

3.1. TECNOLOGÍA EMPLEADA

En primer lugar se realizará una descripción de la plataforma Appcelerator Titanium y se concluirá con una breve descripción de las aplicaciones y herramientas utilizadas para el desarrollo del presente proyecto.

3.1.1. APPCELERATOR TITANIUM

Titamium es uno de los elementos que forman una gran plataforma de desarrollo de aplicaciones para terminales móviles llama Appcelerator (que también da nombre a la empresa).

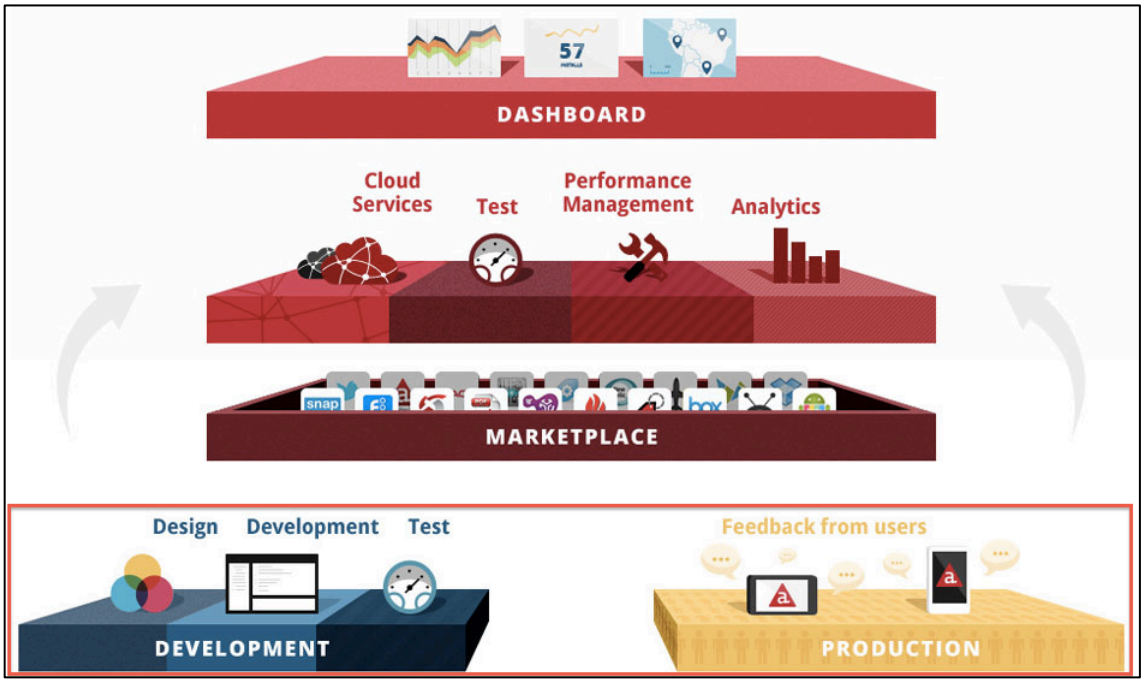


Figura 5

Appcelerator, pone a disposición de su comunidad una plataforma gratuita para desarrollar software para dispositivos móviles permitiendo el intercambio de código, bugs y opiniones, llamada Appcelerator Titanium; también una plataforma de intercambio y venta de módulos reutilizables de código, iconos, frameworks, IDEs, etc., llamada MarketPlace; una plataforma más de Cloud Computing dónde los usuarios podrán ir almacenando sus proyectos, con control de versiones, test por parte de otros usuarios, estadísticas de errores reportados, descargas, etc.; y un último nivel con un DashBoard que permite la comprobación de servicios utilizados de las aplicaciones desarrolladas (por ejemplo, el número de usuarios que se han logueado el martes). El objeto de estudio de este documento es lo que concierne al desarrollo de aplicaciones, marcado con un rectángulo rojo en la Figura 1.

Titanium es una herramienta que trata de simplificar el proceso de desarrollo de aplicaciones para terminales móviles, permitiendo a la comunidad de desarrolladores construirlas, probarlas y publicarlas en las diferentes plataformas existentes independientemente del sistema operativo sobre el que corran. Cada sistema operativo tiene su propio lenguaje de programación y su propia plataforma de publicación de aplicaciones[1] y Titanium permite el desarrollo en un único lenguaje y la publicación automática en las plataformas nativas.

El entorno de desarrollo está basado en el IDE Eclipse, que es el entorno más utilizado por la comunidad de desarrolladores Java, y recibe el nombre de Titanium Studio.

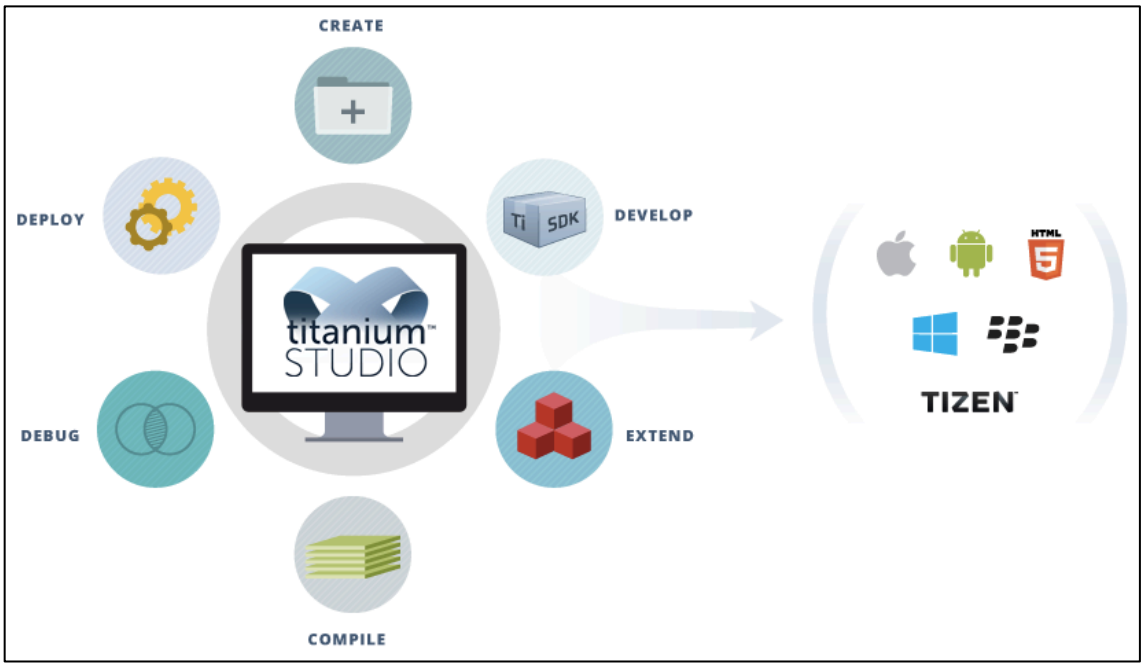


Figura 6

La herramienta trata de solucionar el problema del desarrollo de aplicaciones para múltiples sistemas operativos (iOS, Android, Blackberry, etc.). Para que una aplicación esté disponible en todos los usuarios de dispositivos móviles del mercado, se necesita al menos un desarrollador específico para cada sistema, lo que supone un incremento muy grande en los costes de la creación de una aplicación y sobre todo en el mantenimiento de la misma. Titanium trata de resolver dicha problemática, englobando todo el desarrollo del aplicaciones multiplataforma en un único equipo de desarrolladores que manejen un lenguaje común.

La plataforma Appcelerator Titamium es abierta y permite a través del lenguaje JavaScript construir aplicaciones para diferentes entornos de desarrollo. A día de hoy es capaz de interpretar el código funete JavaScript, para que pueda ser comprendido por los compiladores nativos de cada sistema operativo. La plataforma está formada por un SDK que es código abierto, un IDE de desarrollo basado en Eclipse, un framework MVC llamado Alloy y un conjunto de servicios en la nube.

El SDK utiliza e interpreta ficheros JavaScript (.js). Tiene detrás una comunidad de algo más de cuatrocientos mil desarrolladores y una serie de patrocinadores que trabajan activamente en mejoras del SDK y en frameworks y módulos reutilizables. Esto es interesante para los desarrolladores de

[1] *Por ejemplo, Java y el PlayStore para Android y Objetive-C y App Store para iOS.*

aplicaciones web, ya que JavaScript es el lenguaje más utilizado en el gremio, y tendrían una base sólida para comenzar a desarrollar también aplicaciones para terminales móviles.

Appcelerator Alloy es un proyecto también de código abierto para Titanium. Trata de mejorar la calidad del código a través del uso del patrón arquitectónico MVC. Utiliza ficheros XML para la representación de vistas, JavaScript para la lógica de negocio y CSS para el modelo de datos. También permite la reutilización de código en mayor medida y de una forma mucho más limpia.

También posee un servicio de Cloud Computing, como almacenamiento de fotografías en la nube, notificaciones en tiempo real, pasarelas de pago, etc., que son interesantes mencionar pero que no van a ser utilizadas en el desarrollo de este trabajo. El MarketPlace tampoco será utilizado ni probado en este proyecto.

A lo largo de este trabajo, comprobaremos la efectividad de la herramienta de desarrollo y si consigue realmente evitar el problema del desarrollo multiplataforma, mejorando la productividad y reduciendo los tiempos de desarrollo y los costes que conlleva el proceso de codificación y de mantenimiento del código.

3.1.2. HERRAMIENTAS UTILIZADAS

En esta sección se dará una breve descripción a las herramientas utilizadas para la elaboración del presente trabajo. Tanto para el desarrollo de la aplicación como para la evaluación de la plataforma Titanium.

Appcelerator Titanium Studio

Es el IDE de desarrollo de aplicaciones utilizado para la codificación de la aplicación MobiMarket. Esta basado en el IDE Eclipse y permite la depuración, compilación y despliegue de aplicaciones tanto en emuladores como en dispositivos móviles.

Está directamente ligado con los SDK de las aplicaciones que se quieran desarrollar (en el caso del presente trabajo Android e iOS) y utiliza sus propios compiladores y emuladores para la creación y ejecución de aplicaciones.

El SDK es gratuito.

URL: http://www.appcelerator.com/platform/titanium-studio/

Appcerator Alloy

Alloy es un framework de código abierto creado específicamente para Titanium. Fuerza el uso del patrón arquitectónico Modelo-Vista-Controlador para el desarrollo de aplicaciones.

Utiliza ficheros XML para las vistas, ficheros CSS para el modelado de datos y ficheros JavaScript para la lógica de negocio.

Con el uso del framework el código se puede ver mucho más limpio y ordenado, es mucho más fácil de comprender. Además es mucho más sencilla la reutilización de código.

El framework Alloy ha sido utilizado para el desarrollo de la aplicación MobiMarket Móvil. Todo el código sigue las premisas del framework.

Alloy es gratuito.

URL: http://www.appcelerator.com/platform/alloy/

Bitbucket

Bitbucket es un servicio de alojamiento en la nube, para proyectos que utilizan control de versiones. Los controles de revisiones aceptados por Bitbucket son Mercurial o Git.

El servicio pertenece a la empresa Atlassian Software y ha sido construido en el lenguaje Phython.

Tiene servicios gratuitos y comerciales dependiendo del número de usuario que lo utilicen. Es gratuito hasta cinco usuarios.

En el proyecto de MobiMarket se ha utilizado el servicio gratuito y el control de revisiones Mercurial.

URL: http://www.bitbucket.org

SourceTree

SourceTree es un cliente de interfaz gráfica para el manejo de repositorios git y mercurial.

Simplifica mucho la labor del manejo de repositorios a través de web y mediante línea de comandos.

Se integra completamente con los sistemas de almacenamiento Bitbucket y Github.

Sus principales características es que permite clonar y crear repositorios con facilidad, detectar y resolver conflictos durante la actualización de código fuente y consultar el historial de cambios realizados en un proyecto.

Se ha utilizado SourceTree como GUI para Bitbucket a lo largo del proyecto MobiMarket para la gestión del repositorio y el control de versión.

SourceTree precisa de licencia pero es gratuita.

URL: http://sourcetreeapp.com

SoapUI

SoapUI es una aplicación de código abierto cuya funcionalidad es el testeo de WebServices.

Permite la inspectión de los Web Services, la invocación, el desarrollo, la carga y la depuración de los WebService.

Esta desarrollado en Java Swing. Puede integrarse en los entornos Eclipse y NetBeans.

Durante el proyecto MobiMarket Móvil se ha utilizado su versión gratuita para la comprobar que los WebServices estuvieran activos y principalmente que la comunicación ellos era correcta. Es decir, se depuraban, mediante SoapUI, los ficheros JSON a través de los cuales la aplicación se comunicaba con los WebServices.

URL: http://www.soapui.org

SDK Android

El SDK de Android contiene la API de librerías y herramientas de desarrollo necesarias para construir, testear y depurar aplicaciones que corran posteriormente en el sistema operativo Android.

Para el desarrollo de la aplicación MobiMarket se ha descargado el SDK en su versión 4.2.2 y las versiones antiguas 2.2 y 2.3 necesarias para codificar aplicaciones en Titanium.

El SDK también contiene herramientas como SysTrace, que comprueba los recursos que consume una aplicación en un terminal concreto. También contiene el entorno Eclipse con los recursos mínimos y necesarios para desarrollar aplicaciones Android.

El SDK de Android es gratuito.

URL: http://developer.android.com/sdk/index.html

XCode

Xcode es el IDE de desarrollo de Apple que viene incluido en los sistemas operativos Mac OS X. En las últimas versiones funciona con el entorno gráfico de desarrollo de interfaces Interface Builder.

Además aglutina los compiladores de código C, C++, Objetive-C y Java entre muchos otros. También incluye emuladores para dispositivos móviles de Apple, tanto iPhone como iPad.

Posibilita la compartición de código fuente entre varios equipos a través de la herramienta Bonjour.

En el proyecto MobiMarket es necesario principalmente su compilador para desplegar aplicaciones de Titenium en dispositivos móviles de Apple. Se ha utilizado su versión 6.1 para el desarrollo de la aplicación.

El entorno de desarrollo es gratuito, pero la licencia para desplegar aplicaciones es de 99$ al año. La licencia ha sido facilitada por CTIC para el desarrollo del proyecto.

Robotium

Robotium es un framework creado para implementar test automáticos de aplicaciones Android. Se pueden testear cualquier aplicación que funcione a través de GUI en Activities y Dialogs (componentes Android para el desarrollo de aplicaciones).

Los test de Robotium se ejecutan como test JUnit.

La licencia necesaria para utilizar este framework es Apache 2.0 que es gratuita.

En el proyecto MobiMarket Robotium se ha utilizado para tomar muestras de tiempo de ejecución en la aplicación nativa y en la aplicación desarrollada en Titanium.

URL: http://code.google.com/p/robotium/

3.2. INTRODUCCIÓN A LA EVALUACIÓN

En el presente documento se introduce al método de evaluación empleado para la evaluación de la plataforma de desarrollo de aplicaciones móviles Titanium.

3.2.1. DESCRIPCIÓN DE LA EVALUACIÓN

Para someter a evaluación la plataforma Appcelerator Titanium, se ha decidido realizarlo por comparación con las aplicaciones desarrolladas mediante las tecnologías nativas Android e iOS facilitadas por el CTIC. Para organizar la evaluación se han recurrido a los atributos de calidad de la norma ISO 9126. Estos atributos se han adaptado al ámbito del presente estudio y se han extendido en forma de una lista de cuestiones que serán utilizadas para la evaluación.

3.2.2. DESCRIPCIÓN DE LOS CRITERIOS DE EVALUACIÓN.

Los atributos de calidad de la norma ISO 9126 adaptados al contexto del presente documento son los siguientes:

- Funcionalidad: se pretende comprar la funcionalidad básica del entorno de desarrollo Titanium Studio respecto a los entornos nativos, Xcode y Android. También se trata de evaluar el proceso de compilación y depuración de una aplicación a través del entorno así como el grado de dificultad que conlleva el aprendizaje del lenguaje para codificar aplicaciones en Titanium. Las cuestiones a resolver en cuanto a funcionalidad son las siguientes:
  - ¿Es difícil tener un nuevo proyecto de Titanium funcionando en un dispositivo móvil?
  - ¿Cómo es el proceso de compilación y depuración respecto a los entornos de desarrollo XCode y Android?
  - ¿Qué grado de dificultad conlleva programar una aplicación en Titanium?

- Fiabilidad: se pretende comprar el parecido de una aplicación construida en Titanium con el que se obtiene desarrollando a través de los entornos nativos. Además se precisa evaluar los fallos detectados durante el desarrollo de la aplicación en el entorno Titanium Studio. Las cuestiones a resolver en cuanto a fiabilidad son las siguientes:
  - ¿Cuántos interfaces se pueden utilizar en la creación de una aplicación en Titanium respecto a los entornos nativos?
  - ¿Se pueden generar aplicaciones en Titanium exactamente igual a las nativas?
  - ¿Se han encontrado fallos en la herramienta Titanium Studio a lo largo del desarrollo de la aplicación?

- Eficiencia: se pretende comparar el tiempo que tarda en compilar, desplegar y ejecutarse una aplicación desarrollada en Titanium respecto a los que tarda en los entornos nativos, así como los recursos hardware que consume dicha aplicación dependiendo en que tecnología se haya desarrollado. Las cuestiones a resolver en cuanto a eficiencia son:
  - ¿Cuánto tiempo tarda en compilar una aplicación en Titanium respecto a las aplicaciones nativas?
  - ¿Cuánto tiempo tarda en desplegarse una aplicación en Titanium respecto a las aplicaciones nativas?
  - ¿Cuánto tiempo tarda en ejecutarse alguna funcionalidad de una aplicación en Titanium respecto a las aplicaciones nativas?
  - ¿Cuántos recursos de un dispositivo móvil consume una aplicación generada en Titanium respecto a las aplicaciones generadas en Xcode o Android?

- Usabilidad: se pretende comprar las facilidades que la plataforma Titanium pone a disposición de su comunidad de desarrolladores respecto a la que Google o iOS pone a disposición de la suya. Principalmente en cuanto a documentación y a facilidad de manejo de los entornos de desarrollo. Las cuestiones a resolver en cuanto a usabilidad son:
    - ¿Existe una documentación completa a disposición de la comunidad de desarrolladores de Titanium?
    - ¿Es fácil crear una nueva aplicaciónn en Titanium Studio?

- Mantenibilidad: se pretende comparar las licencias necesarias para el desarrollo de aplicaciones en Titanium respecto a Android e iOS. También la cantidad y calidad de las actualizaciones recibidas durante el desarrollo de la aplicación. Las cuestiones a resolver en cuanto a mantenibilidad son las siguientes:
    - ¿Es necesaria algún tipo de licencia para desarrollar aplicaciones en Titanium? ¿Qué diferencias son encontradas respecto a las licencias nativas?
    - ¿Cuántas actualizaciones se han recibido durante el desarrollo de la aplicaciónn?
    - ¿Las aplicaciones mejoran realmente el entorno de desarrollo?

- Portabilidad: se pretende comprar la facilidad de instalación del entorno de programación de Titanium Studio respecto a Eclipse con el SDK Android o Xcode, así como su compatibilidad con los diferentes sistemas operativos existentes. También se pretende evaluar la existencia de librerías externas y las posibilidades del uso de paquetes ya desarrollados por otros programadores. Las cuestiones a resolver en cuanto a portabilidad son las siguientes:
    - ¿Es fácil instalar el entorno de desarrollo Titanium Studio?
    - ¿Puede instalarse el entorno de desarrollo Titanium Studio en diferentes sistemas operativos?
    - ¿Se pueden utilizar librerías externas en Titanium? ¿Son difíciles de importar?

Cada una de las cuestiones de los criterios de calidad a los que se someterá la plataforma Titanium podrán ser respondida de acuerdo a los siguientes valores:

- "Similar al nativo": las apreciaciones entre el desarrollo de la Aplicación en Titanium Studio y en su herramienta nativa son prácticamente iguales.

- "Peor al nativo": se encuentran diferencias reseñables entre el desarrollo de la aplicación nativa y el desarrollo de la aplicación en Titanium Studio.

- "Sólo en nativo": se encuentran amplias diferencias entre el desarrollo de la aplicación nativa y el desarrollo de la aplicación en Titanium Studio, o no puede llevarse a cabo el desarrollo nativo en Titanium.

- "No evaluado": por falta de tiempo o de recursos la cuestión no ha podido ser evaluada.

3.2.3. REALIZACIÓN DE LA EVALUACIÓN

Los atributos serán evaluados a través de las cuestiones que serán respondidas a lo largo del desarrollo de la aplicación para terminales móviles MobiMarket. Se tendrán muy presentes las cuestiones a lo largo de todo el desarrollo del software, y se realizarán anotaciones continuas de forma que se pueda dar respuesta a las cuestiones de la forma mas completa y objetiva posible.

3.3. CRITERIOS DE EVALUACIÓN

En esta sección se definirán los seis criterios de evaluación de la calidad de la norma ISO 9126 [2]adaptados al contexto del proyecto. Cada uno de dichos cuestión que define los criterios se basará en una descripción de la evaluación/comparación y un resultado.

Todas las cuestiones se recogerán a modo de resumen en una tabla que se presenta en el siguiente punto del presente documento.

3.3.1. FUNCIONALIDAD

Se entiende por funcionalidad, la aceptación funcional que poseen las actividades que son realizadas para conseguir el despliegue de una aplicación en un dispositivo móvil. El proceso se basa en el inicio de un nuevo proyecto, la realización de test, el proceso de compilación y la puesta en producción en un terminal móvil.

¿Es difícil tener un nuevo proyecto de Titanium funcionando en un dispositivo móvil?

DESCRIPCIÓN

Para resolver esta pregunta, se explorarán las posibilidades de insertar una nueva aplicación en un terminal móvil.

EVALUACIÓN

Lo primero que se tiene que realizar es la configuración del entorno a los sistemas en los que se desea desplegar aplicaciones. En el caso de MobiMarket se desplegará la aplicación en terminales iPhone y en terminales Android.

El proceso de configuración pasa por ser similar al que un desarrollador de iOS o Android debe realizar. Es indispensable la descarga de los SDK correspondientes. Se indicará posteriormente a la herramienta Titanium Stduio cuál es la ubicación de dichos SDK.

En el caso de iOS habrá que validar los certificados de desarrollo en el xCode de la misma forma que lo realizaría un desarrollador en iOS.

Por lo tanto, tener un proyecto funcionando en un terminal para su testeo se realiza de una forma **exactamente igual** que en los sistemas nativos.

¿Cómo es el proceso de compilación y depuración respecto a los entornos de desarrollo XCode y Android?

DESCRIPCIÓN

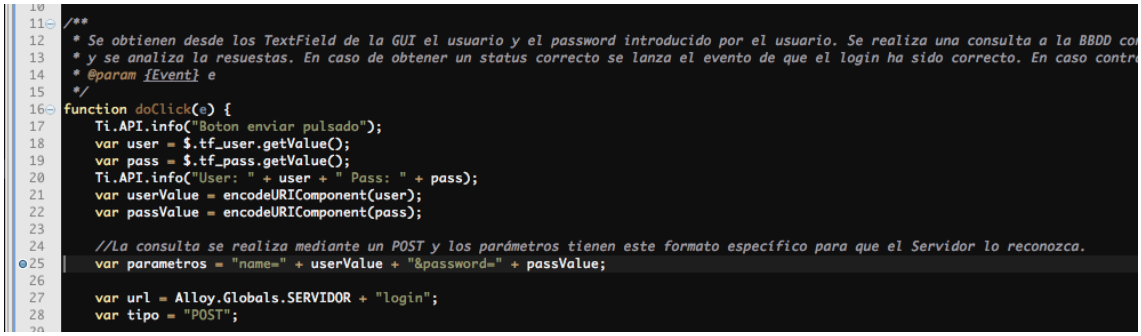
Se evaluará el proceso de compilación y depuración a través de la herramienta de desarrollo Titanium Studio. Se hará lo propio en el IDE Eclipse y en XCode para aplicaciones sencillas. Se compararán las apreciaciones del autor del uso en los tres entornos de desarrollo.

EVALUACIÓN

El entorno de compilación y depuración también es muy similar al de la herramienta XCode y completamente igual al del IDE Eclipse.

[2] *[Introducción a la norma ISO 9126] http://normaiso9126.blogspot.com.es*

Como hemos podido ver en la anterior cuestión, se ejecutan aplicaciones de forma muy similar. Para realizar una depuración, tendremos que establecer puntos de ruptura, que simplemente se colocan haciendo un clic de ratón en el número de línea dónde se desee que se detenga la ejecución.



```
/**
 * Se obtienen desde los TextField de la GUI el usuario y el password introducido por el usuario. Se realiza una consulta a la BBDD co
 * y se analiza la resuestas. En caso de obtener un status correcto se lanza el evento de que el login ha sido correcto. En caso contr
 * @param {Event} e
 */
function doClick(e) {
    Ti.API.info("Boton enviar pulsado");
    var user = $.tf_user.getValue();
    var pass = $.tf_pass.getValue();
    Ti.API.info("User: " + user + " Pass: " + pass);
    var userValue = encodeURIComponent(user);
    var passValue = encodeURIComponent(pass);

    //La consulta se realiza mediante un POST y los parámetros tienen este formato específico para que el Servidor lo reconozca.
    var parametros = "name=" + userValue + "&password=" + passValue;

    var url = Alloy.Globals.SERVIDOR + "login";
    var tipo = "POST";
```

Figura 7

En la figura 7, se puede comprobar como un punto de ruptura es colocado en la línea 25 del controlador "login" de la aplicación MobiMarket. Cuando realizamos una depuración, se insertará un nombre de usuario y una contraseña desde el interfaz de dicho controlador, y se pulsará el botón de aceptar para que se ejecute el método dónde tenemos el punto de ruptura.

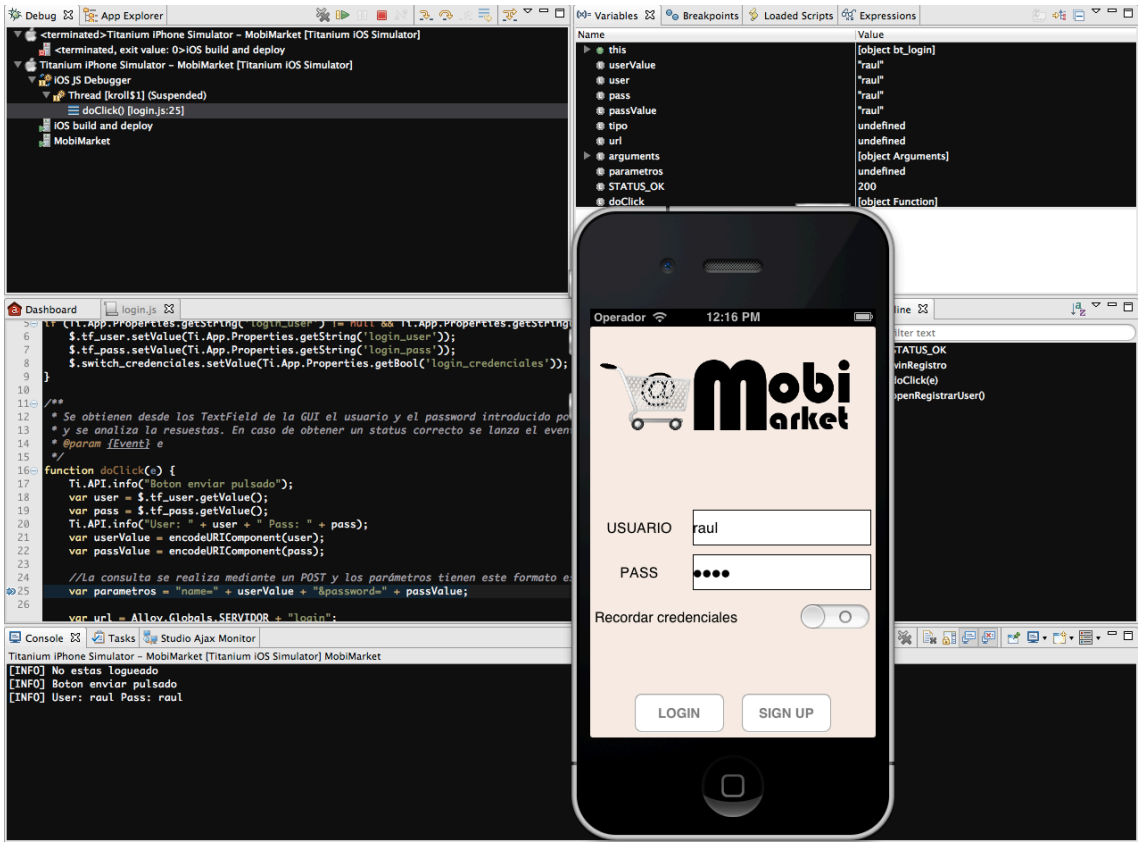


Figura 8

En la figura 8, se puede observar la pantalla de depuración una vez se ha llegado al punto de ruptura. Esta pantalla de depuración es exactamente igual al del IDE Eclipse, tanto para el desarrollo de aplicaciones Java, como para el que atañe a este trabajo de desarrollo de aplicaciones en Android.

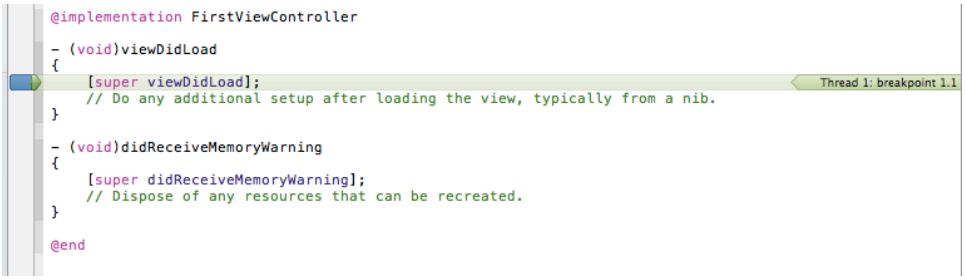


Figura 9

En la figura 9, se puede observar el proceso de depuración en XCode, que también es similar al del IDE Eclipse.

La depuración no funciona correctamente con las consultas asíncronas a la base de datos, pero es un problema menor para aplicaciones de una complejidad menor.

Por lo tanto podemos concluir que el proceso de depuración y compilación de una aplicación en Titanium es **similar a los entornos nativos de Android e iOS.**

¿Qué grado de dificultad conlleva programar una aplicación en Titanium?
DESCRIPCIÓN

Esta pregunta no tiene una respuesta fácil, y obviamente es imposible de responder de forma objetiva. Pero puede resultar interesante para los desarrolladores de aplicaciones móviles, y no puede quedar, en opinión del autor, sin cubrir. Se basa en las impresiones de la curva de aprendizaje del autor desde el comienzo hasta el fin del desarrollo de la aplicación MobiMarket.

EVALUACIÓN

El lenguaje utilizado para el desarrollo de aplicaciones en Titanium es JavaScript, en Android Java, y en iOS Objetive-C. Al autor del presente trabajo le ha parecido relativamente sencillo aprender a programar en JavaScript, ya que se tenían conocimientos previos de Java. Sin embargo, Objetive-C es considerado por el autor como un lenguaje difícil de aprender a pesar de tener conocimiento previos de C. Así que esta pregunta lleva un índice demasiado elevado de subjetivismo. Se tratará de responder a la pregunta con las experiencias del autor al iniciar el desarrollo de la aplicación MobiMarket.

En un primer momento, el desarrollo de una aplicación para Titanium tenía una apariencia engorrosa y sucia. No se utilizaba ningún patrón arquitectónico para su diseño. Tanto los GUI como la funcionalidad de la aplicación se codificaron en los mismos ficheros de JavaScript sin existir una forma ordenada de separarlos. Las apreciaciones, durante los primeros pasos con Titanium fueron malas. Se podían realizar aplicaciones sencillas en poco tiempo para varios sistemas operativos, pero la calidad y reutilización del código eran prácticamente nulas. E incluso imposibilitaba realizar una aplicación, con relativa complejidad como la del proyecto que aquí se describe. Demasiado código en un mismo fichero, imposibilidad de encontrar errores de codificación, etc. En definitiva, este proyecto no se podía llevar a cabo, en el tiempo del que se disponía para realizarlo.

Investigando más a cerca de la herramienta, se descubrió que existían un framework, que llevaba unos meses a disposición de la comunidad de desarrolladores. Dicho framework obliga a los desarrolladores a utilizar una variante del patrón arquitectónico modelo-vista-controlador. Las GUI se diseñan en un fichero XML, el modelado de datos en un fichero CSS y la lógica de negocio en ficheros JavaScript.

Con la utilización de dicho framework, resulta bastante más sencillo, limpio y ordenado la realización de un proyecto. El proceso de programación es bastante similar al que se sigue para codificar una solución en Android. Ficheros XML representan las interfaces de usuario en ambos casos y ficheros JavaScript para Titanium y Java para Android representan la lógica de negocio.

promociones.js | promociones.tss | promociones.xml

```
<Alloy>
    <View class="container" id="view_promociones">
        <View class="container" layout="vertical">
            <View height="10dp" />
            <Label id="lb_nombrePromocion" height="auto" width="auto" clickable="false" textAlign="center"/>
            <View height="10dp" />
            <ScrollableView id="promocionesView" showPagingControl="true" backgroundColor="black" width="250dp" height="250dp" clickable="
            <View height="10dp" />
            <Label id="lb_descripcionPromocion" height="auto" width="auto" clickable="false" textAlign="center"/>
            <Label id="lb_precioPromocion" height="auto" width="auto" clickable="false" textAlign="center"/>
        </View>
        <Button id="bt_addPromocion" bottom="5dp" right="5dp" backgroundImage="/common/images/addButton.png" height="50dp" width="50dp" />
    </View>
    <ActivityIndicator id="activityIndicator" message="Loading..."/>
</Alloy>
```

Figura 10

En la figura 10, se puede observar la vista del controlador promociones, en el que se definen una serie de componentes gráficos para la GUI de la aplicación.

promociones.js | promociones.tss | promociones.xml

```
".container": {
    backgroundColor : Alloy.Globals.BACKGROUND_COLOR
}

"Label" : {
    font : {
        fontSize : '15dp',
        fontFamily : 'Special Elite'
    },
    color : 'black',
    textAlign : 'center'
}

"#lb_precioPromocion" : {
    font : {
        fontSize : '20dp',
        fontFamily : 'Special Elite'
    },
    color : 'green'
}
```

Figura 11

En la figura 11 se observa el modelado de datos, en el que se trabaja con la alineación del texto, el tamaño, la familia y colores entre otras cosas.

promociones.js | promociones.tss | promociones.xml

```
1  //CARGA DE PROMOCIONES DESDE BBDD NOSQUL
2  /**
3   * Se recibe un evento de foco sobre la pestaña promociones, entonces, se realiza una consult
4   * existentes en el servidor que estén activas. Dicho JSON parseado (convertido en array) se
5   * Se utiliza otro método auxiliar y no se realiza en el método onload, porque una vez termin
6   */
7  Ti.App.addEventListener('cargaPromociones', function(e) {
8      Ti.API.info("Recibo evento carga promociones");
9      $.activityIndicator.show();
10
11     var getAllCategories = Ti.Network.createHTTPClient({
12         username : Alloy.Globals.USERNAME,
13         password : Alloy.Globals.PASSWORD,
14         timeout : 3000,
15         onerror : function(e) {
16             alert("Error en CargaPromociones: " + e.error);
17         },
18         onload : function(e) {
19             cargaPromociones(JSON.parse(this.responseText));
20         }
21     });
22     //Se prepara la conexión con el tipo (GET, PUT o POST) y la URL
23     getAllCategories.open("GET", Alloy.Globals.SERVIDOR + "promos/active");
24     //Se lanza la petición
25     getAllCategories.send();
26 });
```

Figura 12

En la figura 12, se puede comprobar la parte relativa a la lógica de negocio. Es una parte del fichero JavaScript del controlador de promociones dónde se realiza una consulta a la BBDD.

Otro punto en contra para Titanium, aún con el uso de Alloy, es que no existe una GUI para el desarrollo de interfaces. Se realizan únicamente a través de un fichero XML, lo que complica bastante el diseño de las mismas. Existen algunas versiones de pago, que no se han probado, pero que no reciben buena crítica por parte de la comunidad de desarrolladores. El diseño gráfico de interfaces es bastante más complicado en Titanium, a día de hoy que en Android o iOS.

Por lo tanto, y siempre en opinión del autor, el desarrollo de aplicaciones, con cierto punto de complejidad, antes de la llegada de Alloy era inviable. Entendiendo Titanium y Alloy como un todo, se puede concluir que es bastante parecido al desarrollo de aplicaciones en Android en cuanto a modelo de codificación, sin embargo ofrece muy pocas posibilidades de diseño de interfaces gráficas, lo que complica muchísimo realizar aplicaciones tan usables como los sistemas nativos. Se concluye que es **peor al nativo.**

3.3.2. FIABILIDAD

En este estudio, se entiende por fiabilidad, la precisión con la que se pueden realizar aplicaciones en un entorno nativo como iOS o Android en Titanium. Es decir, que interfaces gráficos se pueden representar y cuáles no. Si funcionan de igual manera en un entorno u otro.

También el número de fallos que se han producido en el IDE de desarrollo a lo largo de la ejecución del proyecto.

¿Cuántos interfaces se pueden utilizar en la creación de una aplicación en Titanium respecto a los entornos nativos?

DESCRIPCIÓN

Este es, con total seguridad, el apartado más complejo y quizás mas importante del estudio. El frontend, es realmente dónde se define las posibilidades de replicación de una aplicación nativa en

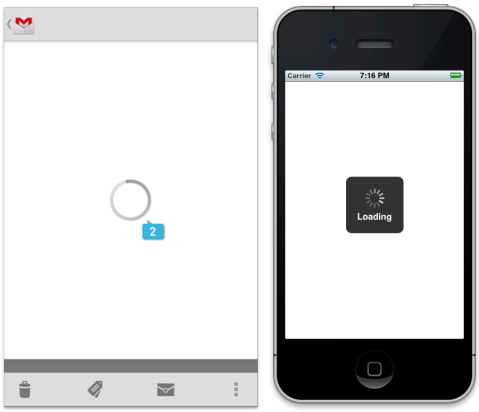
Titanium. Se realizará un repaso a todos los elementos gráficos utilizados en Titanium para tratar de replicar la aplicación nativa.

EVALUACIÓN

En primer lugar, y algo muy importante, es que sólo existen tres tipos de layout en Titanium para el desarrollo de interfaces gráficas. Un layout vertical, uno horizontal y uno libre. Además no existe ninguna GUI realmente funcional que permita el desarrollo gráfico de dichas interfaces de usuario. Estos tres simples layouts distan mucho de los *GridLayout* o *TableLayout* que ofrecen los sistemas nativos. Hay que tener en cuenta, que se realizan aplicaciones para numerosos dispositivos con diferentes resoluciones y diferentes tamaños de pantalla. Con lo cuál, es muy complicado diseñar interfaces que satisfagan las necesidades de cada dispositivo móvil sin un layout en el que encajar las vistas.

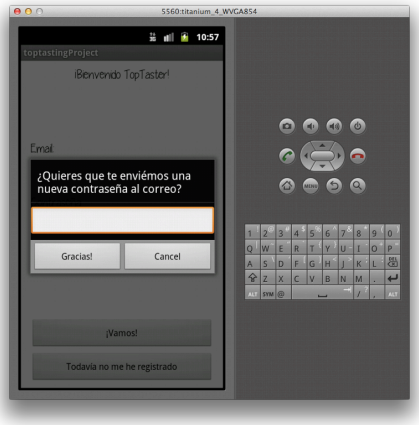
ActivityIndicator

El activityIndicator es la circunferencia de carga que acompaña a los sistemas Android e iOS. Se han tenido problemas a la hora de escoger un estilo de uso en Android. Al principio del desarrollo de la aplicación funcionó sin ningún problema en ambos sistemas operativos. Con una actualización en el dispositivo Android surgió un error en tiempo de ejecución. En la aplicación MobiMarket, se utiliza el ActivityIndicator cada vez que se realiza una consulta a la BBDD, hasta que se recibe una respuesta.



AlertDialog

El AlertDialog es el típico mensaje de alerta que suele ser utilizado para lanzar un aviso. Funciona correctamente en iOS e igual que en los sistemas nativos. Quedaba fuera del alcance del proyecto MobiMarket el uso de AlertDialog personalizados. Pero se ha probado uno con campo de texto y botones que también ha funcionado correctamente.



Button

Los botones también funcionan correctamente tanto en Android como en iOS con una salvedad. En Android no se puede modificar su estilo, sólo existe un tipo de botón que se pueda representar en Android. En iOS sin embargo puede modificarse su estilo. Se han probado botones en anotaciones de mapa, y en la barra de menú de la aplicación. En ambos casos el funcionamiento es correcto. En Android, a la hora de intentar insertar estilos o botones en la barra de aplicación provocaba un fallo en el sistema.

ButtonBar

Las barras de botones únicamente funcionan en iOS. En Android se deben generar de forma manual con una vista y una serie de botones, lo que, aún siendo posible, complica mucho su creación. En el proyecto MobiMarket se ha decidido implementar la posibilidad de logout mediante un menú contextual.

EmailDialog

Al igual que los alert personalizados, no ha sido necesario implementar un diálogo para enviar correos. Sin embargo, se ha probado de todas formas la funcionalidad, mientras se investigaba el funcionamiento de los componentes.

ImageView

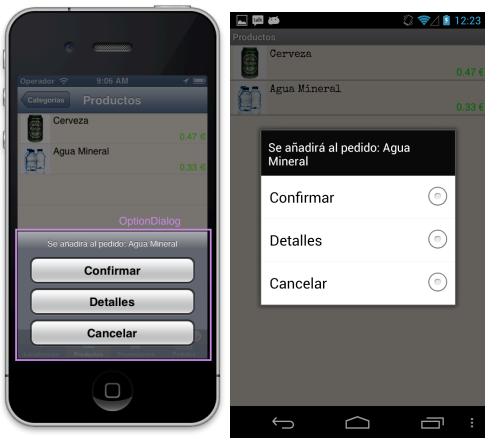
La vista de imagen sirve para representar una imagen o un conjunto de ellas. Admite los formatos PNG y JPEG. No admite GIF. Por lo tanto para representar una animación se deberá introducir una sucesión de imágenes. Salvo por el detalle de las animaciones funciona correctamente tanto en iOS como en Android.

Label

Las etiquetas tienen un funcionamiento completamente igual al que tienen en los sistemas nativos. Funcionan correctamente, y también se pueden aplicar los esteticismos que en iOS o Android, como añadir bordes, redondearlos, etc. Además también se pueden hacer *clickables*.

OptionDialog

Los menús de opción funcionan también de la misma forma en Titanium que en los sistemas nativos. Durante el desarrollo de MobiMarket, se han necesitado menús de opciones con una serie de botones que permitan seleccionar una funcionalidad u otra de la aplicación. Los resultados son equivalentes al nativo.

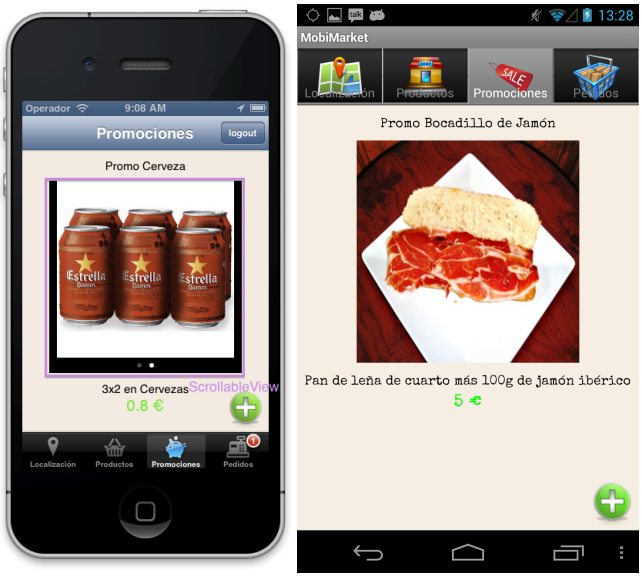


Picker

Los pickers no funcionan correctamente. Si bien es cierto, que hay que mencionar que también tienen incluidos una serie de estilos que facilitan las cosas (como por ejemplo un picker para fecha predefinido), su posicionamiento y tamaño en pantalla no son correctos. Tampoco permite la redimensión del interfaz, lo que provoca, que en dispositivos de mucha resolución su tamaño sea diminuto y no sea demasiado usable. Además para recuperar los datos del picker no se realiza de forma trivial y hay que utilizar una serie de "trucos" para que no se sobrepongan gráfica y lógicamente unas columnas del picker encima de otras. Es necesario, añadir el picker en una ventana nueva con el fondo en un color diferente al transparente. Es decir, para tener un componente picker en Titanium no es sólo colocar el componente en el Canvas, sino que hay que realizar más operaciones. En MobiMarket, tal y como se ven en las imágenes se han utilizado para indicar la fecha de nacimiento y el sexo durante el formulario de registro de nuevos usuarios. En Android se han decidido utilizar menús de opciones para el mismo fin, ya que surgían muchos problemas a la hora de implementarlos.

ScrollableView

Este componente sirve para representar una serie de vistas colocadas en páginas. En MobiMarket se ha utilizado para representar promociones. Las promociones se muestran a los usuarios como una sucesión de imágenes. El funcionamiento es correcto e igual al de los sistemas nativos.



ScrollView

Las vistas con scroll son utilizadas en Titanium principalmente para añadir componentes en su interior, y cuando no cogen en la pantalla del dispositivo, permitir verlos a través de un scroll vertical u horizontal. Existe una problemática, y es que en Android, las vistas que funcionan como contenedor implementan el scroll de forma automática. Además no se puede desactivar de ninguna forma (sí incluyen un atributo scrollable, que es un booleano, pero no funciona actualmente). A lo largo del desarrollo de MobiMarket se han tenido muchos problemas a la hora de implementar una pantalla que tuviera componentes TextField. Cuando un usuario se dispone a escribir texto en alguno de dichos componenes, el teclado del sistema operativo aparece en pantalla. Ahí surge la problemática. Si no tenemos ScrollView, funciona correctamente en Android pero no funciona en iOS. Android lo tiene implementado de serie e iOS no. Si ponemos el ScrollView en iOS funciona correctamente pero en Android se sobreponen unos componentes a otros. Luego no hay forma de que funcione correctamente.

Durante el desarrollo de la aplicación de Mobimarket, se implementa el interfaz gráfico en tiempo de ejecución de forma que las vistas se incluyen dentro de un ScrollView en caso de iOS y de un View genérico en caso de Android. Esto rompe con el patrón arquitectónico de Alloy y se vuelve a programar en JavaScript "puro", lo que obliga a tener un código sucio pero, al menos, funcional.

Además, existe un error al introducir una ImageView dentro del ScrollView en el sistema operativo Android. En cuanto se abre una ventana que tenga el combo anteriormente mencionado fuerza la aplicación a cerrarse. Se ha probado a programar algo similar en nativo, en una aplicación muy sencilla, y funciona correctamente, luego es un error de Titanium. Se ha avisado en la comunidad de desarrolladores y se ha abierto un ticket para que se solucione el problema en siguientes versiones de la plataforma.

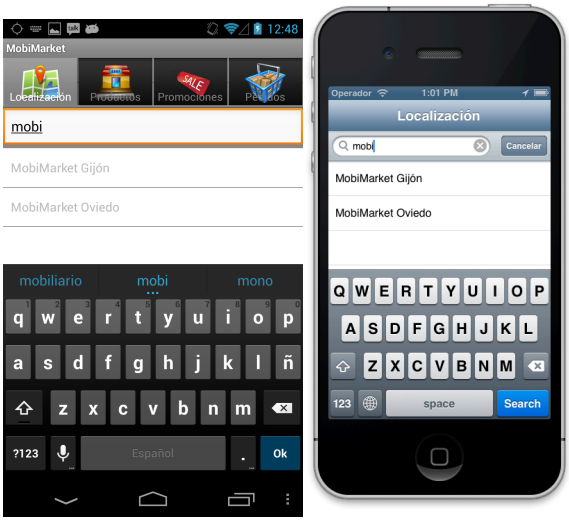
El ScrollView es un componente a mejorar en Titanium ya que, da muchísimos problemas su uso.

SearchBar

La barra de búsqueda era un componente muy difícil de tratar. Toda barra de búsqueda tiene un botón para cancelar que causaba una excepción en Android que forzaba la aplicación a ser cerrada. Entonces se decidió eliminar el botón de cancelar para la versión Android y dejarlo para la versión de iOS. Realizar estas distinciones entre un sistema operativo u otro no es del todo trivial y lleva un tiempo extra de desarrollo.

Hay que mencionar que una de las actualizaciones de la plataforma durante el desarrollo de MobiMarket solucionó el problema en Android.

La barra de búsqueda suele incrustarse dentro de una tabla, y filtra los resultados correctamente de acuerdo a lo que se esté escribiendo en ella. En el proyecto MobiMarket se utiliza para buscar supermercados. Y su funcionamiento es correcto.



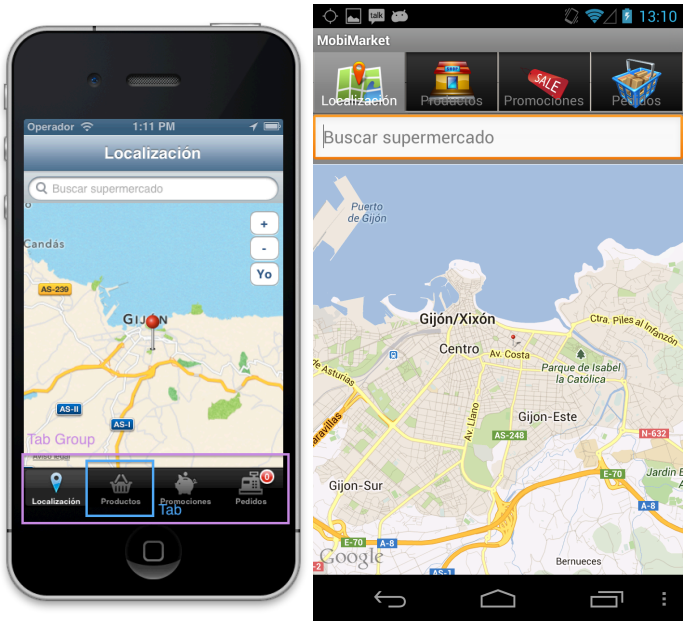
Switch

El Switch es una especie de RadioButton con un valor booleano. Es utilizado en la pantalla principal de MobiMarket para preguntar a los usuarios si se desea salvar los credenciales para la próxima ocasión que se tengan que autentificar.

El funcionamiento es exactamente igual que en los sistemas nativos.

Tab

La aplicación consta de cuatro pestañas que separan sus diferentes secciones. En un primer momento se han tenido problemas más relacionados con el framework Alloy que con Titanium en sí, pero su funcionamiento es correcto.



TabGroup

El componente de grupo de pestañas, sirve simplemente para agrupar pestañas y que el desarrollador pueda comunicarse con cada una de las pestañas desde cualquier parte de la aplicación. Su funcionamiento es exactamente igual que en los sistemas nativos.

TableView

Las tablas tienen un funcionamiento correcto. Hay salvedades, como que el tipo de fuente por defecto en Android es demasiado diminuto y hay que cambiarlo siempre que se quiera mostrar algo en una tabla.

El scroll de las tablas en Android no funciona correctamente y tampoco la redimensión dinámica. Es decir, si hay que introducir nuevas filas a una tabla, no sirve únicamente con añadir nueva información, sino que manualmente hay que introducir el crecimiento de la tabla en píxeles (o densidad de pixeles). Aunque también es un bug, en el que Appcelerator está trabajando para resolver en futuras versiones.

TableViewRow

Las filas que forman una TableView también son un componente codificable, al igual que en los sistemas nativos. No tiene un funcionamiento correcto. Las posibilidades que se ofrecen de insertar imágenes y texto en una fila de tabla no funcionan ni en iOS ni en Android correctamente. En algunos casos no se muestran las imágenes y en otros se fuerza a que se cierre la aplicación.

Para MobiMarket era imprescindible el uso de filas con imágines y texto de diferentes formas, colores y tamaños. Se consiguió realizar creando una vista general dentro de cada fila con varios componentes como ImageView y Label.

Los botones que indican que la fila tiene más información ampliable (hasChildButtons) funcionan correctamente y tienen un diseño exactamente igual al de los sistemas nativos.

TableViewSection

Las diferentes filas de una tabla se pueden agrupar por secciones. No es una funcionalidad necesaria para MobiMarket, pero se ha probado y funciona exactamente igual que en los sistemas nativos. Su diseño también es igual.

TextArea

Al igual que las secciones de tablas, no es necesario un área de texto para el proyecto MobiMarket, sin embargo se ha probado y su funcionamiento es correcto, es exactamente igual al nativo.

TextField

Los campos de texto son utilizados en diferentes apartados del proyecto MobiMarket, como en los apartados de autentificación y registro. Su funcionamiento es exactamente igual que el nativo y su estética también.

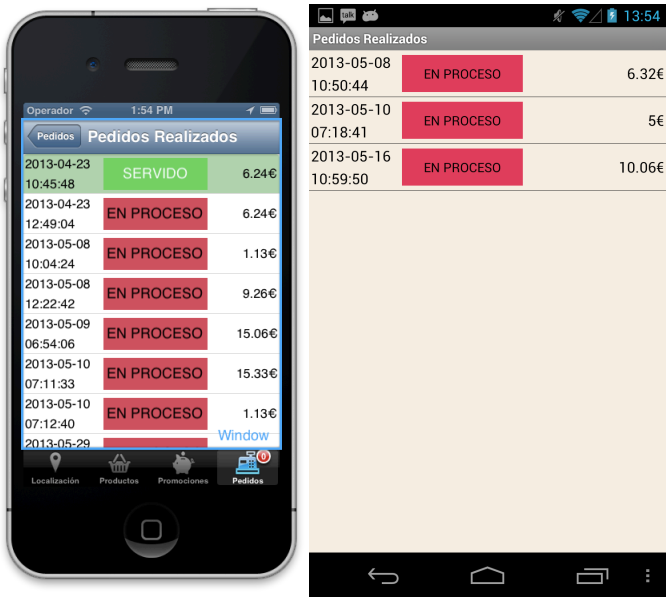
View

El componente genérico de vista es utilizado con frecuencia a lo largo del proyecto MobiMarket principalmente por dos motivos. En primer lugar, al sólo disponer de dos tipos de layouts (horizontal y vertical), para separar un componete de otro se utilizan entre medias views vacíos con un tamaño determinado. El otro motivo es para crear las filas de las tablas de forma personalizada, al tener problemas para crearlas de la forma que indica la documentación de Titanium. Se utilizaba una vista genérica en la que se iban introduciendo los componentes que deseábamos que tuviera cada fila. Es decir, se utilizaba una view como un layout interno de cada fila de forman las diferentes tablas que se han utilizado a lo largo del proyecto. Este componente no existe en los entornos nativos.

Window

El componente Window es imprescindible para el desarrollo de aplicaciones en Titanium. En MobiMarket, cada pestaña tiene su propia ventana. Además para mostrar al usuario las

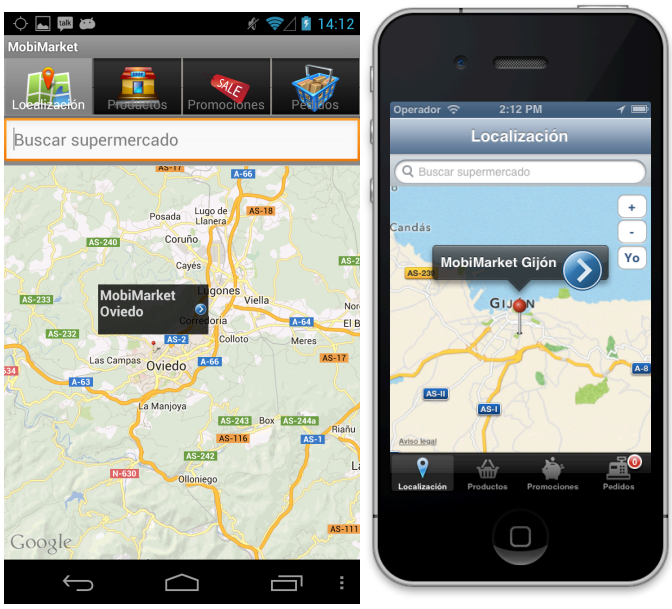
funcionalidades que no están al alcance desde una pestaña determinada (como por ejemplo ver los pedidos anteriores de un usuario) también se utilizan componentes de ventana de forma modal.



Otros componentes

Para el desarrollo del proyecto, tenía mucha importancia el interfaz de mapa. Titanium pone a disposición de su comunidad de desarrolladores una librería Map que funciona muy bien, siendo muy sencillo de implementar y de configurar. Dentro la librería nos encontramos con todos los componentes necesarios para replicar los mapas de los entornos nativos: Maps, MapsButtons, Annotations, Coordenates, etc.

Sólo ha surgido un problema con las anotaciones. El pin de las anotaciones no permite modificar su tamaño en ciertos terminales Android (sí en el emulador) y el menú contextual en iOS tampoco puede modificar su tamaño, lo que queda muy restringido el contenido que se puede insertar en él. En Android tampoco se puede modificar el tamaño de las vistas que se quieran introducir en el menú contextual.



También se ha podido replicar el interfaz de badge de iOS. Es la representación circular de notificaciones de color rojo con un número en su interior. En MobiMarket representa el número de productos diferentes que hay insertados en el carrito de la compra. Su implementación es sencilla y su estética es exactamente igual al nativo. En Android no existe nada similar, así que no hay funcionalidad implementada para la aplicación del sistema operativo de Google.

En la evaluación se ha comprobado que la mayoría de los interfaces que se han tenido que utilizar para el desarrollo de la aplicación MobiMarket pueden reproducirse. Cuando se fuerza a la utilización de varios componentes indexados producen, en varios casos, fallos de ejecución que en entornos nativos funcionan sin problema.

¿Se pueden generar aplicaciones en Titanium exactamente igual a las nativas?
DESCRIPCIÓN

Esta cuestión será afrontada tras concluir el desarrollo de MobiMarket. Una vez concluida la codificación, se podrá describir cuales de los elementos se pueden replicar y cuales no.

EVALUACIÓN

Al principio del desarrollo de MobiMarket no se han tenido muchos problemas para replicar el frontend. En cuanto había que programar interfaces un poco más dinámicos (como una imagen dentro de un ScrollView) la aplicación no respondía correctamente. En cuanto a funcionalidad, con más o menos dedicación si que se ha podido replicar todo. Se ha tenido que buscar mucha información y realizar muchas pruebas y compilaciones antes de conseguir el funcionamiento deseado. Es el caso por

ejemplo, de la codificación de datos o las consultas a la BBDD noSQL CouchDB cuando se precisaba de cursores.

Por lo tanto para replicar una aplicación se depende de la complejidad de la misma. La conclusión es similar a la obtenida en la cuestión anterior. Principalmente el frontend es lo que marca si una aplicación es igual o no a otra, y el frontend está formado por los componentes mencionados en la pregunta anterior. A día de hoy se podrían replicar aplicaciones más o menos sencillas. Mobimarket no se ha podido realizar exactamente igual a la aplicación nativa que ha guiado este proyecto. Sí muy parecida, pero no igual. Si aumentamos el nivel de complejidad de los interfaces en esta aplicación, sería muy difícil ya replicar aplicaciones. Las animaciones y la composición dinámica resultan imposibles en Titanium.

Se concluye, que **no se pueden replicar aplicaciones exactamente iguales** a las nativas. Se pueden hacer con funcionalidades muy parecidas pero con interfaces algo distintos.

¿Se han encontrado fallos en la herramienta Titanium Studio a lo largo del desarrollo de la aplicación?
DESCRIPCIÓN

Esta cuestión se responderá realizando anotaciones del número de fallos de la herramienta y la causa de los mismos a lo largo del desarrollo del proyecto MobiMarket.

EVALUACIÓN

A continuación se mostrarán los fallos encontrados en la herramienta en el desarrollo de la aplicación MobiMarket.

- FALLO 1: el primer fallo detectado, surgió durante una actualización a la hora de compilar el proyecto.
    - Causante: Surgía un error al construir la aplicación cuando se compilaba bajo un sistema operativo OS X 10.7 o superior, que es el que se utilizó para el desarrollo del proyecto.
    - Solución: En la reproducción del fallo en la versión actualizada, ya había sido solventado.
- FALLO 2: otro fallo apareció con la actualización del SDK de Titanium del la versión 3.0.1 a la versión 3.1.0.GA. No se permitía la compilación de aplicaciones en Android.
    - Causante: se detectó que el problema venía de que habían cambiado la ruta de ciertas aplicaciones. Titanium trataba de ejecutar dos aplicaciones (aapt y dx) desde una ruta dentro del SDK de Android dónde no existían.
    - Solución: para solventarlo, se crearon accesos directos a las aplicaciones desde la ruta dónde Titanium trataba de ejecutarlos.
- FALLO 3: No es posible crear un nuevo proyecto utilizando el framework de Alloy desde el menú de Archivo. Únicamente se podía crear desde el dashboard que aparece cada vez que se abre la herramienta. Una vez cerrado el dashboard tampoco hay forma de volver a abrirlo.
    - Causante: no se ha programado dicha funcionalidad.
    - Solución: no tiene solución actualmente. sólo se puede volver a acceder al dashboard para crear un nuevo proyecto Alloy cerrando el entorno de desarrollo y volviendo a abrirlo.
    - Nueva solución: En la última versión del entorno han solventado el fallo y ya se puede crear un proyecto desde el menú archivo.
- FALLO 4: fallo en los ficheros de modelado de datos (.tss).
    - Causante: basándose en la documentación de Titanium, la internalización de la aplicación es llevada a cabo mediante la clase *Locale*. Cuando dicha clase es utilizada,

la herramienta lo toma como un error de codificación y es subrayado en rojo. El siguiente código que sea escrito, es tomado como erróneo y ventajas como el autocompletado son perdidas.
    - Solución: la solución que se ha tomado, es completar los campos de texto cuando el fichero de modelado de datos está completo y funcionando. Se espera que en las próximas versiones el fallo sea subsanado.
- FALLO 5: algún fallo fuerza al emulador de Android a cerrarse repentinamente y sin ningún tipo de mensaje de error.
    - Causante: desconocido por el autor. Se han compilado y probado aplicaciones nativas de Android y funcionan correctamente en el emulador. Investigando entre la comunidad de desarrolladores, parece un fallo aún por resolver. Afecta a sistemas operativos Mac OS X.
    - Solución: aún no existe solución al fallo. La simulación a través de un dispositivo móvil se realiza correctamente y sin ningún tipo de problema.
- FALLO 6: con la última actualización del SDK (a la versión 3.1.1.GA), los proyectos construidos con el SDK de Titanium 3.0.0.GA no pueden ser compilados.
    - Causante: desconocido por el autor.
    - Solución: la solución que se ha encontrado es crear un nuevo proyecto y copiar todos los ficheros fuente en el y volver a compilar el proyecto como un proyecto nuevo.

El entorno de desarrollo **no es ni tan completo ni tan fluido** como el entorno Eclipse para Android o xCode para Objetive-C. Está bastante lejos en este momento pese a ser un entorno también basado en el IDE Eclipse.

3.3.3. EFICIENCIA

Se entiende por eficiencia de Titanium los tiempos de compilación y carga de la aplicaciones desarrolladas en la plataforma. Se comparará la respuesta tanto a nivel software como a nivel hardware. Es decir, se trata de medir si consume más o menos recursos que las aplicaciones nativas para una misma funcionalidad.

También se había planteado recoger una muestra de diez usuarios dejándoles interactuar con la aplicación nativa y con desarrollada en Titanium y que expresaran sus opiniones a través de una encuesta, pero por falta de tiempo no se ha podido llevar a cabo el experimento.

¿Cuánto tiempo tarda en compilar una aplicación en Titanium respecto a las aplicaciones nativas?
DESCRIPCIÓN

Se codificará una aplicación sencilla, como el "Hola Mundo" en Titanium, Android e iOS, que junto con la versión final de MobiMarket y la aplicación nativa facilitada por el CTIC servirán para tomar tiempos de compilación. Los resultados del experimento serán anotados en una tabla.

EVALUACIÓN

Todas las pruebas del experimento se han realizado a través de un MacBook de mediados de 2007 con 3GB de RAM. Los tiempos de compilación se comparan en la siguiente tabla:

	Hola Mundo	MobiMarket
Titanium / Android	48s 317ms	57s 649ms
Android	35s 234ms	42s 111ms
Titanium / iOS	36s 402ms	41s 196ms

iOS	02s 238ms	04s 103ms

Como se puede observar los tiempos de compilación entre Titanium y Android no están muy desequilibrados, pero el XCode sí compila a una velocidad muy superior.

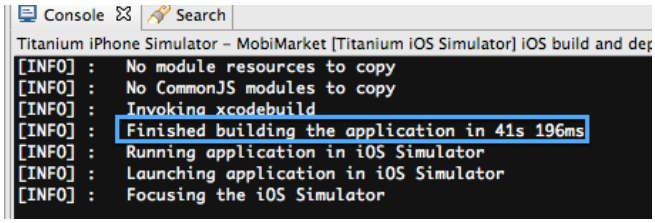


Figura 13

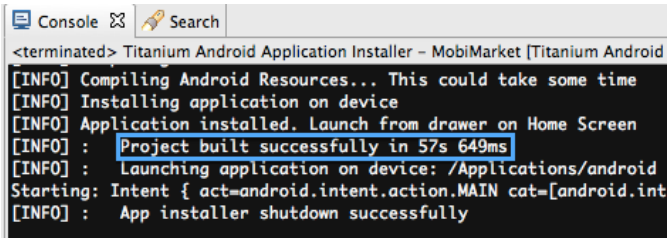


Figura 14

Además a esos tiempos hay que añadirles el tiempo de la preparación que el compilador hace de los ficheros fuente utilizados en el framework Alloy puedan ser leídos por el intérprete de Titanium, antes de enviarlos a los compiladores nativos. Es de aproximadamente medio segundo para el "Hola Mundo" y de dos segundos y medio en el caso de MobiMarket. Este tiempo se eliminaría si en lugar de programar utilizando el patrón arquitectónico que propone Alloy se programa únicamente en JavaScript.

¿Cuánto tiempo tarda en desplegarse una aplicación en Titanium respecto a las aplicaciones nativas?

DESCRIPCIÓN

Se realizará un experimento muy parecido al que se describe en la cuestión anterior. Se recogeran los tiempos de despliegue de aplicaciones tomados, desde el momento de la compilación hasta el momento en el que se lanza la aplicación y podemos interactuar con ella.

EVALUACIÓN

Las pruebas se han realizado con los emuladores y dispositivos arrancados y en reposo, ya que, sobre todo en el caso de Android, el emulador es muy lento y tarda mucho tiempo en estar a pleno funcionamiento.

DESPLIEGUE EN EMULADOR		
	Hola Mundo	MobiMarket
Titanium / Android	15s 543ms	16s 611ms
Android	12s 234ms	12s 666ms
Titanium / iOS	10s 402ms	11s 196ms

iOS	06s 881ms	07s 200ms
DESPLIEGUE EN DISPOSITIVO		
	Hola Mundo	MobiMarket
Titanium / Android	08s 878ms	10s 364ms
Android	08s 114ms	09s 997ms
Titanium / iOS	15s 332ms	15s 645ms
iOS	12s 003ms	12s 005ms

Los tiempos de despliegue de la aplicación **son muy parejos** tanto en Titanium como en los sistemas nativos.

Cabe mencionar, que el emulador en el caso de Android, funciona muy lento. En uso del emulador para aplicaciones también funciona con muy poca fluidez, pero aún así, mucho más fluido que en Titanium.

¿Cuánto tiempo tarda en ejecutarse alguna funcionalidad de una aplicación en Titanium respecto a las aplicaciones nativas?

DESCRIPCIÓN

El experimento se realiza de una forma muy parecida a la desarrollada en las dos cuestiones anteriores pero utilizando para realizar la comparación de tiempos una aplicación llamada Robotium. Sólo se ha podido realizar el experimento en el emulador de Android.

Robotium es una aplicación, en la que se programan los pasos que se quiere realizar de una ejecución. En nuestro caso, se ha establecido la siguiente ejecución de una transacción básica:

- El usuario se loguea.
- El usuario añade un producto de carnicería.
- El usuario añade un producto de panadería.
- El usuario confirma el pedido.
- El usuario hace logout.

El código de las pruebas de Robotium se encuentra incluido en los apéndices del presente trabajo.

EVALUACIÓN

Hay que tener en cuenta que el experimento se ha realizado en el emulador de Android, y son mucho más lentos que los que tendríamos en un terminal.

	Nativa	MobiMarket
RobotiumTest	18s 317ms	19s 559ms

Hay que tener en cuenta, que las dos aplicaciones no son exactamente iguales y para finalizar una compra se necesita un paso más en la aplicación nativa que en MobiMarket. Los tiempos obtenidos son **inferiores** en la aplicación nativa que en la aplicación de MobiMarket.

No se han encontrado herramientas de testeo para la aplicación en iOS.

¿Cuántos recursos de un dispositivo móvil consume una aplicación generada en Titanium respecto a las aplicaciones generadas en Xcode o Android?

Esta prueba a quedado sin realizarse porque se han tenido muchos problemas con la aplicación que se pretendía utilizar. La aplicación para medir el consumo de hardware recibe el nombre de SysTrace y está incluida en el SDK de Android. Para realizar el experimento es necesario un móvil "rooteado". Se ha tratado de realizar el experimento con un Nexus 4, pero la aplicación no era compatible. No se ha podido "rootear" ningún otro terminal, así que la prueba de rendimiento hardware no se ha podido llevar a cabo en este proyecto.

No se han encontrado herramientas de testeo para la aplicación en iOS.

3.3.4. USABILIDAD

Se entiende por usabilidad, la facilidad de utilizar el entorno de desarrollo para crear aplicaciones. Para ello, se comprobará la documentación y recursos que la Appcelerator pone a disposición de su comunidad de desarrolladores, así como lo usable que es, en opinión del autor, la herramienta de desarrollo (el IDE Titanium Studio).

¿Existe una documentación completa a disposición de la comunidad de desarrolladores de Titanium?

DESCRIPCIÓN

Para desarrollar esta cuestión se comentará la experiencia del autor con la documentación para el desarrollo de la aplicación MobiMarket. Se comparará la documentación de Titanium con la documentación que Google pone a disposición de la comunidad de programadores Android y la que Apple proporciona a la comunidad de desarrolladores de iOS.

EVALUACIÓN

Existe una gran cantidad de documentación, muy bien organizada y que se corrige día a día cuando algún desarrollador encuentra fallos a la hora de codificar soluciones en Titanium.

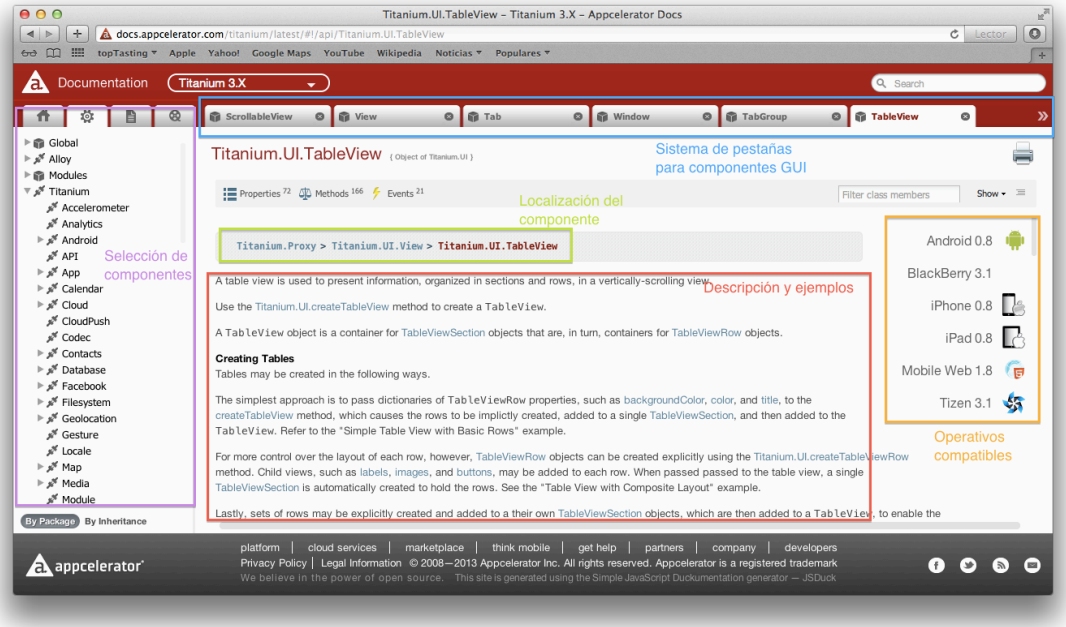


Figura 15

Se posee un sistema de navegación muy intuitivo y bastante completo. Existe un buscador, una ruta del paquete, una lista de componentes, una lista de compatibilidades y la descripción y ejemplos del componente seleccionado.

Los ejemplos son simplistas, y están claramente diseñador para ser utilizados a modo de "copia y pega", indicando a los desarrolladores dónde debería ir el código para ser adaptado a unas necesidades específicas.

Los componentes que se van consultando, se despliegan a través de una serie de pestañas, lo que hace ahorrar bastante tiempo cuando se necesita consultar más de un componente en la documentación.

También se pone a disposición de la comunidad de desarrolladores a través de la documentación un buscador inteligente que funciona con bastante precisión.

La documentación está únicamente en inglés.

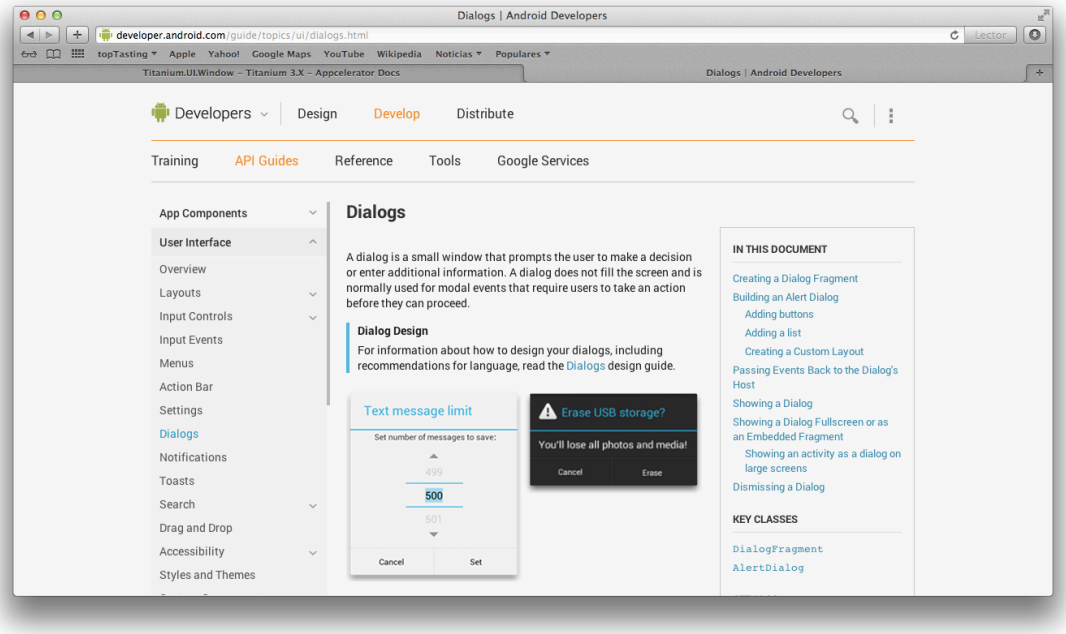


Figura 16

Las apreciaciones del autor, es que, la documentación en Android es más engorrosa de utilizar, aunque en cuanto a contenido es bastante similar.

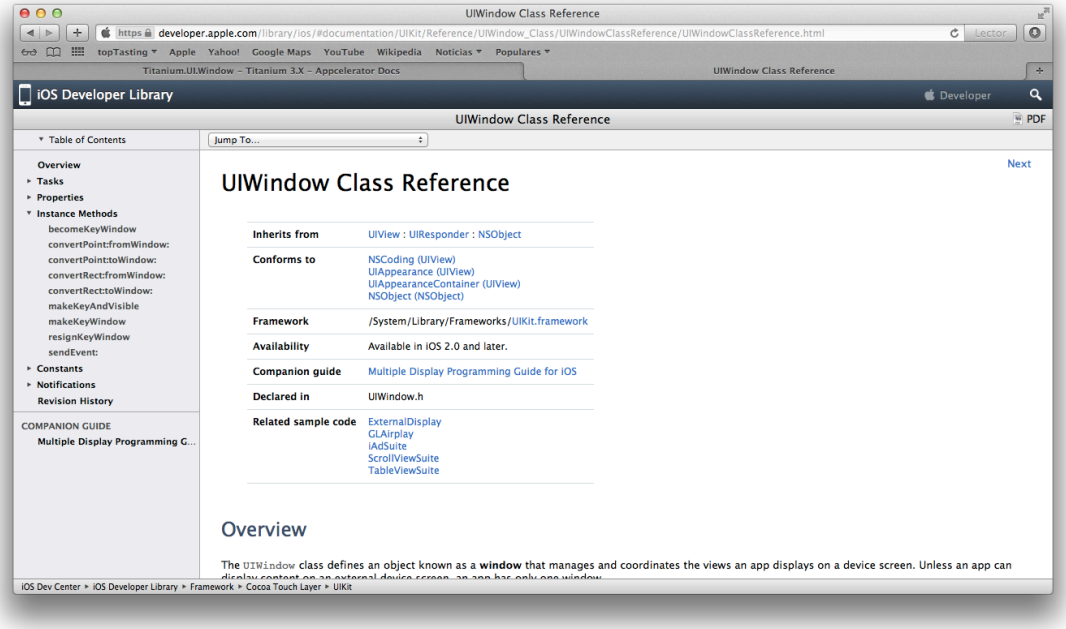


Figura 17

En iOS se encuentra un caso parecido al de Android. Existe una documentación más voluminosa que en Titanium, pero es mas engorroso de utilizar.

Además, existe una comunidad de desarrolladores muy activa. A lo largo del desarrollo del proyecto, se han preguntado varias cuestiones y dos de ellas ha sido respondidas al día siguiente. El número de desarrolladores en Titanium está lejos de los que se tienen en la comunidad Android o en la comunidad iOS, pero sí que es una comunidad activa y creciente. Desde el momento del registro para la descarga de la plataforma y el comienzo del desarrollo, hasta que se ha dado por concluida la codificación de la aplicación la comunidad ha crecido en ciento once mil, doscientos cuatro usuarios.

En los foros de la comunidad, también se han podido encontrar respuesta a la mayoría de las dudas que han surgido a lo largo del desarrollo de la aplicación.

Lo que se concluye, es que la documentación en Titanium **es similar** a la documentación en los sistemas nativos, y aunque no se tenga tanto contenido es, incluso, más usable.

¿Es fácil crear una nueva aplicación en Titanium Studio y tenerla funcionando para test?
DESCRIPCIÓN

Esta es una de las primeras cuestiones a las que se tratará de buscar solución. Se expresará como crear una aplicación nueva partiendo del entorno de desarrollo Titanium Studio correctamente configurado. Se comparará la creación de una aplicación en Titanium, con el proceso que se sigue en los entornos nativos Eclipse y Xcode. Es importante mencionar, que en esta cuestión se evalúa la dificultad de crear un nuevo proyecto, la dificultad en cuanto a nivel de aprendizaje del lenguaje se le dará respuesta en una de las cuestiones de funcionalidad.

EVALUACIÓN

Con la instalación y configuración correcta del entorno, es muy sencillo tener una nueva aplicación corriendo en Titanium Studio. La operación es realizada de una forma idéntica al entorno Eclipse de

una aplicación en Android. Hay que tener en cuenta que el entorno Titanium Studio está basado en el IDE Eclipse.

Se puede lanzar un proyecto en un dispositivo de testeo que se tenga acoplado al computador, o bien en los emuladores que vienen contenidos en los SDK de Android e iOS. Además a la hora de crear un nuevo proyecto, también se permite seleccionar una serie de sencillas plantillas, con la apariencia principal que se desea para la aplicación. Para la creación de la aplicación MobiMarket con la que se ha evaluado la herramienta, hemos seleccionado una plantilla de Alloy para una aplicación con pestañas. Sin llegar a escribir una línea de código hemos conseguido lo que se pretendía.

El proceso de compilado y ejecución también es idéntico al de la herramienta nativa de Android.

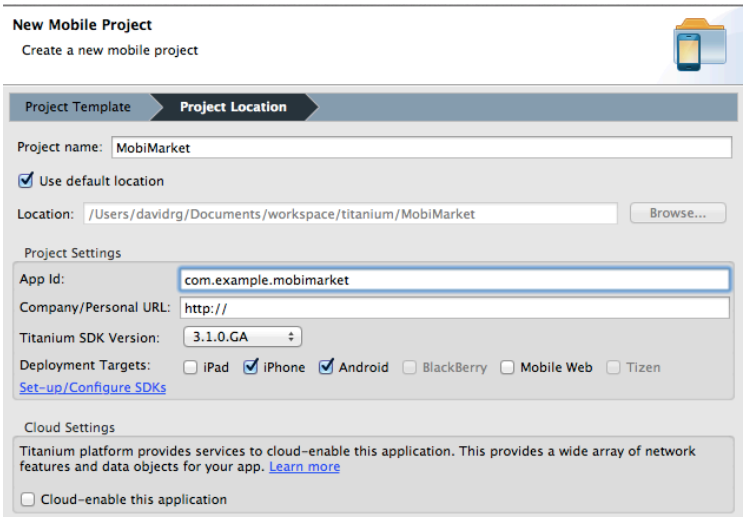


Figura 18

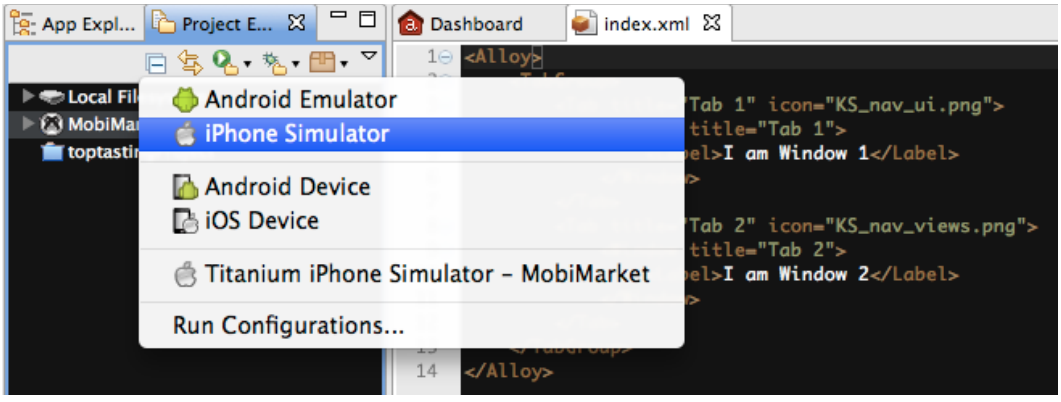


Figura 19

En la figura 18 y 19 se puede observar el proceso de creación y de ejecución de un nuevo proyecto en Titanium. Es absolutamente similar a Android, ya que ambos están basando en el IDE Eclipse.

Figura 20

En la figura 20 se representa el simulador de iOS ejecutándose tras haber realizado la primera compilación de la aplicación MobiMarket.

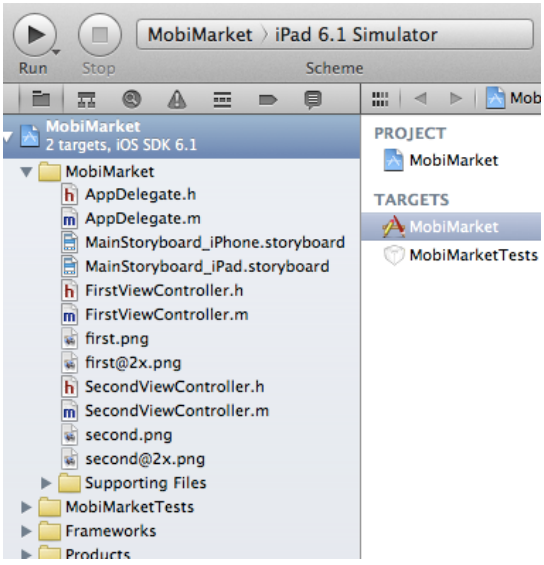


Figura 21

En la figura 21 se representa como se ejecutaría una aplicación en el entorno XCode, que también es similar a Titanium Studio y a Eclipse para Android.

Por lo tanto podemos concluir que el proceso de creación y ejecución de una nueva aplicación en Titanium es **similar al nativo**.

3.3.5. MANTENIBILIDAD

Se entiende por mantenibilidad el proceso de actualización y de corrección de errores de la plataforma. Se enumerarán el número de actualizaciones recibidas durante el desarrollo del proyecto, así como, las apreciaciones de mejora tras la actualización.

¿Es necesaria algún tipo de licencia para desarrollar aplicaciones en Titanium? ¿Qué diferencias son encontradas respecto a las licencias nativas?

DESCRIPCIÓN

Se evaluará todo los datos referidos a la licencia para programar en Titanium si existiera alguna y se comparará con las licencias necesarias para desarrollar aplicaciones en iOS y en Android.

EVALUACIÓN

En Titanium, al igual que en Android, no es necesaria licencia de desarrollo alguna. Pero sí es obligatorio registrarse como desarrollador para acceder al SDK y al entorno de desarrollo. Es totalmente gratuito.

La documentación que se pone a disposición de la comunidad de desarrolladores, es accesible a toda persona que quiera acceder a ella.

Las únicas licencias necesarias serán para el despliegue de aplicaciones en dispositivos iOS. Dichas licencias son idénticas a las necesarias en xCode. De hecho, no es necesario introducir los certificados en el entorno de desarrollo Titanium Studio, ya que este accede directamente al repositorio de XCode para localizarlas. El desarrollador simplemente deberá indicar que certificado desea utilizar.

Por lo tanto el uso de licencias **es igual** en nativo que en Titanium.

¿Cuántas actualizaciones se han recibido durante el desarrollo de la aplicación?

DESCRIPCIÓN

Se anotará el número de actualizaciones recibidas para la plataforma Titanium desde el comienzo del desarrollo de la aplicación MobiMarket hasta el final del mismo. También se anotarán las actualizaciones recibidas del SDK de Android e iOS así como sus entornos de desarrollo.

EVALUACIÓN

Se ha comenzado a desarrollar el proyecto MobiMarket el mismo día que se ha empezado el proyecto Appcelerator sacaba la actualización 3.0.0.GA. En la nueva actualización se han encontrado varias diferencias con las versiones anteriores con las que, el autor del proyecto, había dado sus primeros pasos en Titanium para familiarizarse con el lenguaje. El proyecto se terminó con la versión 3.1.1.GA tras tres actualizaciones.

El proyecto se desarrolló, en un alto porcentaje, durante dos meses en los que se recibieron tres actualizaciones de Titanium y una de Alloy.

En el mismo periodo se recibieron dos actualizaciones de Android y otras dos de XCode. Con lo cuál, se puede concluir que las actualizaciones recibidas son, en número, **similares** a las que han recibido los sistemas nativos.

¿Las actualizaciones mejoran realmente el entorno de desarrollo?

DESCRIPCIÓN

Se anotarán las funcionalidades que han mejorado tras cada actualización si estas son palpables. De descarta anotar la descripción que viene con cada actualización con las reparaciones y las mejoras, sino las apreciaciones más reseñables por parte del autor.

EVALUACIÓN

Las actualizaciones han mejorado el funcionamiento del sistema, aunque también han dado algún problema no esperado.

Ejemplos de mejora han sido la corrección de algún bug de componentes gráficos para el desarrollo de interfaces, como el *SearchBar* que forzaba el cierre en el operativo Android cuando dicho componente se situaba dentro de una tabla. Otra gran mejora, en cuanto a usabilidad, ha sido introducir el sistema de autocompletado. Tener que mirar cómo se llamaba cada componente hacía perder demasiado tiempo a un desarrollador. También se ha introducido la posibilidad de crear un nuevo proyecto Alloy desde el menú archivo en lugar de tener que hacerlo únicamente cuando el entorno de desarrollo es abierto. Se ha introducido la internacionalización en Alloy, que en un principio no era funcional. Existen un gran número de mejoras palpables en cada actualización.

Algunas actualizaciones también han sido un tanto problemáticas. La más destacable es que, dependiendo del operativo en el que la herramienta de desarrollo es instalada, el compilador busca aplicaciones necesarias en una ruta que no existe. También han surgido problemas de compilación de aplicaciones desarrolladas en versiones anteriores.

Las actualizaciones de Android y de XCode no se han probado (queda fuera del alcance de este trabajo), pero se ha buscado opinión de desarrolladores en los foros de la comunidad y no se han encontrado demasiados halagos ni demasiadas quejas. Es obvio, que la plataforma de desarrollo Titanium está mucho más verde y las actualizaciones se hacen notar mucho más.

Se concluye que las actualizaciones se hacen notar, y mejoran varios problemas en cada una de ellas.

3.3.6. PORTABILIDAD

Se entiende por portabilidad la facilidad de instalación de la herramienta en diferentes sistemas operativos y el funcionamiento en cada uno de ellos.

¿Es fácil instalar el entorno de desarrollo Titanium Studio?

DESCRIPCIÓN

Es una de las primeras cuestiones a solventar. Se especificarán las características propias de la instalación de Titanium Studio respecto a la instalación del entorno Eclipse para Android y de XCode para iOS. Es el primer paso obligatorio para poder desarrollar aplicaciones en Titanium.

El sistema de Titanium Studio ha evolucionado mucho desde sus inicios hasta ahora. En un primer momento todo se debía realizar desde la línea de comandos, y eran necesarias varias configuraciones previas antes de arrancar un nuevo proyecto. Desde el 2011 se ha desarrollado un IDE a raíz de Eclipse que permite la generación proyectos de una forma muy similar a Android (también utiliza el IDE Eclipse para la elaboración de proyectos) y XCode como se ha podido comprobar en una de las cuestiones anteriores. En esta cuestión se instalará y evaluará el entorno gráfico, el IDE de desarrollo de Titanium Studio. Se descarta utilizar la línea de comandos.

EVALUACIÓN

Para la configuración de Titanium Studio es necesaria la descarga y configuración del SDK de Android e iOS (incluido en la plataforma XCode), Java y Python para que se pueda iniciar un proyecto. Además se exige mover librerías de una carpeta de instalación a otra dependiendo del sistema operativo que se vaya a utilizar.

Dista un poco del proceso de despliegue de aplicaciones en los entornos nativos, que tras descargar el SDK correspondiente, se puede iniciar sin problemas un nuevo proyecto.

Para configurar el entorno Android, no sirve sólo con instalar las librerías de la última versión del SDK, sino que también es necesario instalar el SDK de la versión de Android 2.3.3 (API 10). Y no sólo las librerías necesarias del SDK propiamente dicho sino también las librerías que hacen referencias a las Google APIs y a la imagen que representa la arquitectura dónde el emulador simulará el despliegue de la aplicación.

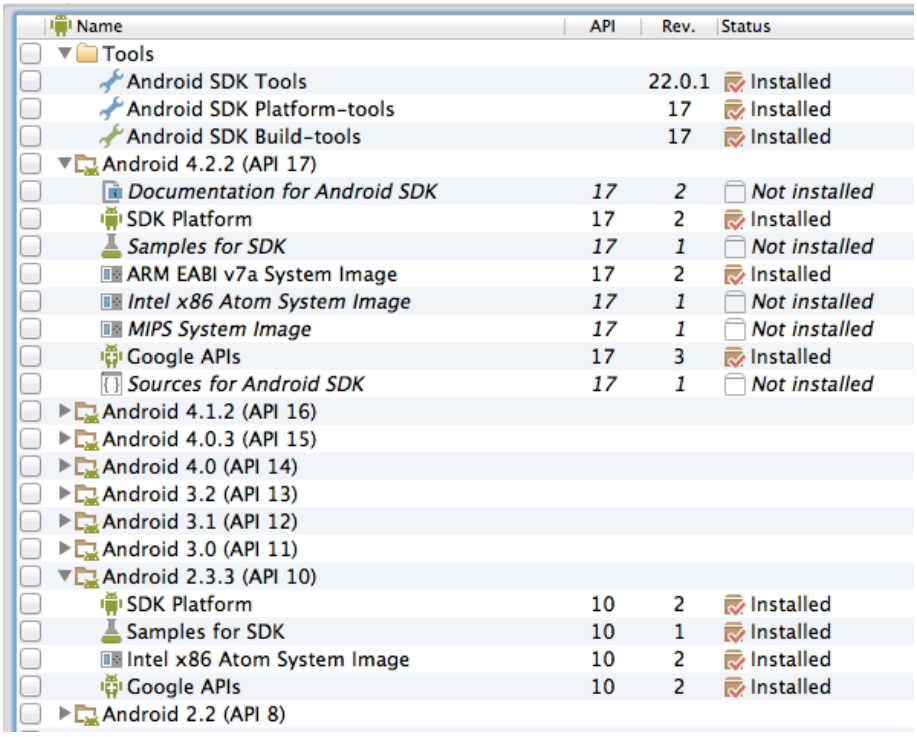


Figura 22

En un principio, cuando se comenzó a desarrollar la aplicación, eran necesarias las librerías de Android 2.2 (API 8). Pero con la actualización a Titanium 3.1.0.GA se comenzó a utilizar los paquetes del SDK 2.3.3.

La configuración para el desarrollo en iOS es realizada de forma automática por el entorno de desarrollo Titanium Studio, una vez descargado y configurador el XCode 6.1.

Se concluye que el proceso de instalación del entorno de desarrollo **es similar** al de los entornos nativos. Quizás es un poco más complicado, puesto que tiene que ser instalado el entorno Android, el entorno iOS y el entorno Titanium. Pero todos ellos se instalan y configuran de forma similar.

¿Puede instalarse el entorno de desarrollo de Titanium Studio en diferentes sistemas operativos?
DESCRIPCIÓN

Se buscará información acerca de las posibilidades de instalación de Titanium y se instalará en los principales sistemas operativos que lo soporten. De codificará y ejecutará una aplicación sencilla para comprobar el correcto funcionamiento.

EVALUACIÓN

Al igual que el entorno Android, puede ser instalado en sistemas operativos Windows, Mac OS X y Unix. Xcode únicamente puede ser instalado en entornos Mac OS X.

Para codificar soluciones para dispositivos iOS es necesario la instalación de XCode también para trabajar con Titanium, luego es necesario instalar el entorno en un sistema operativo Mac OS X.

El desarrollo de MobiMarket implicaba la comparativa con una aplicación nativa desarrollada en iOS y en Android, por lo tanto es necesario la instalación de XCode. Por ello, el sistema operativo elegido para la realización del proyecto ha sido Mac OS X versión 10.7.5 (Lion). El computador utilizado ha sido un MacBook de medidados de 2007.

Finalmente no se ha instalado el entorno en Ubuntu o Debian (las especificaciones del producto afirma que funciona) pero sí en un entorno Windows y su funcionamiento para desplegar aplicaciones para Android es correcto. Para iOS, por lo mencionado anteriormente no se ha podido probar en el sistema operativo Windows.

Luego el entorno de desarrollo de Titanium **es similar** en cuanto a portabilidad al entorno Android y más portable que XCode (sólo es portable por sistemas Mac OS X en su versión Lion o Mountain Lion).

¿Se pueden utilizar librerías externas? ¿Es difícil importar librerías externas?
Para la realización del proyecto MobiMarket no ha sido necesario el uso de librerías externas y no se ha dispuesto de tiempo para realizar experimentos a parte.

3.4. RESULTADOS Y CONCLUSIONES

En esta sección se creará una tabla resumen con todos los resultados obtenidos a lo largo de la evaluación de la herramienta con las valoraciones cualitativas mencionadas en la introducción. Tras la tabla de resultados se expondrán las conclusiones obtenidas tras la finalización del proyecto de evaluación.

RESULTADOS DE EVALUACIÓN DE TITANIUM	
1. Funcionalidad	
CUESTIÓN	**VALORACIÓN**
¿Es difícil tener un nuevo proyecto de Titanium funcionando en un dispositivo móvil?	Similar al nativo
¿Cómo es el proceso de compilación y depuración respecto a los entornos de desarrollo XCode y Android?	Similar al nativo
¿Qué grado de dificultad conlleva programar una aplicación en Titanium?	Peor al nativo
2. Fiabilidad	
CUESTIÓN	**VALORACIÓN**
¿Cuántos interfaces se pueden utilizar en la creación de una aplicación en Titanium respecto a los entornos nativos?	Peor al nativo
¿Se pueden generar aplicaciones en Titanium exactamente igual a las nativas?	Peor al nativo
¿Se han encontrado fallos en la herramienta Titanium Studio a lo largo del desarrollo de la aplicación?	Peor al nativo
3. Eficiencia	
CUESTIÓN	**VALORACIÓN**
¿Cuánto tiempo tarda en compilar una aplicación en Titanium respecto a las aplicaciones nativas?	Peor al nativo
¿Cuánto tiempo tarda en desplegarse una aplicación en Titanium respecto a las aplicaciones nativas?	Similar al nativo
¿Cuánto tiempo tarda en ejecutarse alguna funcionalidad de una aplicación en Titanium respecto a las aplicaciones nativas?	Peor al nativo
¿Cuántos recursos de un dispositivo móvil consume una aplicación generada en Titanium respecto a las aplicaciones generadas en Xcode o Android?	No evaluado
4. Usabilidad	
CUESTIÓN	**VALORACIÓN**
¿Es fácil crear una nueva aplicación en Titanium Studio?	Similar al nativo
¿Existe una documentación completa a disposición de la comunidad de	Similar al nativo

desarrolladores de Titanium?	
5. Mantenibilidad	
CUESTIÓN	**VALORACIÓN**
¿Es necesaria algún tipo de licencia para desarrollar aplicaciones en Titanium?	Similar al nativo
¿Qué diferencias existen entre las licencias nativas y la de Titamium?	Similar al nativo
¿Cuántas actualizaciones de Titamium se han recibido durante el desarrollo de la aplicación?	Similar al nativo
¿Las actualizaciones mejoran realmente la plataforma de desarrollo?	Similar al nativo
6. Portabilidad	
CUESTIÓN	**VALORACIÓN**
Es fácil instalar el entorno de desarrollo Titanium Studio?	Similar al nativo
¿Puede instalarse el entorno de desarrollo en diferentes sistemas operativos?	Similar al nativo
¿Se pueden utilizar en Titanium librerías externas?	No evaluado
¿Cómo es el proceso de importación de librerías?	No evaluado

Tras la evaluación de la herramienta se han obtenido las siguientes conclusiones:

- En cuanto a funcionalidad, debe de crearse una GUI para el desarrollo de interfaces si se quiere llegar a competir con los IDEs de desarrollo nativo. En cuanto a las posibilidades de creación de proyectos, compilación y depuración está a un nivel más que aceptable respecto a las plataformas nativas.
- En cuanto a la fiabilidad está lejos de los entornos nativos. Posee menos interfaces, menos librerías y tiene ciertos problemas derivados de JavaScript (lenguaje de codificación de aplicaciones para Titanium) en cuanto a la correcta compilación de los ficheros interpretados por Titanium en los compiladores nativos. Además surgen problemas a la hora de indexar más de un componente gráfico forzando malas presentaciones o forzando el cierre de la aplicación.
- En cuanto a eficiencia también está por debajo Titanium respecto a los sistemas nativos. Aunque las diferencias tampoco son excesivas, y con el incremento de potencia del hardware de los terminales móviles pueden hacerse despreciables en un futuro próximo.
- La usabilidad de los elementos que Appcelerator pone a disposición de su comunidad de desarrolladores es excelente. En opinión del autor, en lo referido a la documentación incluso por encima de los entornos nativos.
- La mantenibilidad también se ha comprobado que es similar a la de los entornos nativos. Se percibe que la plataforma Titanium está mucho menos madura que Android o XCode, pero sea notado esfuerzo en reparar errores con constantes actualizaciones.
- En cuanto a la portabilidad, se puede afirmar que el entorno corre correctamente sobre sistemas operativos Mac OS X y Windows y las especificaciones también afirman que funciona sobre operativos Ubuntu y Debian. Hay que tener en cuenta la limitación establecida por Apple de desarrollar aplicaciones para iOS únicamente en operativos Mac OS X. No se han podido comprobar el uso de librerías externas, porque no han sido necesarias para el

desarrollo de la aplicación MobiMarket y no se ha dispuesto tiempo para probarlo fuera del proyecto.

En opinión del autor, el entorno Titanium a día de hoy se puede utilizar para el desarrollo simple de aplicaciones que las empresas puedan utilizar para testear si una nueva idea puede funcionar en el mercado sin la necesidad de invertir grandes cantidades de dinero en desarrollos paralelos para múltiples plataformas. Una vez validada la idea, es aconsejable desarrollar la aplicación en sus respectivos sistemas nativos por las limitaciones encontradas.

Aún así el proyecto Appcelerator Titanium aún no está maduro, y si la curva de mejora sigue creciendo como hasta ahora, es probable que en un futuro pueda competir con Android e iOS en cuanto al desarrollo de aplicaciones para móviles.

CAPÍTULO 4. LA APLICACIÓN MOBIMARKET

4.1. INTRODUCCIÓN

En esta sección se describirá el contexto y el alcance de la aplicación MobiMarket. El objetivo principal del proyecto es la evaluación de la herramienta Titanium, que será dónde será codificada la aplicación.

El presente documento no se extenderá en cuanto a análisis y diseño de la aplicación. Se documentará lo justo y necesario para el desarrollo de la aplicación.

El formato de este capítulo se regirá por una adaptación de la norma IEEE 830 de 1998. Aún existiendo muchos otros estándares más modernos, claros y completos, se ha decidido utilizar este, porque es mucho más simple y cumple mejor con el propósito. Este documento no pretende ser una especificación completa y detallada, sino un apoyo para el autor del proyecto a desarrollar una aplicación cuyo fin es la evaluación de la herramienta de desarrollo no la aplicación en sí.

4.1.1. ÁMBITO DEL SISTEMA

El propósito de este documento será la descripción y definición de los requisitos de software de forma que sean fácilmente consultables por personal y desarrolladores interesados en conocer el funcionamiento de la aplicación. También servirá para que el personal del CTIC pueda comprobar si todos los requisitos aquí registrados coinciden con los de la aplicación nativa, con la que se comprará el rendimiento una vez el proyecto haya sido concluido.

La aplicación a desarrollar para la evaluación de la herramienta recibirá el nombre de **Proyecto MobiMarket**.

Permitirá a un determinado supermercado, mostrar todos los centros que tenga repartidos por el mundo a sus clientes, a través de terminales móviles, pudiendo acceder a la información principal de dichos centros. Entendemos por información principal datos como el nombre del centro, la localización, el teléfono, etc.

El sistema localizará la posición del usuario a través de un sistema de GPS y mostrará todos los supermercados que tiene a su alrededor. También permitirá buscar un supermercado concreto e incluso mostrar todos los centros existentes. Toda la información relevante de los supermercados serán mostradas al usuario. El nombre, la dirección, el teléfono, el horario así como algún tipo de fotografía entre otras cosas.

Los usuarios también podrán consultar las categorías de todos los productos que el supermercado posee (como por ejemplo “Frutería”) y obviamente, los productos que se venden en dichos supermercados. Cada producto mostrará también información básica como su nombre, su precio o una descripción de lo que el usuario podría comprar.

Los supermercados tendrán una serie de promociones que irán variando en el tiempo y que es interesante que sus clientes (los usuarios del sistema) también conozcan a través de la aplicación.

Los productos y las promociones de cada supermercado podrán ser comprados por los usuarios a través del terminal móvil. Para ello, los usuarios estarán deberán facilitar la información necesaria para el envío de pedidos a domicilio.

Con este sistema se pretende facilitar la compra a los clientes, desde cualquier lugar dónde se encuentren. Sin necesidad de desplazarse al supermercado. Sin necesidad de estar en casa y tener que realizar la compra a través del ordenador. Es decir, los pedidos podrán ser enviados a través del sistema desde cualquier lugar dónde el dispositivo móvil tenga acceso a la red de datos. De esta forma,

el supermercado espera ampliar el número de ventas a través de un mayor número de compras de sus clientes actuales y ampliar su número de clientes de forma viral.

También es importante mencionar que el sistema de información no sólo va dirigido al supermercado cliente, sino que también será utilizado por el CTIC para comprobar el rendimiento de la aplicación en comparación al funcionamiento de un sistema similar desarrollado en entornos de programación móvil nativos. El proyecto será desarrollado utilizando la plataforma Appcelerator Titanium, que interpretará un lenguaje común (JavaScript principalmente) para que diferentes compiladores (gcc y javac) puedan comprender los ficheros de código fuente y crear los instaladores necesarios para que la aplicación pueda ser desplegada en dispositivos móviles iPhone y Android.

A lo largo de este proyecto no se mencionará en ningún momento ni el nombre ni ningún tipo de funcionalidad de la aplicación nativa facilitada con la que se comprará el rendimiento, por temas de confidencialidad del CTIC.

4.1.2. ALCANCE

La aplicación llegará a los usuarios a través de las diferentes medios de distribución que la comunidad de desarrolladores para dispositivos móviles tienen a su disposición. Concretamente GooglePlay e iTunes Store, ya que el alcance de este proyecto se limita a realizar la aplicación para terminales iPhone y Android.

La aplicación no es completamente independiente, pertenece a una plataforma mayor. La información está centralizada y es también accesible a través de una aplicación web a mediante la cuál, el personal del supermercado encargado de empaquetar los pedidos podrá acceder. Los administradores de la aplicación web de MobiMarket tendrán la posibilidad de editar todo el contenido de la base de datos. Tanto la aplicación web como la BBDD dónde se almacena la información ya son productos terminados y probados que también quedan fuera del alcance de este proyecto.

El objetivo será ligar correctamente la información de la BBDD con la nueva aplicación, y la aplicación web ya en producción, tratando la información de forma bidireccional. Las consultas a mencionada base de datos se realizan a través de servicios web, de forma que la información sea accesible utilizando diferentes aplicaciones y entornos.

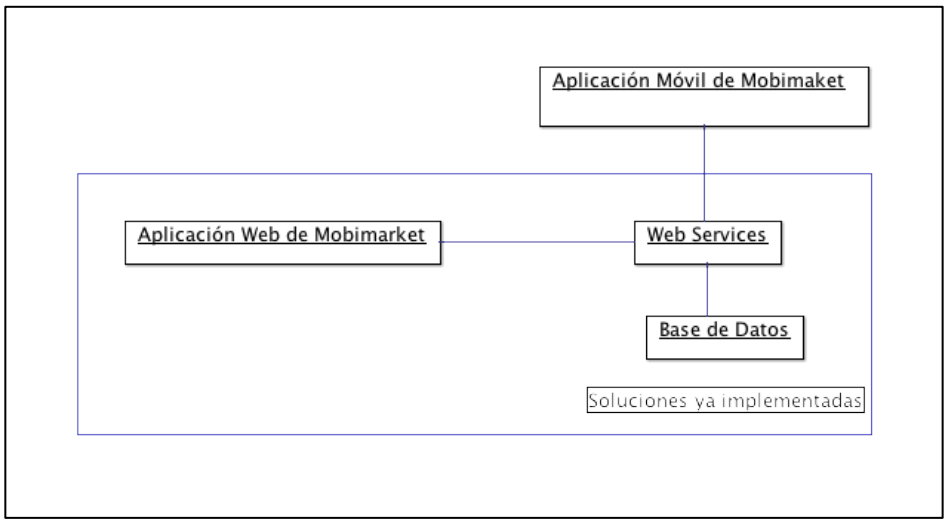


Figura 23

La gestión del supermercado, será tratada desde la aplicación web desarrollada para dicho fin. La aplicación móvil a desarrollar a lo largo de este proyecto, tendrá como únicos destinatarios los clientes del supermercado y servirá para realizar consultas de supermercados y de los productos que pone a disposición para su público.

Por lo tanto, la aplicación móvil sólo llega a un único actor que es el cliente del supermercado. Los repartidores y el personal del almacén, quedan fuera del alcance de este proyecto.

No es menos importante mencionar, que el sistema se desarrollará en el entorno de desarrollo Titanium Studio, con objeto de aprendizaje del lenguaje, evaluación de dicha herramienta y comparativa entre el rendimiento del producto final con un producto similar ya desarrollado en sus respectivos entornos nativos.

PARTES INTERESADAS

Las partes interesadas a las que atañe la aplicación serán los clientes del supermercado, el personal de almacén y los repartidores de pedidos.

- Cliente del supermercado: es la parte interesada más importante para la presente solución. Son los actores que interactuarán directamente con la aplicación móvil a desarrollar. Realizarán los pedidos a través de ella.
- Personal del almacén: son los encargados de recibir los pedidos realizados por los clientes a través de la aplicación web, preparar el pedido y entregárselo al repartidor con el domicilio al que debe realizarse la entrega.
- Repartidor: el repartidor entregará el pedido al domicilio pertinente y volverá al almacén a por más entregas.

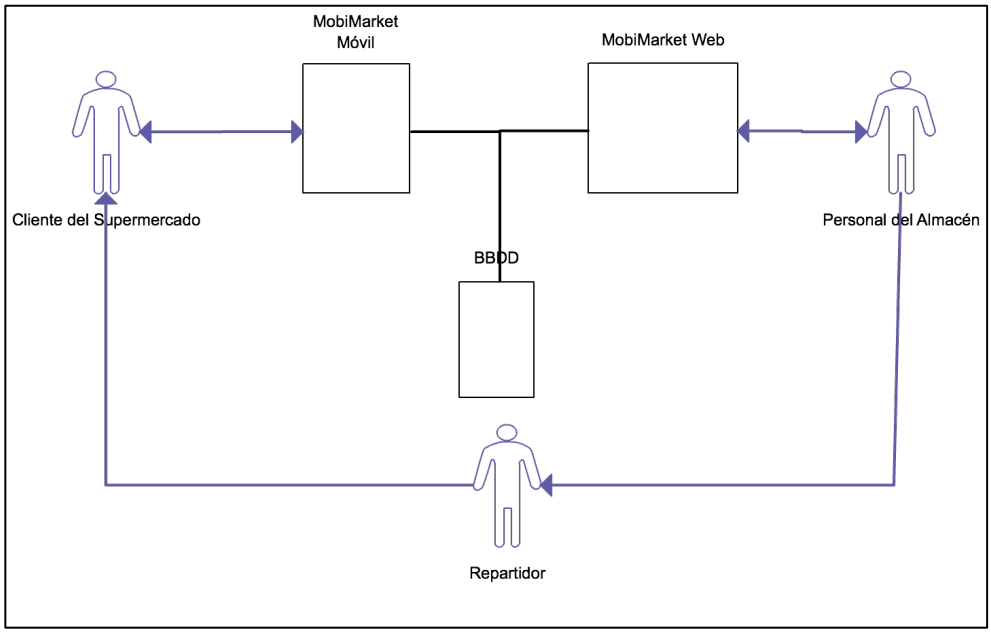


Figura 24

Los **clientes del supermercado** consultarán información de los supermercados y realizarán pedidos a través de la aplicación móvil del MobiMarket. Estos pedidos llegarán a la BBDD que los enviará a la aplicación web. La aplicación web será consultada por el **personal del almacén** que preparará el pedido y se lo entregará al **repartidor**, que a su vez lo llevará al domicilio del cliente del

supermercado. El repartidor deberá informar de que el pedido ha sido entregado y el personal del almacén será el encargado de cambiar el estado de los pedidos a través de la aplicación web.

No se requieren grandes conocimientos de ningún tipo de materia ni especial destreza para el manejo de la aplicación. Cualquier persona mínimamente familiarizada con el uso de terminales móviles no debería de tener ningún problema en utilizar la aplicación.

No obstante, se adjuntan una serie de manuales de usuario en la presente memoria que pueden ser consultados en el apartado *2.6. Manuales de Usuario*.

4.2. ESPECIFICACIÓN DE REQUISITOS

Esta sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores diseñar un sistema que satisfaga los requisitos que aquí se engloban, y que permita al equipo de pruebas planificar y realizar las pruebas que demuestren que el sistema satisface dichos requisitos. Todo requisito aquí especificado describirá comportamientos externos del sistema, perceptibles por parte de los usuarios, operadores y otros sistemas.

4.2.1. REQUISITOS FUNCIONALES

En esta sección se describirán todas aquellas acciones que deberá llevar a cabo el software. Para representarlos, utilizaremos diagramas de casos de uso y una tabla con los requisitos funcionales.

Al final se añadirá una tabla dónde se indique los requisitos funcionales abarcados por cada caso de uso, de forma que sea fácilmente comprobable que todos ellos quedan satisfechos.

Agruparemos los requisitos funcionales por la funcionalidad de cada una de las partes de la aplicación: "Supermercados", "Categorías y Productos", "Promociones" y "Carrito de la Compra".

Supermercados

- RF 1: Los usuarios podrán consultar información de los supermercados.
  - RF 1.1: De un supermercado interesa saber su nombre, id, dirección, teléfono, horario laboral y coordenadas (en longitud y latitud).
  - RF 1.2: La sección de localización contendrá un sistema de búsqueda de supermercados.
    - RF 1.2.1: Los supermercados se buscarán por su nombre.
    - RF 1.2.2: No será necesario introducir el nombre completo del supermercado para que este sea localizado. Ej.: El texto "Oviedo", debe localizar a todos los supermercados que contengan la cadena "Oviedo" entre su nombre.
  - RF 1.3: La sección localización contendrá un sistema de representación en forma de mapa con la ubicación de los supermercados en sus coordenadas.
    - RF 1.3.1: Cuando la aplicación sea abierta, se mostrarán en el mapa todos los supermercados conocidos.
    - RF 1.3.2: Se podrá añadir en el mapa un supermercado en concreto.
    - RF 1.3.3: Se podrá añadir en el mapa todos los supermercados encontrados.
    - RF 1.3.4: Si no se introduce ningún carácter para buscar supermercados se localizarán todos ellos.

Categorías y productos

- RF 2: Los usuarios podrán consultar un producto determinado a través de la categoría a la que pertenezca.
  - RF 2.1: De cada categoría interesa registrar su imagen asociada, id y nombre.
  - RF 2.2: De cada producto interesa registrar su imagen asociada, id, descripción, nombre, precio y tipo.
    - RF 2.2.1: Un producto podrá ser añadido al carrito de la compra.

Promociones

- RF 3: Los usuarios podrán consultar las promociones disponibles en el supermercado.
  - RF 3.1: De una promoción interesa registrar su id, imagen asociada, nombre, descripción, fecha de inicio, fecha de fin y precio.
  - RF 3.2: Las promociones serán mostradas ordenadas por fecha.

- RF 3.3: Sólo se mostrarán las promociones que se encuentran activas en el momento de la consulta. (Es decir, que la fecha de la consulta está comprendida entre la fecha de inicio y fin de la misma).
- RF 3.4: Las promociones podrán ser añadidas al carrito de la compra.

Pedidos

- RF 4: La aplicación contendrá una sección para el pedido del usuario.
  - RF 4.1: De cada pedido interesa saber los productos y promociones añadidos al mismo con la información a la que se hace referencia en RF 3.1 y RF 2.2.
  - RF 4.2: Un producto o promoción podrá ser eliminado del carrito.
  - RF 4.3: Se podrá modificar la cantidad de un producto o promoción determinada del pedido.
  - RF 4.4: Podrán eliminarse todos los productos del carrito al mismo tiempo.
    - RF 4.4.1: Deberá de confirmarse la acción antes de eliminar todos los productos del pedido.
  - RF 4.5: El pedido será enviado y almacenado en la base de datos.
    - RF 4.5.1: Antes de enviar el pedido debe de confirmarse el mismo.
  - RF 4.6: El sistema podrá consultar y mostrar los pedidos que ha realizado con anterioridad el usuario.
    - RF 4.6.1: De los pedidos anteriores interesa saber la fecha y hora en la que fue realizado, el estado actual (en proceso o procesado) y el precio total del pedido.

Usuarios

- RF 5: Un usuario deberá autenticarse en el sistema para acceder a la aplicación principal (RF 1.x.x hasta RF 4.x.x.).
  - RF 5.1: Un usuario utilizará un nombre de usuario y una contraseña para autenticarse.
  - RF 5.2: El usuario y la contraseña deben estar registrados en la base de datos.
- RF 6: Un usuario deberá registrarse en la aplicación antes de poder autenticarse.
  - RF 6.1: Los datos que deberá insertar el usuario son su nombre, apellidos, dirección, email, código postal, fecha de nacimiento, sexo, teléfono, nombre de usuario, contraseña y confirmar contraseña.
    - RF 6.1.1: Todos los datos son obligatorios.
    - RF 6.1.2: El email deberá de incluir una única arroba y un único punto en la parte derecha de la arroba que no puede ser ni al final ni al principio.
    - RF 6.1.3: El número de teléfono debe estar compuesto por nueve números.
    - RF 6.1.4: El código postal de debe estar compuesto por cinco números.
    - RF 6.1.5: El sexo debe ser “Male” o “Female”
    - RF 6.1.6: La fecha de nacimiento sigue un estándar y debe de tener el formato 2013-02-23T12:00:00.000Z. Primero el año en cuatro dígitos, luego el mes en dos dígitos y luego el día en dos dígitos también. El carácter ‘T’ y luego la hora, minutos, segundos y milisegundos. Termina con el carácter ‘Z’. Para más información consultar la norma ISO 8601.
    - RF 6.1.7: El nombre de usuario debe de ser único. No debe estar registrado en la BBDD.
    - RF 6.1.8: Los campos contraseña y confirmar contraseña deben de ser iguales.

4.2.2. ESPECIFICACIÓN DE CASOS DE USO

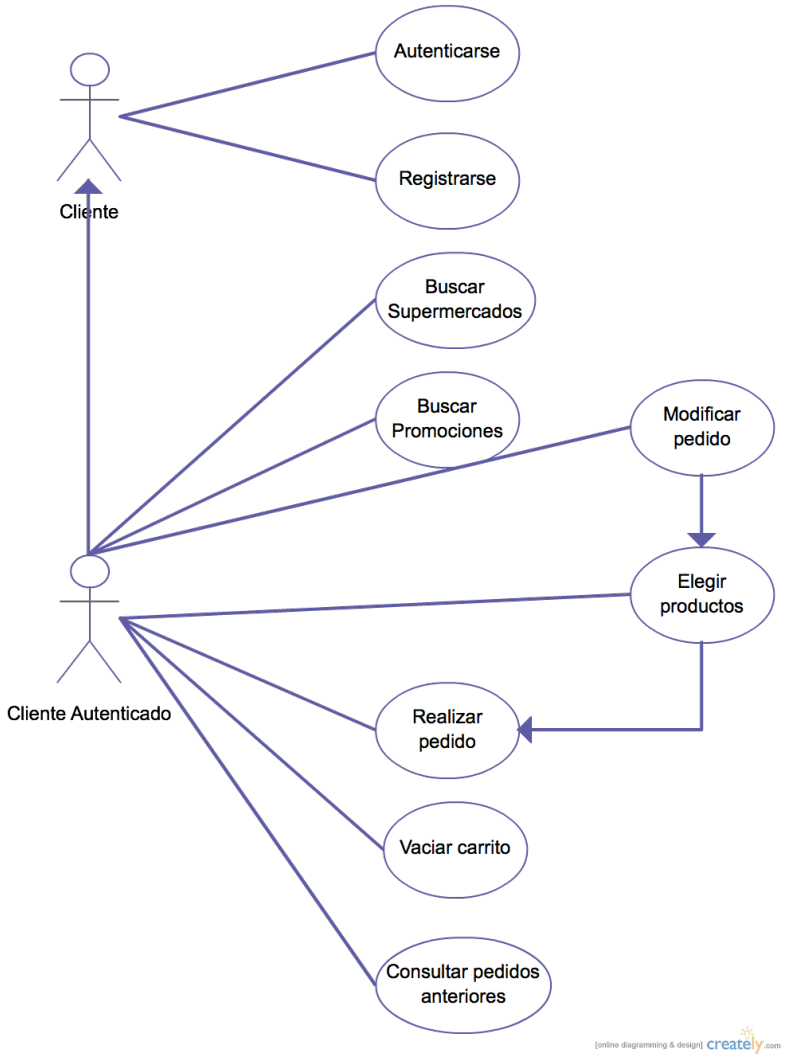


Figura 25

Nombre:	CU01: Autenticarse
Descripción: *Permite a un usuario confirmarle al sistema que sus datos ya están almacenados en la BD, de forma que pueda acceder a la información y realizar pedidos.*	
Actores: *Cliente del supermercado*	
Precondiciones: *El Usuario no está autenticado.*	

El Usuario está registrado en el sistema (CU2: Registrarse).
Flujo Normal: 1. *El cliente del supermercado quiere autenticarse.* 2. *El sistema muestra el formulario de login y espera que sea enviado.* 3. *El cliente rellena el formulario de login y lo envía.* 4. *El sistema comprueba que el cliente es un cliente registrado y además su contraseña es correcta.* 5. *El sistema lanza la aplicación principal.*
Flujo Alternativo: *4.A. El sistema comprueba la validez de los datos, si no son correctos, se avisa al actor de ello permitiéndole que los corrija.*
Poscondiciones: *El usuario está autenticado.*

Nombre:	CU02: Registrarse
Descripción: *Permite a un usuario almacenar sus datos personales y de usuario en el sistema.*	
Actores: *Cliente de supermercado*	
Precondiciones: *El usuario no está registrado en el sistema.* *El usuario no está autenticado (CU1: Autenticarse)*	
Flujo Normal: 1. *Un cliente del supermercado quiere registrarse en el sistema.* 2. *El sistema muestra el formulario de registro con campos de texto vacíos y espera que a que el cliente lo envíe.* 3. *El cliente envía el formulario.* 4. *El sistema comprueba que todos los campos han sido rellenados y además son correctos. Vuelve al CU1: Autentificarse.*	
Flujo Alternativo: *4.A. El sistema comprueba la validez de los datos, si no son los correctos, se avisa al actor de ello permitiéndole que los corrija.*	
Poscondiciones: *El usuario es añadido al sistema de persistencia de la aplicación.*	

Nombre:	CU03: Buscar Supermercado
Descripción:	

Un cliente quiere buscar información de un supermercado.
Actores: *Usuario*
Precondiciones: *Estar autenticado (CU 1: Autentificarse)*
Flujo Normal: 1. *El cliente solicita supermercados buscando por nombre.* 2. *El sistema procesa la petición y muestra los supermercados encontrados, esperando que alguno sea seleccionado.* 3. *El cliente selecciona uno de los supermercados que el sistema le muestra.* 4. *El sistema muestra información básica del supermercado seleccionado.* 5. *El cliente solicita más información acerca del supermercado seleccionado.* **6.** *El sistema le muestra al usuario toda la información que posea del supermercado seleccionado.*
Flujo Alternativo: *2A. El sistema no encuentra ningún supermercado disponible con el nombre introducido y queda a la espera de una nueva petición.*
Poscondiciones: ---
Notas: *Los supermercados disponibles se mostrarán en una interfaz de mapa. Además se permitirá realizar una búsqueda por grupos o individual escribiendo el nombre (o parte del nombre) de uno o varios supermercado en concreto. Ejemplo: "Mobimarket Uría Oviedo", que mostrará ese supermercado en concreto en el mapa. "Gijón", mostrará todos los supermercados de Gijón.*

Nombre:	CU04: Buscar Promociones
Autor:	David RG
Fecha:	26/04/2013
Descripción: *El cliente del supermercado quiere información de las promociones que los supermercados tienen disponibles en un momento dado.*	
Actores: *Cliente del supermercado*	
Precondiciones: *El usuario esté autenticado (CU1: Autenticarse)*	
Flujo Normal:	

1. *El cliente solicita información sobre las promociones disponibles.*
2. *El sistema muestra la promoción que primero caduca de todas las disponibles en un momento dado.*
3. *El cliente pide la siguiente promoción.*
4. *El sistema devuelve la siguiente promoción en fecha de caducidad.*
5. *Volvemos al paso 3.*

Flujo Alternativo:

3.A.1. El cliente pide la anterior promoción.

3.A.2. El sistema muestra la promoción anterior en el tiempo. Si es la promoción que primero caduca, vuelve a mostrar la misma.

3.A.3. Se salta al paso 5 del flujo principal.

4.A. Es la última promoción disponible y entonces vuelve a mostrar la misma.

Poscondiciones:

Nombre:	CU05: Buscar Productos
Autor:	David RG
Fecha:	26/04/2013

Descripción:

El cliente quiere información de los productos que los supermercados ponen a su alcance y comprobar toda la información referente a ellos.

Actores:

Cliente del supermercado

Precondiciones:

El usuario está autenticado (CU1: Autenticarse)

Flujo Normal:

1. *El cliente solicita los productos de un supermercado.*
2. *El sistema muestra información de todas las categorías de productos disponibles.*
3. *El cliente selecciona una categoría.*
4. *El sistema muestra la información básica de todos los productos disponibles de la categoría seleccionada.*
5. *El cliente quiere ampliar información de un producto en concreto.*
6. *El sistema muestra el producto seleccionado con la información detallada.*

Flujo Alternativo:

4.A.1. No hay productos de una categoría y entonces el sistema avisa al usuario de ello.

4.A.2. El sistema queda a la espera de que se seleccione una categoría.

4.A.3. Se vuelve al paso 3.

Poscondiciones:

Nombre:	CU06: Elegir Productos
Autor:	David RG
Fecha:	26/04/2013
Descripción: *El usuario quiere añadir un producto al carrito de la compra.*	
Actores: *Cliente del supermercado.*	
Precondiciones: *El cliente está autenticado (CU1: Autenticarse)*	
Flujo Normal: 1. *El cliente añadir un producto/promoción al carrito de la compra.* 2. *El sistema procesa el producto/promoción seleccionada y la añade al carrito.* 3. *El sistema comprueba si el producto/promoción ya había sido introducido previamente en el carrito. Si es así lo incrementa en una unidad. Sino lo añade nuevo.*	
Flujo Alternativo: --	
Poscondiciones: *Un nuevo producto es añadido al carrito, o bien, un producto previamente añadido es ampliado en una unidad.*	

Nombre:	CU07: Modificar Pedido
Autor:	David RG
Fecha:	26/04/2013
Descripción: *El usuario modifica los productos y/o cantidades de los mismos, una vez son añadidos al pedido.*	
Actores: *Cliente del supermercado.*	
Precondiciones: *El usuario tiene que haber añadido productos o promociones al pedido (CU6: Elegir Productos).*	

Flujo Normal: 1. *El cliente solicita información del carrito de la compra.* 2. *El sistema muestra los productos y promociones, con cantidad y precio añadidos al pedido en ese momento.* 3. *El cliente selecciona un producto de los contenidos en el carrito de la compra.* 4. *El sistema muestra información ampliada del producto y queda a la espera de que se modifiquen las cantidades.* 5. *El cliente modifica la cantidad del producto seleccionado.* 6. *El sistema actualiza el carrito de la compra y le muestra al usuario las unidades actualizadas.*
Flujo Alternativo: *5.A.1. El cliente quiere disminuir las cantidades de un producto que sólo tiene una unidad.* *5.A.2. El sistema ofrece la posibilidad de eliminar el producto del carrito de la compra.* *5.A.3. El cliente confirma que quiere borrar el producto del carrito.* *5.A.4. El sistema el elimina el producto del pedido. Se vuelve al paso 2 del flujo normal.* *5.A.3.A. El cliente rechaza borrar el producto y se vuelve al paso 4 del flujo normal.*
Poscondiciones: *El carrito ha sido actualizado con las variaciones insertadas por el cliente.*

Nombre:	CU08: Realizar Pedido
Autor:	David RG
Fecha:	26/04/2013
Descripción: *El cliente del supermercado confirma el pedido realizado, de forma que se procese para un posterior envío.*	
Actores: *Cliente del supermercado.*	
Precondiciones: *El usuario está autenticado.*	
Flujo Normal: 1. *El cliente solicita información del carrito de la compra.* 2. *El sistema muestra todos los productos que ha añadido y el precio total del pedido, quedando a expensas que se confirme el pido.* 3. *El cliente quiere confirmar el pedido.* 4. *El sistema muestra el precio total del pedido esperando confirmación.* 5. *El cliente confirma el pedido.* 6. *El sistema almacena el pedido e informa al cliente de que su pedido está siendo procesado.*	

Flujo Alternativo:
4.A. El usuario rechaza la confirmación del pedido y se vuelve al paso 2. *5.A. No existen productos en el carrito, con lo cual se avisa al usuario de ello y se vuelve al paso 2 del flujo normal.*
Poscondiciones:
El pedido ha sido almacenado correctamente en la base de datos.

Nombre:	CU09: Vaciar el pedido
Autor:	David RG
Fecha:	26/04/2013
Descripción:	
Se eliminarán todos los productos que previamente se hayan introducido en el carrito de la compra, de modo que se proceda de nuevo a la creación de un pedido desde cero.	
Actores:	
Cliente del supermercado.	
Precondiciones:	
El usuario está autenticado (CU1: Autenticarse)	
Flujo Normal:	
1. *El cliente solicita información de los productos que tiene insertados en el carrito de la compra.* 2. *El sistema muestra los productos y promociones introducidos en el carrito hasta el momento.* 3. *El cliente quiere vaciar el carrito de la compra.* 4. *El sistema permanece a la espera de la confirmación de vaciar el carrito de la compra.* 5. *El cliente confirma que quiere vaciar el carrito.* **6.** *El sistema elimina todos los productos que hay en el momento de la confirmación en el carrito.*	
Flujo Alternativo:	
4.A. El usuario cancela la confirmación y se volverá al paso 2 del flujo principal.	
Poscondiciones:	
El carrito no contiene ningún elemento.	

Nombre:	CU10: Consultar Pedidos Anteriores
Autor:	David RG
Fecha:	16/04/2013

Descripción: *Se accederá a todos los pedidos que un usuario registrado ha realizado desde el momento de su registro hasta un momento dado.*
Actores: *Cliente del supermercado.*
Precondiciones: *El usuario está autenticado (CU01: Autenticarse)*
Flujo Normal: 1. *El cliente accede al carrito de la compra.* 2. *El sistema muestra todos los productos que ha añadido hasta ese momento en el carrito y permanece a la espera de se le solicite información de pedidos anteriores.* 3. *El usuario solicita información de los pedidos realizados previamente.* 4. *El sistema procesa la petición, identifica él cliente que la ha solicitado y le muestra todos los pedidos que ha realizado previamente.*
Flujo Alternativo: --
Poscondiciones: ---

4.2.4. REQUISITOS DE INTERFAZ DE USUARIO

En esta sección se recogen los requisitos que afectan a la interfaz de usuario. Es lo que se desarrollaría a modo de prototipo sin ningún tipo de funcionalidad lógica.

Tiene bastante importancia ceñirse a ellos, ya que tienen mucho parecido con los componentes de interfaz utilizados en la aplicación nativa (facilitada por el CTIC) con la que se va a comprar MobiMarket Móvil una vez desarrollada en Titanium.

Se han creado una serie de prototipos para guiar la construcción de las interfaces de usuario de la aplicación MobiMarket Móvil. Dichos prototipos se pueden consultar en los apéndices del presente documento.

- IU 1: La aplicación se mostrará al usuario en pestañas
  - IU 1.1: Existirán, en la aplicación principal, cuatro pestañas. Una para la localización, otra para los productos, otra para las promociones y otra para el carrito de la compra.
- IU 2: La localización del usuario y de los supermercados se mostrará en un interfaz gráfico de mapa.
  - IU 2.1: Existirá un campo de texto para buscar supermercados.
    - IU 2.1.1: Los supermercados buscados se mostrarán en una tabla con un mínimo de cinco y un máximo de 7 filas al mismo tiempo en pantalla
  - IU 2.2: Los supermercados que se muestren en el mapa en un momento dado serán *clickables* por parte del usuario y estarán en color rojo.
    - IU 2.2.1: Cada supermercado mostrará un diálogo contextual con el nombre del supermercado y la posibilidad de acceder a una nueva pantalla con las características del supermercado.

  - IU 2.3: La pantalla de supermercado tendrá una imagen, el nombre, el teléfono, la dirección y el horario laboral del centro.
- IU 3: Las categorías y los productos se mostrarán en tablas, con un mínimo de seis ítems y un máximo de nueve.
  - IU 3.1: Cada fila de categoría tendrá una imagen y el propio nombre de la categoría.
    - IU 3.1.1: Cada fila de categoría tendrá un botón indicador con una flecha a la derecha. (*Con esto se pretende dar a entender al usuario que clickar en la fila nos lleva a otra pantalla)*.
  - IU 3.2: Cada fila de producto tendrá una imagen, el nombre del producto y el precio.
    - IU 3.2.1: Cada fila de producto tendrá un botón indicador con una flecha hacia la derecha.
    - IU 3.2.2: Cada producto podrá mostrarse en una nueva pantalla con una imagen, un nombre, una descripción, un precio y un botón que permita añadir dicho producto al carro de la compra.
- IU 4: Las promociones se mostrarán una colección de imágenes. Y se utilizará un botón circular con un signo '+', en la misma pantalla, para añadir la promoción al carrito.
  - IU 4.1: En la ventana de promociones se mostrará además el nombre, la descripción y el precio de cada promoción.
- IU 5: La ventana del carrito mostrará una tabla con los productos añadidos al mismo.
  - IU 5.1: El carrito mostrará un máximo de nueve filas en pantalla para mostrar al usuario los productos que ha insertado en el mismo.
  - IU 5.2: Cada fila de la tabla del carrito mostrará la imagen del producto, el nombre, la cantidad y el precio de cada producto.
    - IU 5.2.1: Cada fila del carrito tendrá un botón indicativo con una flecha señalando hacia la derecha.
    - IU 5.2.2: Cada producto del carrito abrirá una ventana dónde se muestre el nombre del producto, la descripción, el precio por unidad, la cantidad, el precio total del producto, y un par de botones que permitan sumar o restar productos.
    - IU 5.2.3: Existirá una y sólo una fila para cada producto añadido.
  - IU 5.3: En la ventana del carrito se mostrará el precio total del pedido.
  - IU 5.4: La ventana del carrito tendrá un botón para vaciar el carrito y otro para confirmar el pedido.
  - IU 5.5: Se verá en todo momento el número de productos distintos que hay insertados en el carrito.
- IU 6: Existirá una ventana para que el usuario pueda registrarse.
  - IU 6.1: El registro se realizará mediante un formulario.
    - IU 6.1.1: Los campos de formulario relativos a la contraseña de usuario mostrarán ocultados los caracteres que teclee el usuario.
- IU 7: Existirá una ventana para que el usuario pueda autentificarse.
  - IU 7.1: La ventana de autentificación será la primera que aparezca cuando se arranque la aplicación.
  - IU 7.2: La ventana tendrá dos campos de texto, uno para insertar el usuario y otro para la contraseña.
    - IU 7.2.1: El campo de contraseña ocultará los caracteres que el usuario escriba.
  - IU 7.3: La ventana tendrá dos botones, uno para enviar el login y otro para acceder a la ventana de registro.

- IU 8: Un usuario podrá confirmar en pedido en tres o menos clic independientemente del lugar de la aplicación dónde se encuentre.
- IU 9: Un usuario podrá eliminar uno o todos los productos del pedido en tres o menos clic independientemente del lugar de la aplicación dónde se encuentre.

4.2.5. REQUISITOS NO FUNCIONALES

Los requisitos no funcionales tales como la fiabilidad, escalabilidad, adaptabilidad, seguridad, integridad y otros "-ilities" quedan fuera del alcance de este proyecto, ya que dependen excesivamente de la BBDD que ya es un producto implementado y en producción.

- RnF 1: Las consultas a la BBDD se realizarán en un tiempo menor a los tres segundos.
  - RnF 1.1: Deberá de cachearse la información consultada, de forma que una consulta sólo tenga que realizarse una vez. Por ejemplo, si ya se han cargado las categorías, no será necesario recargarlas más veces.
  - RnF 1.2: El tiempo de carga para el cambio entre ventanas y pestañas se deberá ser menor de un segundo.
- RnF 2: La aplicación funcionará en entornos IOS5 o superior y Android 2.2 o superior.
  - RnF 2.1: El entorno de desarrollo correrá sobre un sistema operativo Mac OS X Lion.
- RnF 3: La aplicación será desarrollada en el entorno de desarrollo Titanium Studio.
  - RnF 3.1: La aplicación será desarrollada utilizando el framework Alloy para el entorno de desarrollo Titanium Studio, utilizando el patrón arquitectónico MVC.
- RnF 4: El sistema tendrá un control de acceso a la aplicación a través de login.
  - RnF 4.1: El acceso a la base datos se realizará con la contraseña encriptada a través de un sistema hexadecimal en base 64.
  - RnF 4.2: No podrán existir dos usuarios con el mismo nombre de usuario.
  - RnF 4.3: El nombre de usuario sólo podrá contener letras y números.
- RnF 5: La aplicación estará disponible en español y en inglés.

4.2.6. RESTRICCIONES

A continuación se muestra una serie de restricciones a tener en cuenta durante el desarrollo de la aplicación.

- El sistema se desarrollará en un equipo Mac OS X compatible con la herramienta Titanium Studio, en la que será obligatorio desarrollar la aplicación. Es imprescindible, puesto que el objeto de este sistema no sólo es la generación de una aplicación para un supermercado, sino también comprobar las posibilidades de el entorno de desarrollo multiplataforma.
- La aplicación será desarrollada a través del entorno de desarrollo Titanium Studio.
- Se utilizará el *framework* Alloy diseñado para Titanium Studio, que fuerza al programador a utilizar el patrón arquitectónico MVC.
- La aplicación se comunicará con una base de datos no relacional. Concretamente con una base de datos *CouchDB*.
- La aplicación deberá ejecutarse en terminales Android y terminales iPhone.
- El funcionamiento e interfaces será similar al de la aplicación nativa facilitada por el CTIC, para comparar posteriormente rendimientos.
- La aplicación tendrá algún tipo de registro y autentificación.
- La aplicación sólo correrá correctamente en terminales móviles Android con la versión del operativo 2.2 o superior y en iPhone 4S o inferior.

- La BBDD que utilizará la aplicación será una base de datos noSQL. No funcionará con una BBDD SQL.

4.3. DISEÑO

En esta sección se establecerá un diseño muy simplista de lo que será la aplicación. No pretende ser tan detallado ni complejo como el diseño que sigue la metodología Métrica V3.

Simplemente se establecerá una visión global del sistema y su comunicación con otras aplicaciones externas, con las que se forma el proyecto global de MobiMarket.

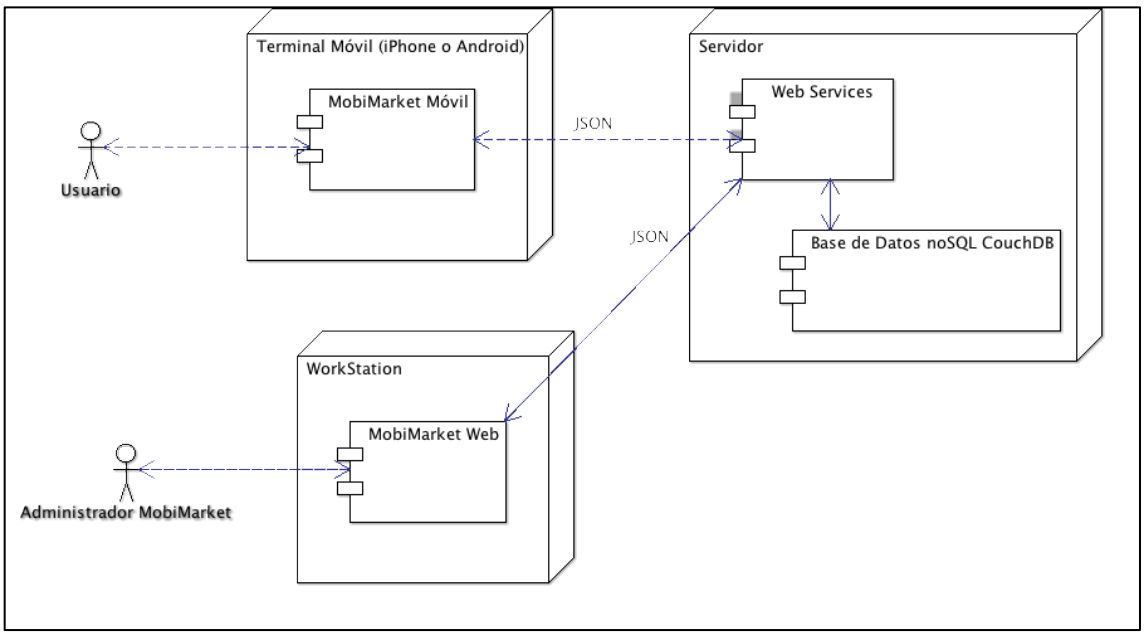


Figura 26

El software se comunicará con una base de datos noSQL. Para ello se utiliza la comunicación mediante el envío de consultas a los servicios web (PUT, POST o GET).

La aplicación intercambiará información a través de ficheros JSON, tanto para recibir información desde la base de datos (por ejemplo para recibir las categorías existentes), como para enviarla (enviar un pedido).

A continuación se muestran las peticiones necesarias para obtener toda la información necesaria de la base de datos. Para ello se mostrará el tipo de consulta (PUT, POST o GET), la URL (dirección del servicio web) y una descripción de la consulta. También se muestra el formato del JSON que se recibe o se envía.

Se recomienda ver el apéndice 7.5. "Consultas de ejemplo a los WebServices de la BBDD" del presente documento.

Las partes relativas a la aplicación web y al servidor ya están probadas y en funcionamiento. En este proyecto se desarrollará la parte móvil. La aplicación para los terminales móviles desde dónde los clientes del supermercado podrán realizar sus compras.

La parte móvil se ha separado en cuatro subsistemas más simples para ir desarrollando y probando cada una de las partes.

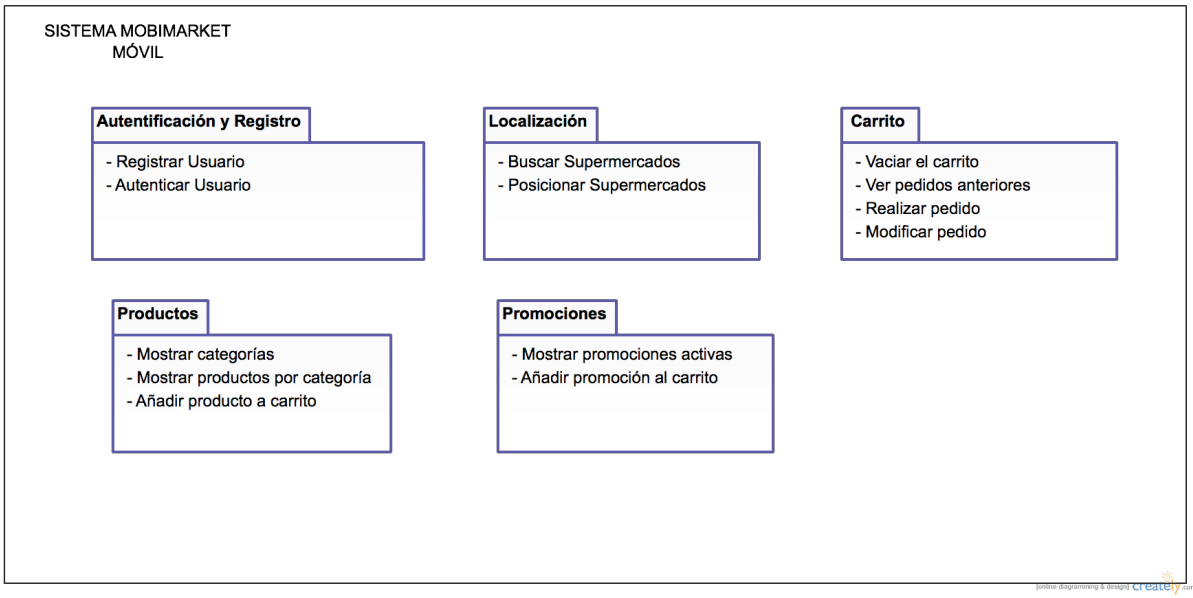


Figura 27

En la figura 27, se pueden comprobar el número de subsistemas en los que se ha separado el sistema completo, con su respectiva funcionalidad a grandes rasgos.

4.4. PRUEBAS

En esta sección se proponen una serie de pruebas que se realizarán y corregirán a lo largo del desarrollo de la aplicación. No se expresa un documento específico para el diseño de pruebas y otro para la ejecución. Las pruebas a realizar serán simples y orientadas a requisitos. La falta de tiempo ha provocado que no se hayan podido realizar pruebas más exhaustivas.

Las pruebas se separarán por cada uno de los subsistemas que forman el proyecto completo.

4.4.1. SUBSISTEMA LOCALIZACIÓN

En primer lugar se documentarán las pruebas que atañen al subsistema de localización de acuerdo a la distribución del sistema MobiMarket Móvil como se puede ver en la *Especificación de Requisitos* del presente documento.

REQUISITOS DE PRUEBA

- RP 1: Búsqueda de supermercados
  - RP 1.1: Se introduce el nombre de un supermercado
  - RP 1.2: Todos los campos de un supermercado contienen información y es accesible.
- RP 2: Visualización de supermercados
  - RP 2.1: Los supermercados se muestran en el mapa.

CASOS DE PRUEBA

ID:	**Descripción:**	**Salida Obtenida:**
CP 1.1	Comprobar que se introduce el nombre completo de un supermercado y este se muestra, y además aparece este únicamente en el mapa	Pasa
	**Entradas:** nombreBusqueda = "MobiMarket Gijón"	**Salida Esperada:** [MobiMarket Gijón]
ID:	**Descripción:**	**Salida Obtenida:**
CP 1.2	Comprobar que no se introduce ningún carácter y se muestran todos los supermercados, y además todos ellos aparecen en el mapa	Excepción: NullPointerException
	**Entradas:** nombreBusqueda = ""	**Salida Esperada:** [MobiMarket Oviedo, MobiMarket Gijón 1, MobiMarket Gijón 2]
ID:	**Descripción:**	**Salida Obtenida:**
CP 1.3	Comprobar que se introduce el nombre completo de un supermercado y este se muestra, y además aparece este únicamente en el mapa	Pasa
	**Entradas:** nombreBusqueda = “Gijón"	**Salida Esperada:** [MobiMarket Gijón 1, MobiMarket Gijón 2]

ID:	Descripción:	Salida Obtenida:
CP 1.4	Comprobar que todos los campos de supermercado contienen información para un supermercado dado, y además es mostrado en el mapa.	Pasa
	**Entradas:** idSupermercado = "16d9f0b7ff0ffe2f202ff55e97008ff2"	**Salida Esperada:** address = "C Uria 22", bussinessHours = "24 H", coordinate.latitude = 43.5382, coordinate.longitude = -5.6546, created_at = 2013-03-19T16:55:38.612Z, name = "MobiMarket Gijón", phone = "9854754578", type = "restaurant"

CASOS DE PRUEBA FALLIDOS:

ID:	Descripción:	Salida Obtenida:
CP 1.2	Comprobar que no se introduce ningún carácter y se muestran todos los supermercados, y además todos ellos aparecen en el mapa	Pasa
	**Entradas:** nombreBusqueda = ""	**Salida Esperada:** [MobiMarket Oviedo, MobiMarket Gijón 1, MobiMarket Gijón 2]
Motivo del fallo: FALLO1: El campo nombreBusqueda se inicializaba a *null*. Se ha cambiado la inicialización a la cadena vacía (“”).		

4.4.2. SUBSISTEMA PRODUCTOS

A continuación se documentarán las pruebas que atañen al subsistema de productos de acuerdo a la distribución del sistema MobiMarket Móvil como se puede ver en la *Especificación de Requisitos* del presente documento.

REQUISITOS DE PRUEBA

- RP 2: Búsqueda de productos
  - RP 2.1: Una categoría tiene información de todos sus campos.
    - RP 2.1.1: Se obtiene información de una categoría que tiene productos.
    - RP 2.1.2: Se obtiene información de una categoría sin productos.
  - RP 2.2: Todos los campos de un producto contienen información.
  - RP 2.3: Se añade un producto al carrito.
    - RP 2.3.1: Se añade un producto ya existente al carrito.

CASOS DE PRUEBA

ID:	Descripción:	Salida Obtenida:
CP 2.1	Comprobar que la aplicación recibe información de las categorías.	Pasa

	**Entradas:** []	**Salida Esperada:** [Bebidas Alcoholicas, Bebidas no alcoholicas, Carnicería, Charcutería, Especias, Fruetería, Lacteos, Pastelería, Panadería, Pescadería, Platos Preparados, Verduras]
ID: CP 2.2	**Descripción:** Comprobar que todas las categorías contienen información en todos sus campos.	**Salida Obtenida:** Pasa
	**Entradas:** idCategoria = “16d9f0b7ff0ffe2f202ff55e97013bc2”	**Salida Esperada:** Id = “16d9f0b7ff0ffe2f202ff55e97013bc2”, created_at = “2013-03-21T14:50:47.783Z”, name = “Bebidas no alcohólicas”, type = “category”
ID: CP 2.3	**Descripción:** Comprobar una categoría que no existe.	**Salida Obtenida:** Pasa
	**Entradas:** idCategoria = “estacategorianoexiste”	**Salida Esperada:** NO_CATEGORIA
ID: CP 2.4	**Descripción:** Una categoría que contiene productos los devuelve correctamente.	**Salida Obtenida:** Pasa
	**Entradas:** idCategoria = "16d9f0b7ff0ffe2f202ff55e97013bc2"	**Salida Esperada:** [Cerveza Mahou sin alcohol, Agua mineral]
ID: CP 2.4.1	**Descripción:** Una categoría que no contiene productos no provoca un error de consistencia.	**Salida Obtenida:** Pasa
	**Entradas:** idCategoria = "16d9f0b7ff0ffe2f202ff55e97013cg3"	**Salida Esperada:** NO_PRODUCTS
ID: CP 2.5	**Descripción:** Un producto nuevo es añadido correctamente al carrito.	**Salida Obtenida:** Pasa
	**Entradas:** idProducto = “16d9f0b7ff0ffe2f202ff55e9701f3be”, cantidadProducto = 0, numeroDeProductosEnCarrito = 0	**Salida Esperada:** cantidadProducto = 1, numeroDeProductosEnCarrito = 1

ID: CP 2.5.1	**Descripción:** Un producto nuevo es añadido correctamente al carrito.	**Salida Obtenida:** Pasa
	**Entradas:** idProducto = "16d9f0b7ff0ffe2f202ff55e9701f3be", cantidadProducto = 0, numeroDeProductosEnCarrito = 3	**Salida Esperada:** cantidadProducto = 1, numeroDeProductosEnCarrito = 4
ID: CP 2.6	**Descripción:** Un producto ya existente en el carrito es añadido al carrito.	**Salida Obtenida:** Pasa
	**Entradas:** idProducto = "16d9f0b7ff0ffe2f202ff55e9701f3be", cantidadProducto = 1, numeroDeProductosEnCarrito = 3	**Salida Esperada:** cantidadProducto = 2, numeroDeProductosEnCarrito = 3
ID: CP 2.6.1	**Descripción:** Un producto ya existente en el carrito es añadido al carrito.	**Salida Obtenida:** Pasa
	**Entradas:** idProducto = "16d9f0b7ff0ffe2f202ff55e9701f3be", cantidadProducto = 7, numeroDeProductosEnCarrito = 3	**Salida Esperada:** cantidadProducto = 8, numeroDeProductosEnCarrito = 3

4.4.3. SUBSISTEMA PROMOCIONES

A continuación se documentarán las pruebas que atañen al subsistema de productos de acuerdo a la distribución del sistema MobiMarket Móvil como se puede ver en la *Especificación de Requisitos* del presente documento.

REQUISITOS DE PRUEBA

- RP 3: Búsqueda de promociones
  - RP 3.1: Una promoción tiene información en todos sus campos.
  - RP 3.2: Una promoción aparece si se encuentra activa en una fecha concreta.
  - RP 3.3: Se añade una promoción al carrito.
    - RP 3.3.1: Se añade una promoción ya existente al carrito.

CASOS DE PRUEBA

ID: CP 3.1	**Descripción:** Se pide una promoción que se encuentra en la fecha activa.	**Salida Obtenida:** Pasa
	**Entradas:** idPromo = "16d9f0b7ff0ffe2f202ff55e9701160f" fecha = "2013-03-20T00:00:00.000Z"	**Salida Esperada:** Id = "16d9f0b7ff0ffe2f202ff55e9701160f", description = "Pan de leña de cuarto más 100g de jamón ibérico", from="2013-03-20T00:00:00.000Z", to="2014-03-20T00:00:00.000Z", price = 5, name = "Promo Bocadillo de Jamón", itemType = "promo"
ID: CP 3.1.1	**Descripción:** Se pide una promoción con fecha de fin a un día antes de que comience.	**Salida Obtenida:** Pasa
	**Entradas:** idPromo = "16d9f0b7ff0ffe2f202ff55e9701160f", fecha = "2013-03-19T00:00:00.000Z"	**Salida Esperada:** INACTIVE_PROMO
ID: CP 3.1.2	**Descripción:** Se pide una promoción con fecha de fin a un día despues de que terminara.	**Salida Obtenida:** Pasa
	**Entradas:** idPromo = "16d9f0b7ff0ffe2f202ff55e9701160f", fecha = "2014-03-21T00:00:00.000Z"	**Salida Esperada:** INACTIVE_PROMO

CASOS DE PRUEBA FALLIDOS:

- Ninguno.

4.4.4. SUBSISTEMA CARRITO

A continuación se documentarán las pruebas que atañen al subsistema de pedidos de acuerdo a la distribución del sistema MobiMarket Móvil como se puede ver en la *Especificación de Requisitos* del presente documento.

REQUISITOS DE PRUEBA

- RP 4: Visualización de Pedidos
  - RP 4.1: El campo de cantidad de un pedido es modificado.
    - RP 4.1.1: El campo cantidad es aumentado.
    - RP 4.1.2: El campo cantidad es disminuido.
    - RP 4.1.3: El campo cantidad es disminuido a cero o menor que cero.
  - RP 4.2: Un carrito de la compra con productos, pasa a estar vacío.
  - RP 4.3: El contador de pedidos se incrementa cuando se añade un nuevo pedido.
  - RP 4.4: Todos los pedidos previamente realizados por un usuario contienen información en todos sus campos.
  - RP 4.5: Un producto del carrito contiene información en todos sus campos.

CASOS DE PRUEBA

ID: CP 4.1	**Descripción:** La cantidad de un producto es ampliada en dos unidades y se muestra en el interfaz gráfico el cambio.	**Salida Obtenida:** Pasa
	**Entradas:** producto = Cerveza sin alcohol, cantidad = 2, precioTotal = 3.27€, ampliarCantidad = 1, amplicarCantidad = 1	**Salida Esperada:** producto = Cerveza sin alcohol, cantidad = 4, precioTotal = 4.68€
ID: CP 4.2	**Descripción:** La cantidad de un producto es disminuida en dos unidades y se muestra en el interfaz gráfico.	**Salida Obtenida:** Pasa
	**Entradas:** Producto = Jamón Ibérico, cantidad = 3, precioTotal = 15€, diminuirCantidad = 1, disminuirCantidad = 1	**Salida Esperada:** Producto = Jamón Ibérico, Cantidad = 1, Precio total = 5€
ID: CP 4.2.1	**Descripción:** La cantidad de un producto es disminuida en una unidad llegando a cero y se muestra en el interfaz gráfico.	**Salida Obtenida:** Pasa. El producto se elimina del carrito, pero se sigue mostrando en el interfaz.
	Entradas: Producto = Promo Bocadillo Jamón, cantidad = 1, precioTotal = 22,3 €, diminuirCantidad = 1, productosEnCarrito = 4	**Salida Esperada:** NO_PRODUCT, precioTotal = 17,3€, productosEnCarrito = 3
ID: CP 4.3	**Descripción:** Se vacía el carrito de la compra, que contiene previamente varios productos, y se actualiza el interfaz.	**Salida Obtenida:** Pasa
	**Entradas:** productosEnCarrito = 3, precioTotal = 23,5€	**Salida Esperada:** NO_PRODUCTS, precioTotal = 0 €

ID: CP 4.4	**Descripción:** Se realiza un nuevo pedido desde la aplicación.	**Salida Obtenida:** Falla. Error desde la BBDD. Posiblemente, el JSON que se envía no este correctamente formado.
	**Entradas:** totalPedidosUsuario = 3	**Salida Esperada:** totalPedidosUsuario = 4
ID: CP 4.5	**Descripción:** Se realiza una petición al servidor de los pedidos realizados por un usuario y se muestran en pantalla.	**Salida Obtenida:** Pasa
	**Entradas:** User = “agus”	**Salida Esperada:** totalPedidosUsuario = 7, pedidos = [pedido003, pedido005, pedido007, pedido008, pedido009, pedido 010, pedido015]

CASOS DE PRUEBA FALLIDOS:

ID: CP 4.2.1	**Descripción:** La cantidad de un producto es disminuida en una unidad llegando a cero y se muestra en el interfaz gráfico.	**Salida Obtenida:** Pasa.
	**Entradas:** Producto = Promo Bocadillo Jamón, cantidad = 1, precioTotal = 22,3 €, diminuirCantidad = 1, productosEnCarrito = 4	**Salida Esperada:** NO_PRODUCT, precioTotal = 17,3€, productosEnCarrito = 3
Motivo del fallo: FALLO1: No se actualizaba el interfaz de carrito tras eliminar los productos. Se ha modificado para que sí se haga.		
ID: CP 4.4	**Descripción:** Se realiza un nuevo pedido desde la aplicación.	**Salida Obtenida:** Pasa.
	**Entradas:** totalPedidosUsuario = 3	**Salida Esperada:** totalPedidosUsuario = 4
Motivo del fallo: FALLO1: El JSON era correcto, pero no se estaba realizando la autentificación en la consulta. Con lo cuál, no se tenían permisos de escritura sobre la BBDD.		

4.4.5. SUBSISTEMA AUTENTIFICACIÓN Y REGISTRO

A continuación se documentarán las pruebas que atañen al subsistema de autentificación y registro de acuerdo a la distribución del sistema MobiMarket Móvil como se puede ver en la *Especificación de Requisitos* del presente documento.

REQUISITOS DE PRUEBA

- RP 5: Se registra un nuevo usuario
  - RP 5.1: Se registra un usuario con todos los campos menos uno.
  - RP 5.2: Se registra un usuario con un campo mal formado.
    - RP 5.2.1: Con un CP distinto a cinco dígitos.
    - RP 5.2.2: Con un correo sin arroba y con punto.
    - RP 5.2.3: Con un correo con arroba y sin punto.
    - RP 5.2.4: Con un número distinto a nueve dígitos.
  - RP 5.3: Se registra un usuario con un nombre de usuario ya introducido en la BD.
  - RP 5.4: Se registra un usuario con todos los datos correctos.
  - RP 5.5: Se autentica un usuario
    - RP 5.5.1: Se autentica un usuario conocido para el servidor con el pass correcto.
    - RP 5.5.2: Se autentica un usuario desconocido para el servidor.
    - RP 5.5.3: Se autentica un usuario conocido para el servidor con el pass incorrecto.

CASOS DE PRUEBA

ID:	**Descripción:**	**Salida Obtenida:**
CP 5.1	Se registra un usuario con todos los datos excepto el nombre y se comprueba el mensaje de error.	Pasa
	**Entradas:** nombre = “”, apellido = “rodriguez”, dirección = “pola de lena”, cp=”33630”, teléfono = “666555444”, fecha_nacimiento = “30-05-85”, usuario = “David”, contraseña = “David”, contraseñaO = “David”, sexo = “Male” , correo=”david@gmail.com”	**Salida Esperada:** Mensaje de error: “Todos los campos son obligatorios”. El usuario no está en la BBDD.
ID:	**Descripción:**	**Salida Obtenida:**
CP 5.2	Se registra un usuario con un código postal compuesto por letras.	Pasa
	**Entradas:** nombre = “david”, apellido = “rodriguez”, dirección = “pola de lena”, cp=”holas”, teléfono = “666555444”, fecha_nacimiento = “30-05-85”, usuario = “David”, contraseña = “David”, contraseñaO = “David”, sexo = “Male” , correo=”david@gmail.com”	**Salida Esperada:** Mensaje de error. “El código postal debe de estar formado por cinco dígitos Ej: 33630”. El usuario no está en la BBDD

ID: CP 5.2.1	**Descripción:** Se registra un usuario con un código postal compuesto por 4 dígitos	**Salida Obtenida:** Pasa.
	**Entradas:** nombre = “david”, apellido = “rodriguez”, dirección = “pola de lena”, cp=”3363”, teléfono = “666555444”, fecha_nacimiento = “30-05-85”, usuario = “David”, contraseña = “David”, contraseñaO = “David”, sexo = “Male”, correo=”david@gmail.com”	**Salida Esperada:** Mensaje de error. “El código postal debe de estar formado por cinco dígitos Ej: 33630”. El usuario no está en la BBDD
ID: CP 5.3	**Descripción:** Se registra un usuario con un correo electrónico sin arroba.	**Salida Obtenida:** Pasa
	**Entradas:** nombre = “david”, apellido = “rodriguez”, dirección = “pola de lena”, cp=”3363”, teléfono = “666555444”, fecha_nacimiento = “30-05-85”, usuario = “David”, contraseña = “David”, contraseñaO = “David”, sexo = “Male”, correo=”davidgmail.com”	**Salida Esperada:** Mensaje de error. “El email no es correcto. Ejemplo: correo@me.com”. El usuario no se registra en la BBDD.
ID: CP 5.3.1	**Descripción:** Se registra un usuario con un correo electrónico sin punto después de la arroba.	**Salida Obtenida:** Pasa.
	**Entradas:** nombre = “david”, apellido = “rodriguez”, dirección = “pola de lena”, cp=”3363”, teléfono = “666555444”, fecha_nacimiento = “30-05-85”, usuario = “David”, contraseña = “David”, contraseñaO = “David”, sexo = “Male”, correo=”david@gmailcom”	**Salida Esperada:** Mensaje de error. “El email no es correcto. Ejemplo: correo@me.com”. El usuario no se registra en la BBDD.
ID: CP 5.3.2	**Descripción:** Se registra un usuario con un correo con dos puntos tras la arroba.	**Salida Obtenida:** Falla. Usuario registrado en la BBDD.
	**Entradas:** nombre = “david”, apellido = “rodriguez”, dirección = “pola de lena”, cp=”3363”, teléfono = “666555444”, fecha_nacimiento = “30-05-85”, usuario = “David”, contraseña = “David”, contraseñaO =	**Salida Esperada:** Mensaje de error. “El email no es correcto. Ejemplo: correo@me.com”. El usuario no se registra en la BBDD.

	"David", sexo = "Male", correo="david@gmail.c.om"	
ID: CP 5.4	**Descripción:** Se registra un usuario correctamente.	**Salida Obtenida:** Pasa
	**Entradas:** nombre = "david", apellido = "rodriguez", dirección = "pola de lena", cp="3363", teléfono = "666555444", fecha_nacimiento = "30-05-85", usuario = "David", contraseña = "David", contraseñaO = "David", sexo = "Male", correo="david@gmail.com"	**Salida Esperada:** El usuario se registra correctamente en la BBDD.
ID: CP 5.5	**Descripción:** Se registra un usuario con la contraseña de confirmación distinta.	**Salida Obtenida:** Pasa
	**Entradas:** nombre = "david2", apellido = "rodriguez", dirección = "pola de lena", cp="3363", teléfono = "666555444", fecha_nacimiento = "30-05-85", usuario = "David", contraseña = "David", contraseñaO = "David2", sexo = "Male", correo="david@gmail.com"	**Salida Esperada:** Mensaje de error. "Las contraseñas no coinciden". El usuario no se registra en la BBDD.
ID: CP 5.6	**Descripción:** Se registra un usuario con un nombre de usuario ya registrado en la BBDD.	**Salida Obtenida:** Pasa
	**Entradas:** nombre = "agus", apellido = "rodriguez", dirección = "pola de lena", cp="3363", teléfono = "666555444", fecha_nacimiento = "30-05-85", usuario = "David", contraseña = "David", contraseñaO = "David", sexo = "Male", correo="david@gmail.com"	**Salida Esperada:** Mensaje de error. "El nombre de usuario elegido ya está siendo utilizado". El usuario no se registra en la BBDD.
ID: CP 5.7	**Descripción:** Se autentifica un usuario existente con la contraseña incorrecta.	**Salida Obtenida:** Pasa
	**Entradas:** usuario = "agus", pass = "lolo", login = false	**Salida Esperada:** Mensaje de error. "El usuario o la contraseña son incorrectas". login = false

ID:	Descripción:	Salida Obtenida:
CP 5.7.1	Se autentifica un usuario existente con la contraseña correcta.	Pasa
	**Entradas:** usuario = “agus”, pass = “agus”, login = false	**Salida Esperada:** login = true

CASOS DE PRUEBA FALLIDOS:

ID:	Descripción:	Salida Obtenida:
CP 5.3.2	Se registra un usuario con un correo con dos puntos tras la arroba.	Falla. Usuario registrado en la BBDD.
	**Entradas:** nombre = “david”, apellido = “rodriguez”, dirección = “pola de lena”, cp=”3363”, teléfono = “666555444”, fecha_nacimiento = “30-05-85”, usuario = “David”, contraseña = “David”, contraseñaO = “David”, sexo = “Male”, correo=”david@gmail.c.om”	**Salida Esperada:** Mensaje de error. “El email no es correcto. Ejemplo: correo@me.com”. El usuario no se registra en la BBDD.
Motivo del fallo: FALLO1: No se comprobaba que sólo un punto puede ir en la parte derecha de la arroba.		

4.5. MANUAL DEL DESARROLLADOR

En primer lugar se expondrán una serie de manuales para facilitar la instalación del entorno de desarrollo y la configuración de alguna herramienta que no resulta ser del todo intuitiva.

A continuación se mostrará un manual de configuración básico del entorno de desarrollo, para dejarlo listo para comenzar a codificar aplicaciones. También se incluyen un par de manuales, con utilidades para el desarrollo de aplicaciones, que siendo bastante sencillas, han sido difíciles de encontrar por la red, como introducir un tipo de fuente personalizada a las aplicaciones o indicarle, a través de los ficheros de configuración nativos, que posiciones admiten las aplicaciones en los dispositivos móviles.

4.5.1. INSTALACIÓN Y CONFIGURACIÓN DE TITANIUM STUDIO

En primer se deberá descargar el entorno de desarrollo de Titanium Studio. Está disponible desde su página web[3] y es completamente gratuito. Existen versiones para los sistemas operativos Mac, Windows y Linux. Para acceder a los recursos para desarrolladores es obligatorio registrarse[4]. Todo registro es también libre de pago.

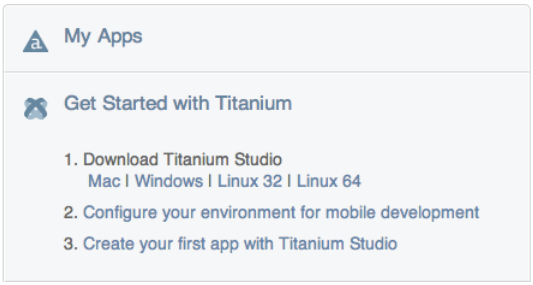


Figura 28

Se abrirá el instalador y seguiremos los pasos que nos muestra hasta concluir la instalación.

Una vez concluida, cada vez que el entorno sea abierto, habrá que indicar cuál es el espacio de trabajo (o workspace) dónde serán almacenadas localmente las aplicaciones que se codifiquen. El entorno de desarrollo de Titanium Studio, es una variante del entorno Eclipse[5] y esta es una de las múltiples coincidencias con dicho entorno.

[3] *https://my.appcelerator.com/resources*

[4] *https://my.appcelerator.com/auth/signup*

[5] http://www.eclipse.org/

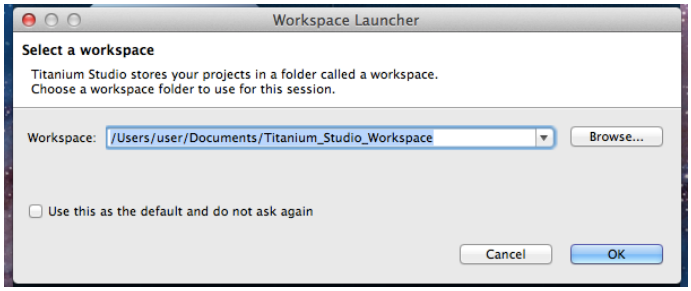


Figura 29

También habrá que autenticarse con los credenciales introducidos previamente en el registro. De esta forma se irán almacenando los ficheros y estadísticas de las aplicaciones a través de un servicio de Cloud Computing.

Figura 30

Una vez abierta la herramienta es muy recomendable tenerla actualizada. Para ello, a través del menú *Help* se podrá acceder a los dos submenús de actualización *Check for Titanium SDK Updates* y *Check for Updates*.

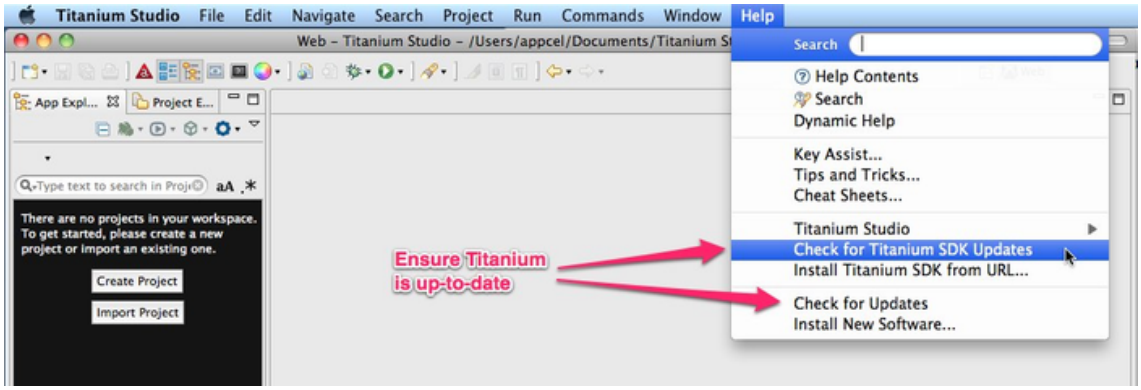


Figura 31

El primer submenú actualizará características relacionadas con la herramienta de desarrollo y el segundo con *frameworks* y utilidades que estén instaladas como complementos al entorno.

Una vez actualizada la herramienta ya será usable para comenzar a desarrollar aplicaciones.

El entorno de desarrollo de Titanium Studio necesita una configuración determinada, dependiendo del sistema operativo para el que se desee codificar aplicaciones ajenas al mismo. Cada sistema operativo tiene su propio *SDK*[6] para desarrolladores.

El computador y el sistema operativo en el que se desarrolle debe cumplir unos requisitos mínimos, así como los kits de desarrollo de los diferentes lenguajes para los que se programe. Titanium, pone a disposición de los desarrolladores a través de su documentación una matriz de compatibilidades[7] en su página web.

Operating System	Min JDK Version	Max JDK Version	Package Arch Version	Download Location
OS X	6 (aka 1.6) rev 10	6 latest revision	32bit and 64bit	Pre-installed
Windows	6 (aka 1.6) rev 10	6 latest revision	**32bit only** (x86 / i586)	Official Website
Ubuntu	6 (aka 1.6) rev 10	6 latest revision	32bit and 64bit	Canonical Archive Repository

Figura 32

4.5.2. CONFIGURACIÓN DEL ENTORNO ANDROID

Para poder desarrollar aplicaciones Android a través de Titanium Studio, habrá que indicarle al entorno dónde se ubica el SDK Android. El kit de desarrollo se encuentra disponible de forma gratuita en la página web[8] que *Google* pone a disposición de los programadores en su plataforma.

Figura 33

[6] *Software Development Kit (Kit de desarrollo de software)*

[7] *http://docs.appcelerator.com/titanium/2.0/#!/guide/Titanium_Compatibility_Matrix*

[8] *http://developer.android.com/sdk/index.html*

La descarga contiene el *SDK* propiamente dicho y el entorno Eclipse, con el único contenido necesario para desarrollar aplicaciones en Android (no incluye el *SDK* para aplicaciones Java, ni C, ni ningún otro lenguaje). Es importante tener instalado Java[9] en el sistema operativo sobre el que correrá Titanium Studio y añadir sus ejecutables al *PATH* (normalmente se configura automáticamente durante la instalación). Para comprobar la versión Java y de *JavaCompiler* con la que se trabajará se pueden ejecutar desde el terminal los comandos *java* y *javac* con el parámetro *–version*. Si Java está correctamente instalado, mostrará las versiones.

```
java -version
javac -version
```

Figura 34

Una vez iniciado, se abrirá el *Dashboard* desde dónde se configurarán los diferentes *SDK* que interesen al programador. El presento proyecto se desarrolla en Android e IOS únicamente, pero también admitiría generar aplicaciones para Blackberry OS, Tizen OS[10] y aplicaciones web.

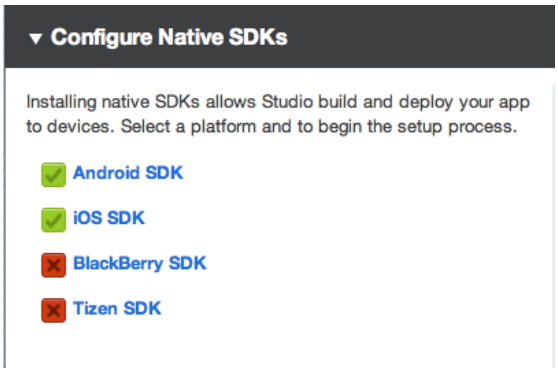


Figura 35

Cuando se pulsa en alguno de los *SDK* a configurar se aparecerá una nueva ventana para indicar la ruta en la que se ubica el kit.

[9] *http://www.java.com/es/download/*

[10] Tizen OS es un sistema operativo para terminales móviles basado en Linux y que trabaja con interfaces HTML 5.

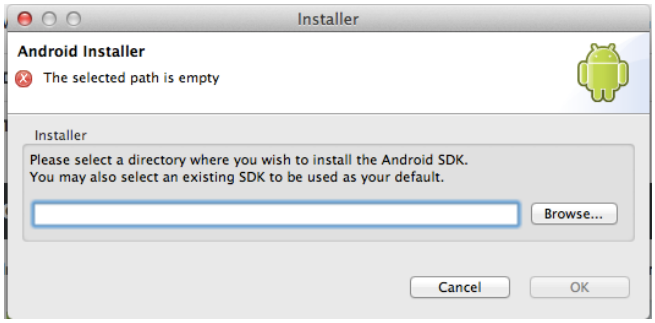


Figura 36

La herramienta reconocerá las librerías necesarias para comenzar a programar en Android y se abrirá el *Android SDK Manager*. Para trabajar con Titanium deberá instalarse el API de desarrollo 8 que coincide con la versión 2.2 de Android, con su correspondiente *SDK* y la *API de Google*.

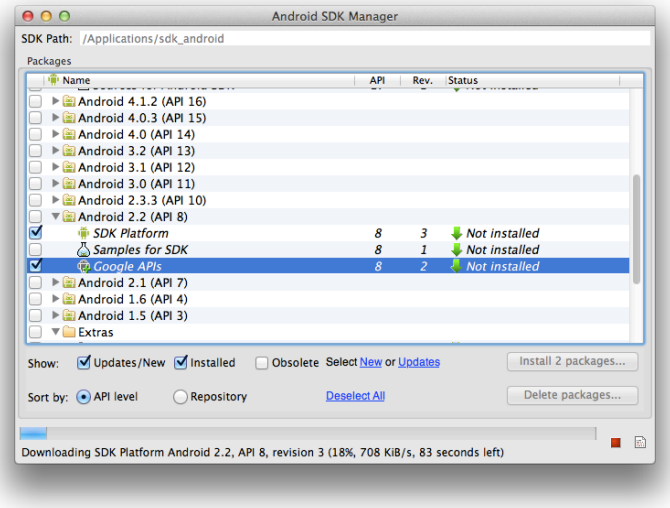


Figura 37

El *SDK* también incluye su respectivo emulador, para ejecutar las aplicaciones y simular como se comportarían en un terminal móvil real.

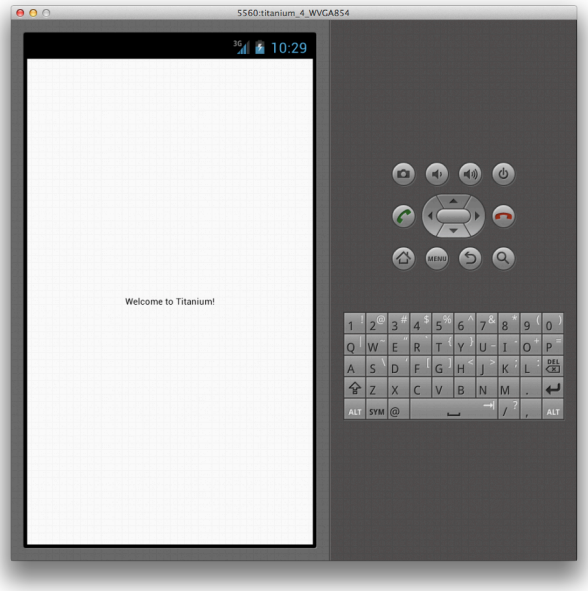


Figura 38

4.5.3. CONFIGURACIÓN DEL ENTORNO IOS

Para configurar el entorno IOS, se deberá instalar el *SDK* correspondiente (que recibe el nombre de XCode), disponible en la página web para desarrolladores[11] de Apple de forma gratuita. Para poder acceder a dicha página se deberá poseer un *ID* de Apple. Para ello habrá que rellenar un formulario de registro que también es libre de pago.

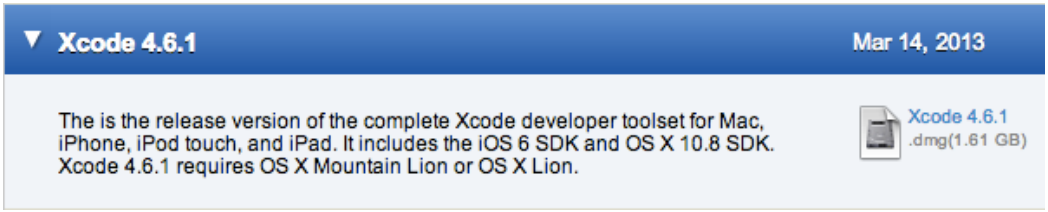


Figura 39

El XCode sólo puede instalarse en equipos que tengan instalado un sistema operativo Mac OS X. Es decir, en Titanium Studio sólo se podrán desarrollar aplicaciones para IOS si se dispone de un sistema operativo Mac OS X.

Una vez descargado, se ejecuta el instalador y se siguen los pasos básicos hasta finalizar la instalación.

[11] *https://developer.apple.com/downloads/index.action*

Cuando este instalado, este se configurará desde el terminal, mediante el comando *xcode-select* (se necesitan permisos de Adminitrador para ejecutar dicho comando), indicando la ubicación dónde se ha instalado el *XCode*.

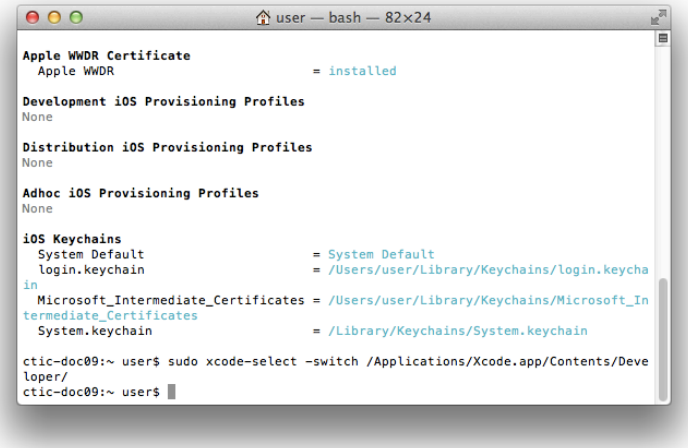


Figura 40

Tras la ejecución del comando, Titanium Studio reconocerá automáticamente el *SDK* y ya se podrá comenzar a desarrollar aplicaciones en *IOS*. El emulador también se configura de forma automática para simular el comportamiento de las aplicaciones en un terminal real. Desplegar aplicaciones en el simulador es gratuito, pero desplegarlas en un dispositivo real, al contrario que Android hay que pagar *99$* anuales por la licencia.

Figura 41

Para configurar el entorno para desplegar aplicaciones en dispositivos IOS necesitamos un conjunto de certificados.

En primer lugar, se deberá acceder a la página de desarrolladores de IOS y solicitar una petición de certificado, que deberá de ser aceptado por el Administrador de licencias de la compañía de desarrollo (que podrá ser individual y coincidirá el desarrollador con el Administrador de licencias).

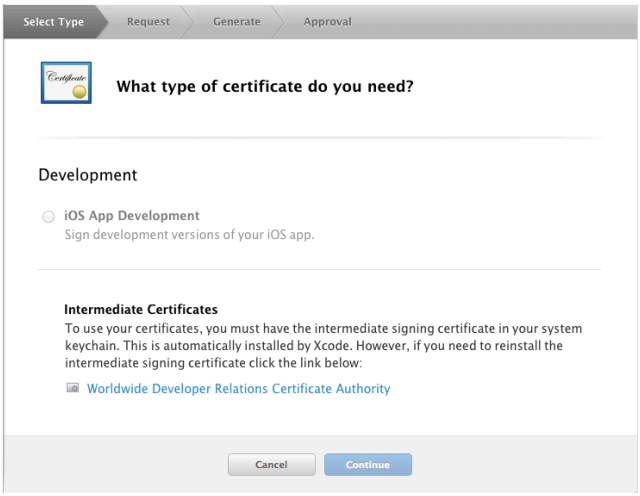


Figura 42

Para ello nos identificaremos con el *ID de Apple* correspondiente, leeremos las clausulas, y por último se pedirá un certificado del usuario y el ordenador en el que se ejecutará el despliegue de aplicaciones. Para generar el certificado necesitaremos en primer lugar una clave privada protegida mediante *RSA*[12] que se obtendrá desde el terminal mediante el comando *openssl genrsa* indicando la ruta dónde se quiere almacenar la clave generada y los bits de codificación.

```
lenovo-502:~ user$ openssl genrsa -out davidrgKey.key 2048
Generating RSA private key, 2048 bit long modulus
.....................+++
.....................................................................................
.....................+++
e is 65537 (0x10001)
lenovo-502:~ user$
```

Figura 43

Una vez generada la clave privada, se generará la clave necesaria para la petición del certificado de desarrollador a través de la primera. Para ello ejecutaremos el siguiente comando desde el terminal.

[12] *Rivest, Shamir y Adleman (RSA) es un sistema criptográfico de clave pública.*

```
lenovo-502:~ user$ openssl req -new -key davidrgKey.key -out CertificateSigningR
equest.certSigningRequest -subj "/emailAddress=aramill@gmail.com, CN=DavidRG, C=
ES"
lenovo-502:~ user$
```

Figura 44

- *–new* se indica que la clave a generar el nueva.
- *-key clavePrivada.key* se indica la ubicación de la clave privada generada previamente.
- *-out CertificadeSigningRequest.certSigningRequest* se establece el fichero de salida de la clave. Este fichero es el necesario para la petición del certificado de desarrollo de aplicaciones en IOS.
- *-subj "/emailAddress=correo@me.com, CN=FirmaUsuario, C=ES"* se indica el correo electrónico del ID de Apple del desarrollador, el nombre con el que se firmarán las aplicaciones y el país de origen.

Añadiendo el fichero de la clave generada, se podrá finalizar la petición de certificado, que el Administrador dará de alta.

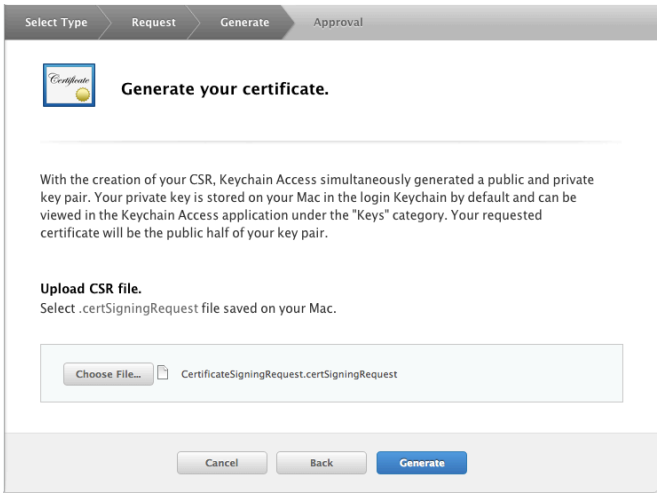


Figura 45

Una vez finalizado el proceso de petición de certificado, el usuario podrá acceder a la web de desarrolladores[13] para descargar los certificados.

[13] *https://developer.apple.com/devcenter/ios/index.action*

Figura 46

A través de la página anterior y tras identificarse, en el menú *Certificates, Identifiers & Profiles* se accederá a los certificados de la compañía desarrolladora.

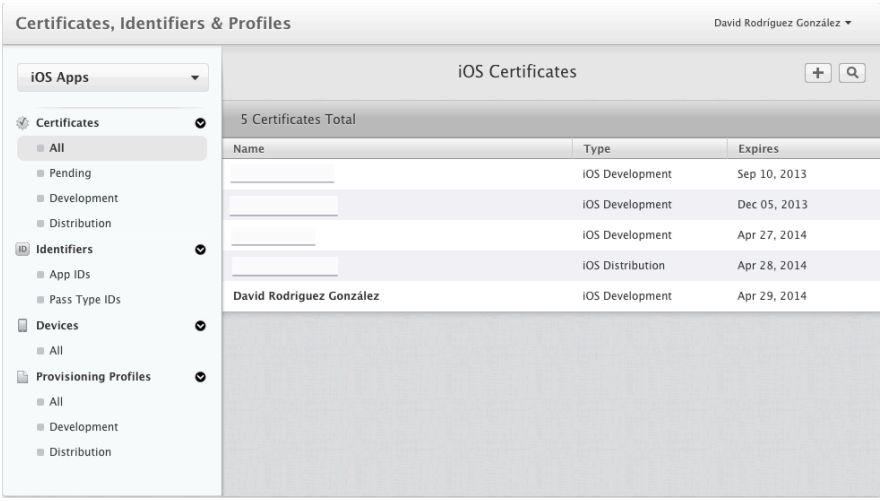


Figura 47

Para desplegar aplicaciones necesitaremos dos certificados. El particular de desarrollador dentro del submenú *Certificates* y un certificado para el dispositivo móvil en el que se van a desplegar las aplicaciones en el submenú *Provisioning Profiles*. Este último certificado es un archivo que indica que se tiene permiso para distribuir aplicaciones, que es válido para distribuirlas y que se puede utilizar con todos los dispositivos que estén registrados en la web del grupo de desarrollo (no hay límite de dispositivos a registrar).

Ambos certificados, una vez descargados, tendrán que ser importados al repositorio de *keys* del sistema operativo Mac OS X.

El repositorio de certificados se encuentra en la ruta */Aplicaciones/Utilidades*.

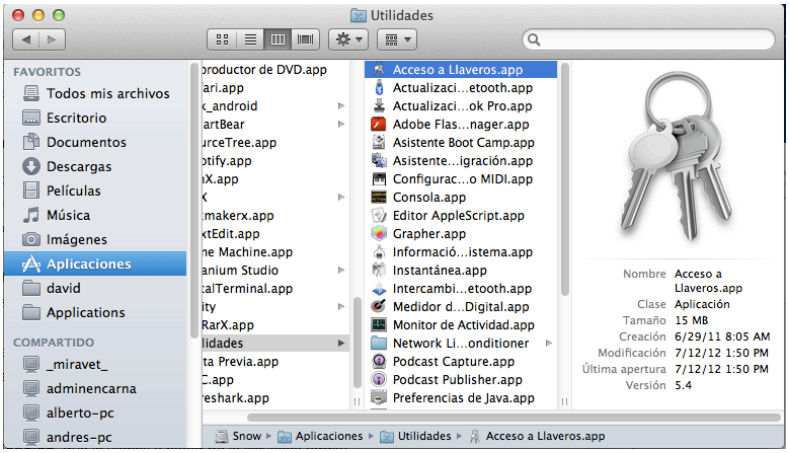


Figura 48

Dentro de la aplicación *Acceso a Llaveros* desde el menú *Archivo/Añadir Llavero* se deberán agregar los certificados descargados y el generado para la petición del certificado.

El certificado de distribución también ha de agregarse al *XCode*. Para ello basta con arrastrarlo al icono de la aplicación. También se puede agregar desde el panel principal de *XCode* pulsando en el botón *Open Others…*

Cuando todos los certificados han sido importados correctamente, Titanium Studio permitirá desplegar aplicaciones en dispositivos.

Figura 49

4.5.4. INSTALAR Y CONFIGURAR EL FRAMEWORK ALLOY

Alloy es un *framework* utilizado por *Appcelerator* para el *SDK* de Titanium Studio, que permite mejorar la calidad del código de las aplicaciones. Está basado en el patrón arquitectónico *MVC*[14].

En las versiones antiguas de Titanium Studio, tenía que ser instalado y configurado de forma manual, pero a partir de la versión 3 viene incluido en el paquete estándar de instalación. Además es recomendado en el libro de buenas prácticas de *Appcelerator* utilizar *Alloy* para la codificación de cualquier tipo de aplicación. Se espera que en las próximas versiones incluir el *framework* de forma totalmente nativa para el desarrollo de aplicaciones.

En Mac OS X habrá que instalar algunos paquetes adicionales mediante el comando *npm* con permisos de Administrador.

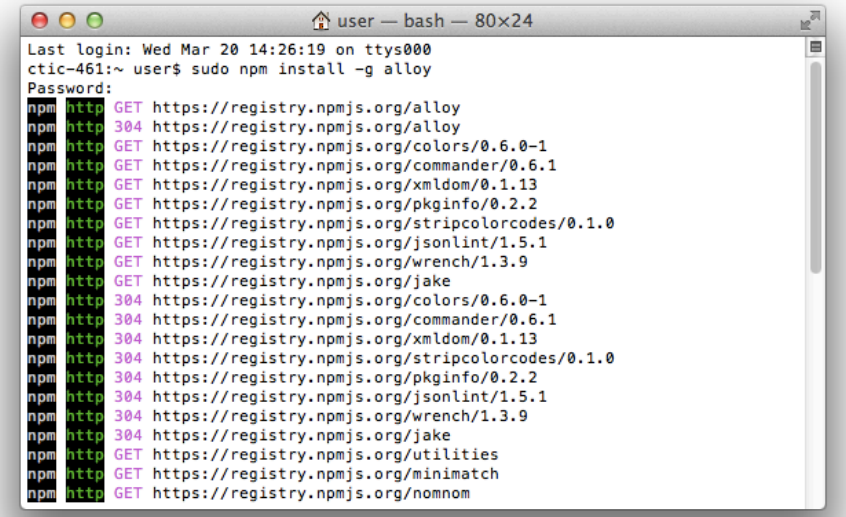


Figura 50

Una vez instalado no es necesario ningún tipo de configuración. El entorno de desarrollo se configura automáticamente durante la instalación.

4.5.5. CONFIGURAR UN TIPO DE LETRA PERSONALIZADO

En Titanium Studio, es sencillo introducir fuentes personalizadas, pero no es un proceso del todo trivial.

[14] *Modelo-Vista-Controlador*

En primer lugar se necesitará, obviamente, el fichero de la fuente o fuentes[15] que se deseen utilizar en la aplicación. Posteriormente habrá que duplicar el fichero. Necesitaremos ubicarlo en distintas rutas dentro del árbol de directorios de nuestra aplicación para que funcione tanto en Android como en IOS.

Para que se admita el tipo de fuente personalizado en Android, este deberá ser incluido en la carpeta *app/assets/android/fonts*.

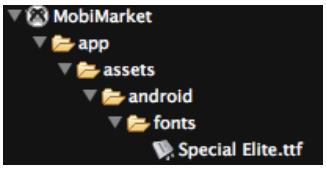


Figura 51

Es muy importante el nombre del fichero, ya que será ese (sin la extensión) el que habrá que nombrar desde los ficheros de estilo para formatear la apariencia de la fuente en la aplicación.

En IOS habrá que realizar un par de pasos adicionales. En primer lugar habrá que compilar la aplicación y *Titanium Studio* generará un fichero *info.plist*[16] en el directorio *build* dentro de la carpeta en la que se ubica la aplicación (en el *workspace*). Ese fichero *info.plist* se moverá al directorio raíz de la aplicación. De ese modo podremos editarlo y tener el fichero de configuración personalizado.

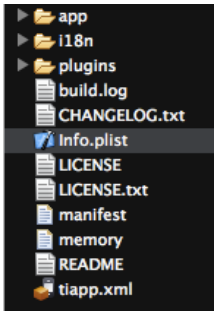


Figura 52

En el fichero *info.plist* se puede localizar el apartado *Fonts provided by application*, dónde se añadirá el nombre del fichero de la fuente, que, a igual que en Android, dejaremos en su carpeta homónima (*/app/assets/iphone/fonts*).

---

[15] *Admite ficheros .TTF y .OTF*

[16] *Es el fichero de configuración para aplicaciones en IOS y OS X generado automáticamente por XCode cada vez que se crea una nueva aplicación. Indica el nombre de aplicación, la versión, el autor, el icono y las fuentes entre muchas otras cosas.*

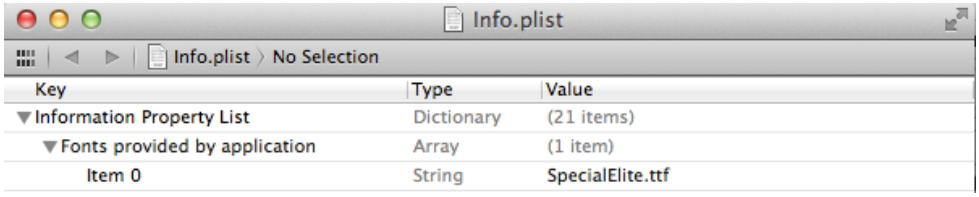


Figura 53

Para hacer una llamada a la fuente, no hay que introducir el nombre del fichero de la fuente, sino su nombre real. Para observar el nombre de la fuente se puede abrir el fichero .ttf o .otf y observar su título.

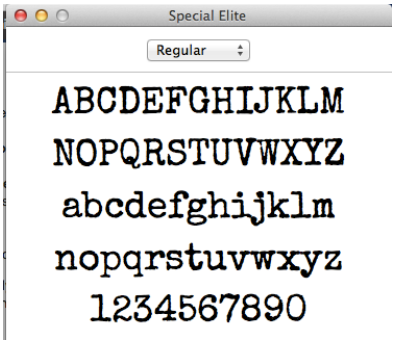


Figura 54

Por último, habrá que limpiar[17] los ficheros del proyecto y volver a compilarlo. Entonces, ya se podrán realizar llamadas a la nueva fuente personalizada, recordando que será el nombre del fichero en Android (sin la extensión) y el nombre de la fuente en sí para IOS. Por comodidad, es recomendable forzar a que sea el mismo.

```
//ESTILOS GLOBALES

"Label" : {
    left: '3%',
    font: {
        size: Alloy.Globals.TAM_FUENTE,
        fontFamily : 'Special Elite'
    },
    textAlign: 'center',
    color : 'black'
}
```

Figura 55

Desde las páginas de estilo o desde el mismo controlador, editando el atributo *font.fontFamily* se podrá formatear el tipo de letra.

[17] *Existen dos formas de limpiar los ficheros. Desde el menú Project -> Clean… o desde la propia carpeta del proyecto eliminando la carpeta generada por el compilador Build*

4.5.6. CONFIGURAR UN ANDROIDMANIFEST.XML PERSONALIZADO

El *AndroidManifest.xml* puede decirse que es el *info.plist*, mencionado en el manual 1.3. de la presente memoria, de Android. Es decir, el fichero de configuración.

Los pasos para la utilización de un *AndroidManifest.xml* personalizado es similar a la del *info.plist*. Se compilará el proyecto para que *Titanium Studio* genere el fichero *AndroidManifest.xml* de forma automática. Este fichero estará en la carpeta *build/android* en el directorio raíz del proyecto.

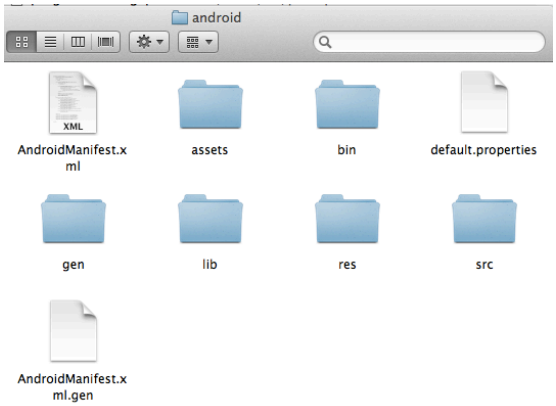


Figura 56

El fichero deberá moverse desde la carpeta */build/android* hasta el directorio */platform/android*[18]. Una vez ubicado ahí, podrá manipularse, para introducir datos de configuración adaptado a las necesidades del programador.

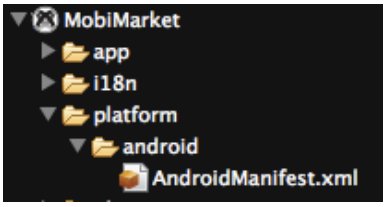


Figura 57

En el caso del presente documento, se ha necesitado un *AndroidManifest.xml* personalizado, para indicarle a la aplicación que la única posición aceptada es *PORTRAIT*[19].

[18] *Es probable que el directorio no exista. Entonces se deberá crear de forma manual en el directorio raíz de la aplicación.*

[19] *Tanto Android como IOS admiten dos posiciones dependiendo la colocación del dispositivo móvil. Portrait (vertical) o Landscape (horizontal).*

```
<activity android:name="org.appcelerator.titanium.TiActivity"
    android:configChanges="keyboardHidden|orientation"
    android:screenOrientation="portrait"/>
<activity android:name="org.appcelerator.titanium.TiTranslucentActivity"
    android:configChanges="keyboardHidden|orientation"
    android:theme="@android:style/Theme.Translucent"
    android:screenOrientation="portrait"/>
<activity android:name="org.appcelerator.titanium.TiModalActivity"
    android:configChanges="keyboardHidden|orientation"
    android:theme="@android:style/Theme.Translucent"
    android:screenOrientation="portrait"/>
<activity android:name="ti.modules.titanium.ui.TiTabActivity"
    android:configChanges="keyboardHidden|orientation"
    android:screenOrientation="portrait"/>
<activity android:name="ti.modules.titanium.ui.android.TiPreferencesActivity"
    android:screenOrientation="portrait"/>
```

Figura 58

En Titanium no se trabaja con *Activities* como en Android. Por lo tanto necesitamos que el compilador genere las Actividades pertienentes para luego poder editarlas e indicarle que la única orientación válida es *portrait*.

En IOS no hay ningún problema, ya que puede indicarse a través de su propio fichero de configuración[20] las posiciones aceptadas para toda la aplicación.

```
<iphone>
    <orientations device="iphone">
        <orientation>Ti.UI.PORTRAIT</orientation>
    </orientations>
```

Figura 59

[20] *Alloy genera el fichero tiapp.xml para la configuración de la aplicación. Sería el similar al AndroidManifest y el info.plist, pero con más limitaciones. Por ese motivo, en muchas ocasiones se necesite personalizar los ficheros de configuración nativos.*

4.6. MANUAL DE USUARIO

A continuación se describirá una manual muy sencillo con los pasos ha realizar para poder finalizar un pedido a través de la aplicación MobiMarket Móvil.

4.6.1. ACCEDER AL SISTEMA

Papara poder utilizar la aplicación es necesario estar registrado en la base de datos.

Desde la pantalla principal se puede acceder al formulario de registro o a la aplicación principal. Para acceder a la aplicación, se deberán introducir correctamente el usuario y la contraseña que se ha introducio en el momento del registro.

Se puede observar, que la aplicación tiene un botón, que permite recordar el usuario y la contraseña para que no tengan que introducirse la próxima vez que se abra la aplicación.

4.6.2. LOCALIZAR UN SUPERMERCADO

En cuanto un usuario ha sido autenticado la aplicación mostrará un mapa mostrando la zona dónde se encuentra el propio usuario y un buscador. La aplicación está formada por cuatro pestañas: la primera muestra la localización y búsqueda de supermercados, la segunda los productos disponibles en los supermercados, la tercer las promociones disponibles y la cuarta y última permite ver y modificar los productos que los usuarios han ido introduciendo en el carrito de la compra.

En el buscador permitirá acceder a los supermercados que contentan en su nombre lo que el usuario ha introducido en el campo de búsqueda. Se podrá clicar en un supermercado concreto o pulsar en la tecla de búsqueda. La diferencia reside en que si se pulsa en un supermercado concreto sólo se mostrará dicho supermercado en el mapa y si se pulsa en la tecla de búsqueda se mostrarán todos los supermercados encontrados por el filtro del nombre.

Pulsando en los supermercados que se pueden observar en el mapa se ampliará la información de dicho supermercado con datos como la dirección, el teléfono, el horario, etc.

4.6.3. AÑADIR UN PRODUCTO AL CARRITO DE LA COMPRA

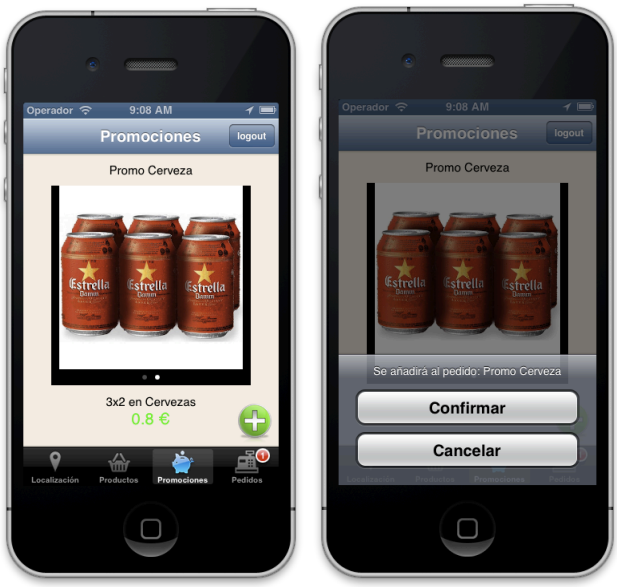
Para añadir un producto al carrito de la compra el usuario deberá de desplazarse hasta la pestaña de productos, seleccionar una categoría y luego un producto.

Una vez seleccionado un producto aparecerá un menú de opciones con la posibilidad de añadirlo directamente al carro de la compra o de ampliar la información del producto.

Una vez añadido el producto, el sistema enviará un mensaje de confirmación y se podrá obersevar el producto dentro del carrito en la cuarta pestaña.

4.6.4. AÑADIR UNA PROMOCIÓN AL CARRITO DE LA COMPRA

El proceso de añadir una promoción al carrito de la compra es muy similar al de añadir un producto. Se tendrá que acceder a la pestaña de promociones y pulsar en el botón de añadir.



Una vez pulsado el botón de añadir una promoción también aparecerá un menú de opciones para confirmar. Una vez confirmado el producto será añadido al carrito de la compra.

4.6.5. MODIFICAR UN PEDIDO

Los pedidos pueden modificarse a través de la pestaña "Pedidos". Se visualizará una lista con todos los productos añadidos al carrito.

Puede pulsarse en un ítem del pedido para ampliar la información del producto y cambiar las unidades que se deseen adquirir.

Para borrar un elemento del pedido, si se está utilizando un sistema iOS se hará pasando el dedo de forma lateral por encima del producto a eliminar y pulsando en el botón de "Delete". Si se utiliza un sistema Android, se deberá dejar pulsado al menos dos segundos sobre el producto a eliminar y confirmar el borrado. También se tiene acceso al botón "Vaciar" cuyo fin es eliminar todos los productos del carrito y dejarlo limpio.

4.6.6. REALIZAR UN PEDIDO Y VER LOS PEDIDOS ANTERIORES

Para realizar un pedido se accedera a la pestaña de "Pedidos".

Una vez en pedidos se pulsará el botón de "Pedido" y aparecerá un botón de opciones. En dicho menú se podrá confirmar el pedido y acceder a los pedidos realizados con anterioridad para comprobar su estado.

CAPÍTULO 5. CONCLUSIONES Y AMPLIACIONES

5.1. CONCLUSIONES

A continuación se mostrarán las conclusiones obtenidas en cuando a la consecución de los objetivos planteados al inicio del trabajo y las conclusiones personales del autor.

5.1.1. CONSECUCIÓN DE OBJETIVOS

Los objetivos se han cumplido prácticamente todos de forma satisfactoria.

Se ha definido un método de evaluación adaptando los atributos de calidad de la norma ISO 9126[21] a las necesidades del presente trabajo.

Se han definido unos criterios claros de evaluación basado en la respuesta a una serie de cuestiones que se plantean para tratar de evaluar los atributos de calidad anteriormente mencionados.

En cuanto al objetivo de la realización de la evaluación y la muestra de resultados no se ha quedado del todo satisfecho. Se puede decir que el objetivo se ha cumplido pero no con las expectativas que se habían generado en un primer momento. Los criterios de evaluación tienen un alto nivel de opinión del autor y valores en su mayor medida cualitativos en lugar de cuantitativos. Han surgido problemas durante la realización del proyecto que no han permitido mostrar los resultados como hubiera gustado. La falta de tiempo y de recursos (personal, tiempo y dispositivos móviles completamente rooteados principalmente) han impedido la conclusión del experimento tal y como se pretendía realizar al principio del proyecto.

Las conclusiones obtenidas a nivel de experimento han sido las esperadas en un principio. El proceso de compilado y despliegue de la herramienta es mas lento que el nativo (aunque más rápido de lo que el autor esperaba) y la velocidad de ejecución de las aplicaciones también es más lenta. A nivel de desarrollo de interfaces el problema insalvable es la carencia de layouts para su diseño.

Respecto a los objetivos del desarrollo de la aplicación se han cumplido todos. Se ha conseguido replicar aproximadamente los interfaces de la aplicación nativa de la que se partía en un momento. Algunos se han tenido que implementar de una forma un poco diferente pero llegando siempre al mismo fin. En cuanto a funcionalidad se han cumplido completamente todos los objetivos. Un cliente del supermercado puedo realizar un pedido y enviarlo a la aplicación web a través de una BBDD noSQL.

5.1.2. CONCLUSIONES PERSONALES

Las conclusiones personales obtenidas principalmente durante el desarrollo de trabajo son dos. Una haber conocido y respirado el ambiente de trabajo de un centro con cierto prestigio como es el CTIC y la segunda es el aprendizaje del uso de una herramienta y un lenguaje válido para mi uso profesional.

El autor no recomienda el uso de la herramienta para desarrollar aplicaciones comerciales complejas, pero sí para la realización de unos primeros prototipos y si una cierta idea puede implantarse entre los usuario de terminales móviles ahorrando una gran cantidad de dinero en el diseño en paralelo para múltiples plataformas.

Ha sido una experiencia mucha mas grata de lo que se esperaba en cuanto a aprendizaje y conocimiento del funcionamiento de un gran centro por dentro.

---

[21] [Introducción a la norma ISO 9126] http://normaiso9126.blogspot.com.es

5.2. AMPLIACIONES

Quedan en esta sección tres ampliaciones propuestas para la realización más completa de este proyecto:

- MEDICIÓN DE RECURSOS HARDWARE: se ha intentado realizar a lo largo de este proyecto. Sólo se ha encontrado la herramienta SysTrace incluida en el SDK de Android para la realización de la medicción pero no se ha conseguido hacer funcionar a través del .apk generado por Titanium. Sí se ha logrado lanzar en las aplicaciones nativas facilitadas para el desarrollo de la prácticas. Para el entorno iOS no se ha encontrado manera alguna de probarlo.

- MEJORA DEL INTERFAZ: aunque el objetivo de la funcionalidad se ha conseguido, los interfaces creados no son tan amigables como se esperaba en un principio. En estos momentos es muy difícil mejorarlos, pero en un futuro, cuando layouts sean incluidos en el SDK de Titanium, serán fácilmente mejorables.

- PRUEBAS CON USUARIOS: otra ampliación interesante sería la de realizar una serie de test a una muestra de población que utilice terminales móviles y que puedan expresar sus opiniones personales a través de un cuestionario.

CAPÍTULO 6. REFERENCIAS BIBLIOGRÁFICAS

6.1. LIBROS

[Manning Publications Co., 2009] Android: Guía para desarrolladores. Frank Ableson, Charlie Collins, Robi Sen. Primera Edición. Editorial Anaya.

[Prentice Hall, 2008] Clean Code: A handbook of agile software craftsmanship. Robert C. Martín

[IEEE, 1998] Especificación de Requisitos según el estándar IEEE 830

[Norma ISO 8601, 1988] Data elements and interchange formats - Information interchange - Representation of dates and times: Organización Internacional de Normalización.

6.2. REFERENCIAS EN INTERNET

No se registran todas las referencias a elementos consultados a lo largo del proyecto porque serían demasiados. Se muestran aquellos que se han utilizado con cierta asiduidad.

[Documentación de Titanium] http://docs.appcelerator.com/titanium/latest/

[Comunidad de desarrolladores en Titanium] http://www.appcelerator.com/developers/

[GeekyTheory: Tutorial] http://www.geekytheory.com/tutorial-0-introduccion-a-appcelerator/

[YouTube: Tutorial de Marco Arreguín] http://www.youtube.com/watch?v=UF3QoCezvf4

[Comunidad de desarrolladores en Android] http://developer.android.com/index.html

[Comunidad de desarrolladores en iOS] https://developer.apple.com/devcenter/ios/index.action

[JSON] http://json.org

[Stackoverflow] http://stackoverflow.com

[Documentación de JavaScript] http://www.w3schools.com/jsref/

[Introducción a la norma ISO 9126] http://normaiso9126.blogspot.com.es

[Especificación de Requisitos según el estándar de IEEE 830]
http://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

CAPÍTULO 7. APÉNDICES

7.1. GLOSARIO Y ACRÓNIMOS

- BBDD noSQL: son BBDD orientadas a documentos, que permiten un mayor escalado horizontalmente y no poseen ningún tipo de esquema fijo. Evitan el uso del producto cartesiano de SQL (JOIN). Las más famosas en la actualidad son MongoDB, MemcacheDB, Redis y CouchDB. Todas ellas son de código abierto.
- BBDD: Bases de Datos.
- Controller: elemento de codificación utilizado en Titanium. Cada controlador está compuesto por un fichero js con la lógica de negocio, un fichero xml con las vistas y un fichero .tss con el modelado de datos.
- CSS: Cascading Style Sheets. Sirve para maquear el formato de un documento. Es muy utilizado en HTML para describir las fuentes, colores, efectos, etc., de las páginas web.
- CTIC: Centro Tecnológico de la Información y Comunicación.
- CU: Caso de uso. Descripción de cada uno de los pasos que se siguen para llegar a cabo un proceso.
- ERS: Especificación de Requisitos de Software.
- IEEE: Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos). Se lee como i-e-cubo.
- INFO.PLIST: Es el fichero de configuración para aplicaciones en IOS y OS X generado automáticamente por XCode cada vez que se crea una nueva aplicación. Indica el nombre de aplicación, la versión, el autor, el icono y las fuentes entre muchas otras cosas.
- ISO: International Organization for Standardization (Organización internacional de normalización).
- IU: Interfaz de Usuario.
- JS: Fichero JavaScript.
- JSON: JavaScript Object Notation. Es un formato ligero para intercambiar datos entre aplicaciones. Es una alternativa al XML.
- MVC: Modelo-Vista-Controlador. Es un patrón arquitectónico para el desarrollo software que separa los datos y las interfaces de usuario del módulo encargado de gestionar los eventos y la comunicación entre las diferentes partes de la aplicación.
- RnF: Requisito no funcinal.
- RR: Requisito de Rendimiento.
- Tizen OS: es un sistema operativo para terminales móviles basado en Linux y que trabaja con interfaces HTML 5.
- TSS: Titanium Style Sheets. Es similar a las CSS pero están adaptadas para ser interpretadas por el compilador de Titanium Studio.
- Vista (View): Son los diferentes componentes de interfaz de usuario que se pueden utilizar para construir aplicaciones en Titanium. También reciben el mismo nombre en Android.
- Web Services: es una tecnología que utiliza un conjunto de protocolos y estándares que permiten el intercambio de información entre aplicaciones que pueden estar desarrolladas en diferentes tecnologías y lenguajes de programación.
- XML: eXtensible Markup Language. Documento utilizado para almacenar datos de forma legible.

7.2. CONTENIDO ENTREGADO EN EL CD

En este apéndice se describirá los contenidos del CD dónde se encuentra incluido este mismo documento.

DIRECTORIO	CONTENIDO
./ Directorio raíz del CD	resumen.pdf: contiene un resumen en 300 palabras del proyecto léeme.txt : fichero explicando toda esta escritura y datos del autor
./mobimarket	Contiene la estructura de directorios de la aplicación que se ha desarrollado para conseguir el objetivo del presente proyecto. También ficheros de log generados por el sistema. Además: Info.plist: fichero de configuración para apliaciones en iOS. Android.Manifest: fichero de configuración para aplicaciones en Android.
./mobimarket/app	Contiene los ficheros de código fuente repartidos en diferentes carpetas. Además de: Alloy.js: fichero de configuración inicial de la aplicación. Config.json: fichero de configuración inicial generado automáticamente por el compilador.
./mobimarket/app/assets	Contiene imágenes y tipos de fuente utilizados por la aplicación.
./mobimarket/app/controllers	Contiene el código fuente de la funcionalidad de la aplicación.
./mobimarket/app/lib	Contiene el código fuente de helpers diseñados para ser utilizados por la aplicación.
./mobimarket/app/styles	Contiene el código fuente que maneja el modelado de datos de la aplicación.
./mobimarket/app/views	Contiene el código fuente de las vistas de la aplicación.
./mobimarket/l18n	Contiene carpetas con los diferentes idiomas a utilizar por la aplicación.
./mobimarket/l18n/en	Contiene el fichero XML con los strings utilizados en inglés.
./mobimarket/l18n/es	Contiene el fichero XML con los strings utilizados en castellano.
./mobimarket/Resources	Recursos generados automáticamente por el compilador utilizados por la aplicación.
./documentacion	Directorio que contiene el presente documento en formato PDF

7.3. CÓDIGO FUENTE

En esta sección se encuentra el código fuente divido por controladores (7.1).

7.3.1. CONTROLADOR ADDBARRABUSQUEDA

FICHERO JAVASCRIPT

```
// Generamos un control sobre la view searchbar.xml
var searchbar = Alloy.createController("searchbar").getView();
//Utilizaremos esta variable como almacén de datos de supermercados para la tabla de la interfaz gráfica
var data = [];

var MAX_SUPERMERCADOS = 10;
var rangoPrimeraFila = 0;
var rangoUltimaFila = MAX_SUPERMERCADOS;

//Si la plataforma es Android tenemos que asignar la barra de búsqueda a la tabla fuera de ningún método (No encuentro la explicación pero desde dentro no funciona)
if (Ti.Platform.osname == 'android') {
	Ti.API.info("Estamos en android");
	$.tablaBusqueda.search = searchbar;
}

/**
 * Este método inserta los supermercados recibidos en un JSON parseado en la tabla de la interfaz gráfica. Para ello se genera una fila con cada uno de los supermercados.
 * Se establece un número máximo de supermercados a mostrar por tabla. Puesto que necesitamos la información de cada supermercado, se asigna a cada fila todos los datos.
 * El titulo no se puede cambiar de tamaño del titulo de una TableViewRow y el que vien por defecto es muy pequeño. Para solucionarlo se añade una Label, que si permite
 * modificarlo. Para un correcto funcionamiento de posicionado de supermercados en el mapa se utiliza el atributo 'filterAttribute' de TableView. De esta forma, le indicamos
 * que nos basamos en la búsqueda en el título de la Label, en lugar de en el título de la TableViewRow. A la TableView se lo indicamos en el controlador (fichero XML).
 * @param {String[]} supermercados
 */
function rellenaDatos(supermercados) {
	data = [];

	if (rangoUltimaFila > supermercados.length) {
		rangoUltimaFila = supermercados.length;
	}

	for (var i = rangoPrimeraFila; i < rangoUltimaFila; i++) {
		Ti.API.info("Nombre: " + supermercados[i].name + "\nDireccion: " + supermercados[i].address + "\nTelefono: " + supermercados[i].phone + "\nHoras: " + supermercados[i].bussinessHours + "\nLatitud: " +

		var supermercado = Ti.UI.createLabel({
			text : supermercados[i].name,
			width : 'auto',
			textAlign : 'left',
			left : '10dp',
			height : '50dp',
			font : {
				fontSize : Alloy.Globals.TAM_FUENTE
```

```
                        }
                });

                var row = Ti.UI.createTableViewRow({
                        id : supermercados[i]._id,
                        filter : supermercado.text,
                        supermercado : supermercados[i]
                });

                row.add(supermercado);
                Ti.App.fireEvent('addSupermercado', {
                        supermercado : supermercados[i],
                        cambioRegion : false
                });

                row.className = 'supermarket_row';

                data.push(row);
        }

        Ti.API.info("Estoy en la tabla de busqueda");
        // Añadimos las filas creadas a la tabla para que sean mostradas en la GUI.
        $.tablaBusqueda.setData(data);

        // En IOS debemos asignar la barra de búsqueda a la tabla desde dentro del método desde
fuera no funciona. Supongo que este error se modificará en versiones posteriores de Ti.
        if (Ti.Platform.osname == 'iphone') {
                Ti.API.info("Estamos en ios");
                $.tablaBusqueda.search = searchbar;
        } else if (Ti.Platform.osname == 'android') {
                //Si estamos en Android hay que forzar a la barra a perder el foco, sino
aparecerá el teclado automáticamente cada vez que se abra la pantalla de localización.
                searchbar.blur();
                searchbar.focus();
        }
};

/**
 * Recibe el evento cargaSupermercados y realiza una consulta a la BBDD para obtener la
información de todos los supermercados disponibles. La respuesta es un JSON que parseamos y
 * enviamos al método rellenarDatos para que inserte los supermercados en la GUI.
 */
Ti.App.addEventListener('cargaSupermercados', function(e) {
        var url = Alloy.Globals.SERVIDOR + "restaurants";
        var tipo = "GET";
        data = [];

        this.httpClient = Ti.Network.createHTTPClient({
                onerror : function(e) {
                        Ti.API.info("Error en cargaSupermercados: " + e.error);
                        alert("ERROR in HTTPClient");
                },
                onload : function() {
                        rellenaDatos(JSON.parse(this.responseText));
                },

                username : Alloy.Globals.USERNAME,
                password : Alloy.Globals.PASSWORD,
```

```
                timeout : 3000
        });

        this.httpClient.open(tipo, url);
        this.httpClient.send();
});

/**
 * En Android no funciona correctamente el evento de foco y tenemos que utilizar el evento
'click' para simularlo. Cuando la barra de búsqueda gana el foco hacemos desaparecer
 * el mapa para ver los supermercados a buscar.
 */
searchbar.addEventListener('click', function(e) {
        if (Ti.Platform.osname == 'android') {
                Ti.API.info("Foco en Android");
                Ti.App.fireEvent('desapareceMapa');
        }
});

/**
 * Cuando la barra de búsuqeda gana el foco hacemos desaparecer el mapa para ver los
supermercados a buscar únicamente. Este evento sólo funciona correctamente en IOS.
 */
searchbar.addEventListener('focus', function(e) {
        if (Ti.Platform.osname == 'iphone') {
                Ti.API.info("Searchbar focus");
                Ti.App.fireEvent("desapareceMapa");
        } else if (Ti.Platform.osname == 'android') {
                searchbar.focus();
        }
});

/**
 * Cuando se pulsa en la tecla buscar del teclado cuando estamos buscando supermercados,
limpiamos el mapa de las anotaciones actuales,
 * hacemos aparecer el mapa e insertamos las anotaciones que el usuario actualmente está
viendo en la tabla.
 * En caso de estar en Android tenemos que forzar a la barra de búsqueda a que pierda el foco
para que desaparezca el teclado.
 */
searchbar.addEventListener('return', function(e) {
        Ti.App.fireEvent('limpiarMapa');
        Ti.App.fireEvent("apareceMapa");

        Ti.API.info("Tamaño de table: " + data.length);
        if (searchbar.getValue() != "") {
                for (var i = 0; i < data.length; i++) {
                        var tipeado = searchbar.getValue().toLowerCase();
                        Ti.API.info("Tipeado: " + tipeado);
                        var dato = data[i].supermercado.name.toLowerCase();
                        Ti.API.info("Dato: " + dato);

                        if (dato.indexOf(tipeado) != -1) {
                                Ti.API.info("Está contenido");
                                Ti.App.fireEvent("addSupermercado", {
                                        supermercado : data[i].supermercado,
                                        cambioRegion : false
```

```
                        });
                } else {
                        Ti.API.info("No está contenido");
                }
        }
    }
    // En Android hay que forzar a la barra de búsqueda para que pierda el foco para
ocultar el teclado.
    if (Ti.Platform.osname == 'android') {
        searchbar.blur();
        searchbar.focus();
    }
});

/**
 * Esta función sólo funciona en IOS y es para el botón cancelar que aparece cuando la barra
de búsqueda gana el foco. Hace aparecer el mapa.
 */
searchbar.addEventListener('cancel', function(e) {
    Ti.API.info("SearchBar cancelar pulsado");
    Ti.App.fireEvent('apareceMapa');
});

/**
 * Este evento está a la escucha de cualquier cambio en el valor de la barra de búsqueda. Es
utilizado para realizar búsquedas en la BBDD.
 */
searchbar.addEventListener('change', function(e) {
    Ti.API.info(searchbar.getValue());
});

/**
 * Este evento permanece a la escucha de que el usuario pulse en un supermercado de los que se
muestran en un instante dado en la tabla de búsqueda.
 * Cuando se pulsa en un supermercado, se limpian las anotaciones anteriores y se añade el
supermercado pulsado al mapa.
 * En Android hay que forzar a la barra de búsqueda para que pierda el foco para ocultar el
teclado.
 */
$.tablaBusqueda.addEventListener('click', function(e) {
    Ti.App.fireEvent('limpiarMapa');
    searchbar.setValue(e.rowData.title);
    Ti.API.info("Pulsando en: " + e.rowData.supermercado.name);
    Ti.App.fireEvent('addSupermercado', {
        supermercado : e.rowData.supermercado,
        cambioRegion : true
    });
    Ti.App.fireEvent('apareceMapa');

    if (Ti.Platform.osname == 'android') {
        searchbar.blur();
    }
});
```

FICHERO TSS

```
".addBarraBusqueda": {
    backgroundColor: '#800',
    height : "40dp",
```

```
    top: 0
}
```

FICHERO XML

```
<Alloy>
	<TableView id="tablaBusqueda" filterAttribute="filter">
	</TableView>
</Alloy>
```

7.3.2. CARRITO

FICHERO JAVASCRIPT

```
/**
 * Se obtienen los productos que actualmente se encuentran en el Carrito de la compra, se crea una nueva fila para cada uno de ellos con su nombre y su id
 * y se insertan en la tabla para que pueda ser mostrado en la GUI.
 * También se calcula el valor total del carrito para que pueda ser visualizdo en su etiqueta correspondiente.
 */
var actualizar = function() {
	Ti.API.info("Estoy en actualizar carrito");
	var datos = [];
	var productos = Alloy.Globals.CARRITO.getProductos();

	Ti.App.fireEvent('actualizarBadge', {
		num : productos.length
	});

	for (var i = 0; i < productos.length; i++) {

		var iconoProducto = Ti.UI.createImageView({
			image : Alloy.Globals.SERVIDOR + productos[i].getId() + "/image",
			height : '50dp',
			width : '50dp',
			left : '1dp'
		});

		var productoLabel = Ti.UI.createLabel({
			text : productos[i].getNombre(),
			textAlign : 'left',
			left : '53dp',
			color : 'black',
			top : '7dp',
			font : {
				fontSize : Alloy.Globals.TAM_FUENTE
			}
		});

		var productoCantidadPrecioLabel = Ti.UI.createLabel({
			text : Alloy.Globals.CARRITO.getCantidad(productos[i].getId()) + " x " + productos[i].getPrecio().toFixed(2) + " €",
			color : 'gray',
			textAlign : 'right',
			right : '2dp',
			bottom : '2dp',
			font : {
```

```
                        fontSize : Alloy.Globals.TAM_FUENTE
                    }
                });

                var row = Ti.UI.createTableViewRow({
                        idProducto : productos[i].getId(),
                        hasChild : true,
                        height : '50dp'
                });

                row.add(iconoProducto);
                row.add(productoLabel);
                row.add(productoCantidadPrecioLabel);

                row.className = 'carrito_row';

                datos.push(row);
        }

        $.tv_carrito.setData(datos);
        $.lb_precioTotal.setText(Alloy.Globals.CARRITO.calculaTotal() + " " +
Alloy.Globals.DIVISA);
        Ti.App.fireEvent('actualizarBadge');
        Ti.API.info("Carrito actualizado");
};

/**
 * Se crea un diálogo de opciones y se invoca al método vaciarCarrito en caso de que se pulse
en el botón de confirmar.
 */
function doClickVaciarCarrito() {
        Ti.API.info("Click en vaciar carrito");
        var dialog = Ti.UI.createOptionDialog({
                title : L('dialog_vaciarCarritoMensaje'),
                options : [L('bt_confirmar'), L('bt_cancelar')]
        });
        dialog.addEventListener('click', function(e) {
                if (e.index == 0) {
                        vaciarCarrito();
                }
        });

        dialog.show();
}

/**
 * Se invoca al método del carrito de vaciar Carrito y se actualizan los datos de la tabla de
productos y de precio para devolverlos al estado inicial,
 * es decir, al estado de vacío.
 */
function vaciarCarrito() {
        Alloy.Globals.CARRITO.vaciaCarrito();
        $.tv_carrito.data = [];
        $.lb_precioTotal.setText("0 " + Alloy.Globals.DIVISA);
        actualizar();
};

/**
```

```
 * Función que crea un controlador sobre el componente pedidosAnteriores y abre la ventana en
la pestaña del Carrito.
 */
function pedidosAnteriores() {
	var winPedidosAnteriores = Alloy.createController('pedidosAnteriores').getView();
	Alloy.Globals.TAB_CARRITO.setWindow(winPedidosAnteriores);
	Alloy.Globals.TAB_CARRITO.open(winPedidosAnteriores);
};

/**
 * Función que muestra un OptionDialog con tres opciones, confirmar el pedido, mostrar los
pedidos anteriores o cancelar el dialog.
 * Esta función será llamada normalmente desde el botón de Pedido.
 */
function doClickPedido() {
	var dialog = Ti.UI.createOptionDialog({
		title : L('dialog_confirmarPedidoMensaje') + " " +
Alloy.Globals.CARRITO.calculaTotal() + Alloy.Globals.DIVISA,
		options : [L('bt_confirmar'), L('bt_pedidosRealizados'), L('bt_cancelar')]
	});
	dialog.addEventListener('click', function(e) {
		if (e.index == 0) {
			if (Alloy.Globals.CARRITO.estaVacio()) {
				alert(L('carritoVacio'));
			} else {
				confirmarPedido();
			}
		} else if (e.index == 1) {
			pedidosAnteriores();
		}
	});

	dialog.show();
};

/**
 * Se calcula el precio total del pedido y la fecha en la que se realizó. Se realiza una
consulta del número de pedidos que están activos y se crea un nuevo
 * pedido con los productos añadidos al carrito. El pedido se convierte a un JSON reconcocible
por la BBDD y se envía. Por último se vacía el carrito.
 */
function confirmarPedido() {
	var precioTotal = Alloy.Globals.CARRITO.calculaTotal();
	var date = new Date();
	var fechaPedido = date.getFullYear() + "-" + ("0" + (date.getMonth() + 1)).slice(-2) +
"-" + ("0" + date.getDate()).slice(-2) + "T" + date.getHours() + ":" + date.getMinutes() + ":"
+ date.getSeconds()
	var productos = Alloy.Globals.CARRITO.getProductos();
	var cantidades = Alloy.Globals.CARRITO.getCantidades();

	var json = "{" + "\"type\" : \"purchase\", " + "\"purchaseItems\" : [";

	for (var i = 0; i < Alloy.Globals.CARRITO.getNumProductos(); i++) {

		json += "{ \"item\" : {" + "\"itemType\" : \"" + productos[i].getItemType() +
"\", " + "\"type\" : \"" + productos[i].getType() + "\", " + "\"price\" : " +
productos[i].getPrecio() + ", " + "\"_id\" : \""
```

```
                if (i < Alloy.Globals.CARRITO.getNumProductos() - 1) {
                        json += ",";
                }
        }

        json += "], " + "\"purchasePrice\" : " + precioTotal + ", " + "\"buyer\" : " + "\"" +
Alloy.Globals.USERNAME + "\", " + "\"created_at\" : " + "\"" + fechaPedido + "\", " +
"\"status\" : \"in_process\""

                    "JSON: " + json);

        enviarPedidoBBDD(json);

        vaciarCarrito();

};

/**
 * Se recibe el JSON que hay qu enviar a la BBDD para añadir un nuevo pedido.
 */
function enviarPedidoBBDD(_pedidoJSON) {
        var url = Alloy.Globals.SERVIDOR + "purchases";
        var tipo = "POST";

        this.httpClient = Ti.Network.createHTTPClient({
                onerror : function(_param1) {
                        Ti.API.info("error: " + _param1.error);
                        alert(L('error_accesoBBDD'));
                },
                onload : function() {
                        Ti.API.info("Estoy en onload");
                        alert("Pedido se está procesando");
                },

                timeout : Alloy.Globals.TIEMPO_RESPUESTA,

                username : Alloy.Globals.USERNAME,
                password : Alloy.Globals.PASSWORD
        });

        this.httpClient.open(tipo, url);
        //Se envía la cabecera indicando el tipo que será un JSON con los datos del usuario a
registrar
        this.httpClient.setRequestHeader('Content-Type', 'application/json; charset=utf-8');
        //Se realiza la petición enviando como parámentros el JSON
        this.httpClient.send(_pedidoJSON);
};

/**
 * Se está a la esucha de que el usuario realice un movimiento lateral en la tabla de
productos. Este evento sólo está disponible en IOS. Cuando se escucha el evento,
 * se permite al usuario editar la fila correspondiente. Es decir, aparece en dicha fila el
botón para borrarla.
 */
$.tv_carrito.addEventListener('touchstart', function(e) {
        if (Ti.Platform.osname == 'iphone') {
                $.tv_carrito.setEditable(true);
```

```
	}
});

/**
 * Para simular el touchmove de IOS utilizamos un diálogo de opciones con la opción de borrar
el elemento. Si se pulsa en borrar se elimina manualmente del Carrito y
 * de la GUI.
 */
$.tv_carrito.addEventListener('longclick', function(e) {
	if (Ti.Platform.osname == 'android') {
		var dialog = Ti.UI.createOptionDialog({
			title : e.rowData.title,
			options : [L('bt_borrar'), L('bt_cancelar')]
		});
		dialog.addEventListener('click', function(e2) {
			if (e2.index == 0) {
				Ti.API.info("Index a borrar: " + e.index);
				$.tv_carrito.deleteRow(e.index);

				Alloy.Globals.CARRITO.quitaProducto(e.rowData.idProducto);
				actualizar();
			}
		});

		dialog.show();
	}
});

/**
 * Opción sólo disponible para IOS. Si se pulsa en el botón eliminar fila cuando esta es
editable, se elimina automáticamente de la GUI y se elimina de forma manual
 * del Carrito de la compra.
 */
$.tv_carrito.addEventListener('delete', function(e) {
	if (Ti.Platform.osname == 'iphone') {
		var producto = e.rowData;
		Alloy.Globals.CARRITO.quitaProducto(producto.idProducto);
		actualizar();
	}
});

/**
 * Envento que está a la escucha de que se pulse en algún producto de los añadidos al carrito.
Entonces se creará un controlador sobre productoCarrito, y se abrirá
 * una nueva ventana, pasándole como parámetro el id del producto en concreto sobre el que se
ha pulsado.
 */
$.tv_carrito.addEventListener('click', function(e) {
	Ti.API.info("click: " + e.rowData.idProducto);
	var winProductoCarrito = Alloy.createController('productoCarrito', {
		'idProducto' : e.rowData.idProducto
	}).getView();
	Alloy.Globals.TAB_CARRITO.setWindow(winProductoCarrito);
	Alloy.Globals.TAB_CARRITO.open(winProductoCarrito);
});

/**
 * Permanece a la esucha del evento 'actualizaCarrito' para actualizar la GUI con los
```

```
productos que se encuentran en el Carrito en ese momento dado.
 */
Ti.App.addEventListener('actualizaCarrito', actualizar);
```

FICHERO TSS

```
".container": {
	backgroundColor : Alloy.Globals.BACKGROUND_COLOR
}

"#bt_vaciarCarrito" : { bottom : '5dp',
	left : '10dp',
	height : 'auto',
	width : '90dp',
	title :L('bt_vaciar'
	

"#bt_pedido" : { bottom : '5dp',
	left : '110dp',
	height : 'auto',
	width : '90dp',
	title :
	L 'bt_pedido'
	

"#view_botones" : {
	width : Ti.UI.FILL,
	borderWidth : '2dp',
	backgroundColor : 'black'
}

"#lb_precioTotal" : {
	text : '0 €',
	width : 'auto',
	height : 'auto',
	right : '10dp',
	color : 'green',
	font : {
		fontSize : '20dp',
		fontFamily : 'Special Elite'
	}
}
```

FICHERO XML

```
<Alloy>
	<View class="container" layout="vertical" id="view_carrito">
		<TableView id="tv_carrito" fullscreen="false" height="85%"
backgroundColor="transparent"/>
		<View id="view_botones" height="15%" bottom="0dp">
			<Button id="bt_pedido" onClick="doClickPedido" />
			<Button id="bt_vaciarCarrito" onClick="doClickVaciarCarrito" />
			<Label id="lb_precioTotal" />
		</View>
	</View>
</Alloy>
```

7.3.3. ESTABLECIMIENTO

FICHERO JAVASCRIPT

```
var args = arguments[0] || {};

$.lb_nombreValue.setText(args.nombre);
$.lb_calleValue.setText(args.direccion);
$.lb_telefonoValue.setText(args.telefono);
$.lb_horarioValue.setText(args.horario);
$.iv_supermercado.setImage(Alloy.Globals.SERVIDOR + args.id + "/image");
```

FICHERO TSS

```
".container": {
	backgroundColor: "white"
}

"#win_establecimiento" : {
	title : L('establecimiento'
				'localizacion'


"Label" : {
	font : {
		fontSize : Alloy.Globals.TAM_FUENTE,
		fontFamily : 'Special Elite'
	},
	color : 'blue',
	text : "",
	left : '10dp'
}

"#lb_calle" : {
	text : L 'lb_calle'

"#lb_telefono" : {
	text : L 'lb_telefono'

"#lb_horario" : {
	text : L 'lb_horario'

"#lb_servicios" : {
	text : L 'lb_servicios'

"#lb_calleValue" : {
	left : '20dp',
	color : 'black'
}

"#lb_telefonoValue" : {
	left : '20dp',
```

```
        color : 'black'
}

"#lb_horarioValue" : {
        left : '20dp',
        color : 'black'
}

"#lb_nombreValue" : {
        font : {
                fontSize : '20dp'
        },
        color : 'black',
        textAlign : 'center'
}
```

FICHERO XML

```
<Alloy>
        <Window id="win_establecimiento" class="container" layout="vertical"
backgroundColor="#F6EEE2">
                <ScrollView layout="vertical">
                        <View height="10dp" />
                        <View layout="horizontal" width="Ti.UI.SIZE" height="100dp">
                                <ImageView id="iv_supermercado" height="100dp" width="100dp" />
                                <View width="10dp" />
                                <Label id="lb_nombreValue" textAlign="center"/>
                        </View>
                        <View height="20dp" />
                        <Label id="lb_calle" />
                        <View height="10dp" />
                        <Label id="lb_calleValue" />
                        <View height="20dp" />
                        <Label id="lb_telefono" />
                        <View height="10dp" />
                        <Label id="lb_telefonoValue" />
                        <View height="20dp" />
                        <Label id="lb_horario" />
                        <View height="10dp" />
                        <Label id="lb_horarioValue" />
                        <View height="20dp" />
                        <Label id="lb_servicios" />
                        <View height="10dp" />
                        <TableView id="tv_serviciosValue" width='200dp' height="150dp" top='0dp'
backgroundColor="#F6EEE2" />
                </ScrollView>
        </Window>
</Alloy>
```

7.3.4. INDEX

JAVASCRIPT

```
/*
 * Se globaliza la TAB de productos y TAB localizacion, ya que se necesitará desde otra
pantalla para manipularla.
 * Es la única forma de que en IOS funcione sin utilizar NavGroup. El NavGroup no puede
utilizarse, porque da problemas en
```

```
 * Android.
 */
Alloy.Globals.TAB_PRODUCTOS = $.tab_productos;
Alloy.Globals.TAB_LOCALIZACION = $.tab_localizacion;
Alloy.Globals.TAB_CARRITO = $.tab_carrito;

//Si estamos en IOS ponemos el badge de la pestaña del carrito a cero. Nos indicará el número
de elementos que tenemos metidos en el carrito.
if(Ti.Platform.osname == 'iphone') {
	$.tab_carrito.badge = 0;
}


var categoriasCargadas = false;
var promocionesCargadas = false;

//Creamos un control sobre la pantalla de Login
var winLogin = Alloy.createController('login').getView();

/**
 * Se abre la ventana de Login
 */
function mostrarLogin() {
	winLogin.open();
};

/**
 * Está a la escucha de que la ventana del carrito gane el foco. En ese momento se lanza el
evento 'actualizaCarrito'.
 * El evento focus no funciona correctamente en Android, por eso se utiliza el evento click.
 */
if (Ti.Platform.osname == 'android') {
	$.tab_carrito.addEventListener('click', function(e) {
		Ti.API.info("Click en tab_carrito");
		Ti.App.fireEvent('actualizaCarrito');
	});
} else if (Ti.Platform.osname == 'iphone') {
	$.win_carrito.addEventListener('focus', function(e) {
		Ti.API.info("Click en tab_carrito");
		Ti.App.fireEvent('actualizaCarrito');
	});
}

/**
 * Está a la escucha de que la ventana de promociones gane el foco y se lanza el evento de
'cargaPromociones'.
 * El evento focus no funciona correctamente en Android, por eso se utiliza el evento click.
 */
$.tab_promociones.addEventListener('click', function() {
	if (Ti.Platform.osname == 'android' && !promocionesCargadas) {
		Ti.App.fireEvent('cargaPromociones');
	}

});
$.win_promociones.addEventListener('focus', function() {
	if (Ti.Platform.osname == 'iphone') {
		Ti.App.fireEvent('cargaPromociones');
	}
```

```
});

/**
 * Está a la escucha de que la ventana de productos gane el foco y se lanza el evento
'cargaCategorias'.
 * El evento focus no funciona correctamente en Android, por eso se utiliza el evento click.
 */
$.tab_productos.addEventListener('click', function() {
	if (Ti.Platform.osname == 'android' && !categoriasCargadas) {
		Ti.App.fireEvent('cargaCategorias');
	}
});

$.win_productos.addEventListener('focus', function() {
	if (Ti.Platform.osname == 'iphone') {
		Ti.App.fireEvent('cargaCategorias');
	}
});

/**
 * Está a la escucha de que la ventana de localización gane el foco para lanzar el evento
'cargaSupermercados'
 */
$.win_localizacion.addEventListener('focus', function() {
	Ti.App.fireEvent('cargaSupermercados');
	Ti.App.fireEvent('cargaPosicion');
});

/**
 * Comprueba si el usuario está logeado. Si no lo está lanza la ventana de login en lugar de
la principal de la aplicación.
 */
if (!Alloy.Globals.IS_LOGIN) {
	Ti.API.info("No estas logueado");
	mostrarLogin();
} else {
	$.index.open();
}

/**
 * Está a la esucha del evento 'loginOk' que se lanzará desde la ventana de Login. Cuando lo
reciba actualiza los datos de usuario y le indica
 * a la aplicación que el Login se ha realizado de forma correcta. Entonces se cierra la
ventana de Login y se abre la principal de la aplicación.
 */
Ti.App.addEventListener('loginOk', function(data) {
	Ti.API.info("USER: " + data.user);
	Ti.API.info("PASS: " + data.pass);

	Alloy.Globals.USERNAME = data.user;
	Alloy.Globals.PASSWORD = data.pass;

	Alloy.Globals.IS_LOGIN = true;

	winLogin.close();
	$.index.open();
});
```

```
/**
 * Evento que permanece a la escucha de que las categorías se carguen. Una vez cargadas cambia
el atributo categoriasCargadas a true. De esta forma
 * se evitará que se vuelvan a cargar.
 */
Ti.App.addEventListener('cargaCategoriasOk', function() {
	categoriasCargadas = true;
});

/**
 * Evento que permanece a la escucha de que las promociones se carguen. Una vez cargadas
cambia el atributo promocionesCargadas a true. De esta forma
 * se evitará que se vuelvan a cargar.
 */
Ti.App.addEventListener('promocionesCargadasOK', function() {
	promocionesCargadas = true;
});

/**
 * Evento que permanece a la escucha del evento personalizado 'actualizarBadge' y cambia el
valor del Badge de Carrito al número de productos que hay
 * metidos en el carrito en el instante de la actualizacion.
 */
Ti.App.addEventListener('actualizarBadge', function() {
	if (Ti.Platform.osname == 'iphone') {
		$.tab_carrito.setBadge(Alloy.Globals.CARRITO.getNumProductos());
	}
});
```

FICHERO TSS

```
"#index": {
    backgroundColor:'#F6EEE2',
    fullscreen: false,
    exitOnClose: true
}

//VENTANAS
"#win_localizacion" : {
	title : L('localizacion'

"#win_productos" : {
	title : L 'categorias'
                        "#F6EEE2"
}
"#win_promociones" : {
	title : L 'promociones'
                        "#F6EEE2"
}
"#win_carrito" : {
	title : L 'carrito'
                        "#F6EEE2"
}

//LABELS
"#label2" : {
	text : L 'productos'
```

```
"#label1" : {
	text : L 'localizacion'
"#label3" : {
	text : L 'promociones'
"#label4" : {
	text : L 'carrito'

//TABS
"Tab" : {
	font : {
		fontWeight : 'bold'
	}
}
"#tab_localizacion" : {
	title : L 'localizacion'
		'/images/location.png'
}
"#tab_productos" : {
	title : L 'productos'
		'/images/products.png'
}
"#tab_promociones" : {
	title : L 'promociones'
		'/images/promo.png'
}
"#tab_carrito" : {
	title : L 'carrito'
		'/images/carrito.png'
}
//ACTIVITY_INDICATOR
"#activityIndicator": {
    style: Ti.UI.ActivityIndicatorStyle.DARK
}
"#activityIndicator[platform=ios]": {
    style: Ti.UI.iPhone.ActivityIndicatorStyle.DARK
}
```

FICHERO XML

```
<Alloy>
	<TabGroup>
		<Tab id="tab_localizacion">
			<Window class="container" id="win_localizacion" layout="vertical">
				<Require src="localizacion" id="localizacion" />
			</Window>
		</Tab>
		<Tab id="tab_productos">
			<Window class="container" id="win_productos">
				<Require src="productos" id="listaProductos" />
```

```
                </Window>
        </Tab>
        <Tab id ="tab_promociones">
                <Window class="container" id="win_promociones">
                        <Require src="promociones" id="promociones" />
                </Window>
        </Tab>
        <Tab id="tab_carrito">
                <Window class="container" id="win_carrito">
                        <Require src="carrito" id="carrito" />
                </Window>
        </Tab>
    </TabGroup>
</Alloy>
```

7.3.5. LOCALIZATION

FICHERO JAVASCRIPT

```
//Está compuesto de dos partes. La barra de búsqueda y el mapa. No se realiza ninguna
operación lógica.
```

FICHERO TSS

```
".container": {
        backgroundColor: "white"
}
```

FICHERO XML

```
<Alloy>
        <View class="container" scrollable="false">
                <Require src="addBarraBusqueda" id="barraBusqueda" />
                <Require src="mapa" id="mapa" />
        </View>
</Alloy>
```

7.3.6. LOGIN

FICHERO JAVASCRIPT

```
//El status correcto para la validación de user y pass
var STATUS_OK = 200;

var winRegistro = Alloy.createController('registro').getView();
if (Ti.App.Properties.getString('login_user') != null &&
Ti.App.Properties.getString('login_pass') != null &&
Ti.App.Properties.getBool('login_credenciales') != null) {
        $.tf_user.setValue(Ti.App.Properties.getString('login_user'));
        $.tf_pass.setValue(Ti.App.Properties.getString('login_pass'));
        $.switch_credenciales.setValue(Ti.App.Properties.getBool('login_credenciales'));
}

/**
 * Se obtienen desde los TextField de la GUI el usuario y el password introducido por el
usuario. Se realiza una consulta a la BBDD con las pertinenetes cabeceras,
 * y se analiza la resuestas. En caso de obtener un status correcto se lanza el evento de que
el login ha sido correcto. En caso contrario se lanza una alerta de error.
```

```
 * @param {Event} e
 */
function doClick(e) {
	Ti.API.info("Boton enviar pulsado");
	var user = $.tf_user.getValue();
	var pass = $.tf_pass.getValue();
	Ti.API.info("User: " + user + " Pass: " + pass);
	var userValue = encodeURIComponent(user);
	var passValue = encodeURIComponent(pass);

	//La consulta se realiza mediante un POST y los parámetros tienen este formato
específico para que el Servidor lo reconozca.
	var parametros = "name=" + userValue + "&password=" + passValue;

	var url = Alloy.Globals.SERVIDOR + "login";
	var tipo = "POST";

	this.httpClient = Ti.Network.createHTTPClient({
		onerror : function(e) {
			Ti.API.info("Error en login: " + e.error);
			alert(L('error_loginFail'));
		},
		onload : function() {
			Ti.API.info("Estoy en onload");

			if (this.status == STATUS_OK) {
				alert("Login OK");
				if ($.switch_credenciales.value) {
					Titanium.App.Properties.setString('login_user',
userValue);
					Titanium.App.Properties.setString('login_pass',
passValue);
					Titanium.App.Properties.setBool('login_credenciales',
true);
				} else {
					Titanium.App.Properties.setString('login_user', "");
					Titanium.App.Properties.setString('login_pass', "");
					Titanium.App.Properties.setBool('login_credenciales',
false);
				}
				Ti.App.fireEvent('loginOk', {
					user : userValue,
					pass : passValue
				});
			} else {
				alert(L('error_loginFail'));
			}
		},

		timeout : Alloy.Globals.TIEMPO_RESPUESTA
	});

	this.httpClient.open(tipo, url);
	//Se envían con la petición la cabecera con el tipo de datos correcto para realizar la
validación de user y contraseña.
	this.httpClient.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
	this.httpClient.send(parametros);
};
```

```
/**
 * Se obtiene el control sobre la ventana de registro y se abre.
 */
function openRegistrarUser() {
	winRegistro.open();
};

Ti.App.addEventListener('registroOk', function(e) {
	winRegistro.close();
	alert(L('alert_usuarioRegistrado'));
});
```

FICHERO TSS

```
".container": {
	backgroundColor: "white"
}

"#lb_user" : {
	title : L('usuario'
	

"#lb_credenciales" : {
	text : L 'lb_credenciales'
	

"#switch_credenciales[platform=android]" : {
	titleOn : L 'si'
	                'no'
	

"#tf_user" : {
	autocapitalization : Titanium.UI.TEXT_AUTOCAPITALIZATION_NONE
}

"#tf_pass" : {
	passwordMask: true
}

"Label" : {
	font : {
		fontSize : Alloy.Globals.TAM_FUENTE,
		fontFamily : 'Special Elite'
	}
}

//TEXTFIELD
"TextField" : {
	borderStyle : "TI.UI.INPUT_BORDERSTYLE_BEZEL",
	borderWidth : '1dp'
}

"Button" : {
	color : 'gray'
}
```

FICHERO XML

```
<Alloy>
	<Window id="win_login" class="container" layout="vertical" backgroundColor="#F6EEE2">
		<ScrollView contentHeight="auto" contentWidth="auto" height="400dp">
			<View id="sv_login" top="10dp" height="400dp" layout="vertical">
				<View height="150dp" width="300dp">
					<ImageView
image="/common/images/logoicon.png"></ImageView>
				</View>
				<View height="50dp" />
				<View height="50dp">
					<Label id="lb_user" left="5dp" width="100dp"
text="USUARIO" textAlign="center" />
					<TextField id="tf_user" right="5dp" width="200dp"
height="40dp" backgroundColor="white" value="" />
				</View>
				<View height="50dp">
					<Label id="lb_pass" left="5dp" width="100dp" text="PASS"
textAlign="center" />
					<TextField id="tf_pass" right="5dp" width="200dp"
height="40dp" backgroundColor="white" value="" />
				</View>
				<View height="50dp">
					<Label id="lb_credenciales" left="5dp" width="200dp"
textAlign="left" />
					<Switch id="switch_credenciales" value="false"
right="5dp"/>
				</View>
			</View>
		</ScrollView>
		<View id="view_loginButtons" height="65dp">
			<Button id="bt_login" title="LOGIN" bottom="10dp" left="50dp"
width="100dp" onClick="doClick"/>
			<Button id="bt_signUp" title="SIGN UP" bottom="10dp" right="50dp"
width="100dp" onClick="openRegistrarUser" />
		</View>
	</Window>
</Alloy>
```

7.3.7. MAPA

FICHERO JAVASCRIPT

```
Ti.Geolocation.purpose = L('mapPurpose');

//Si estamos en Android ocultamos los botones de ampliar y reducir zoom en el mapa, ya que
GoogleMaps los incluye de serie.
if (Ti.Platform.osname == 'android') {
	$.view_botones.setVisible(false);
}

/**
 * Se recibe un supermercado en JSON parseado, se crea una anotación y se añade al mapa.
También un booleano que le indica si tiene que cambiar
 * la región que se muestra a la del nuevo supermercado añadido.
 * @param {String[]} _supermercado
 * @param {bool} _cambioRegion
 */
```

```
var addSupermercadoEnMapa = function(_supermercado, _cambioRegion) {
	Ti.API.info("addSupermercadoEnMapa");
	Ti.API.info("Nombre recibido en mapa: " + _supermercado.name);

	var annotation = Ti.Map.createAnnotation({
		latitude : _supermercado.coordinate.latitude,
		longitude : _supermercado.coordinate.longitude,
		id : _supermercado._id,
		title : _supermercado.name,
		direccion : _supermercado.address,
		horario : _supermercado.bussinessHours,
		telefono : _supermercado.phone,
		servicios : _supermercado.services,
		pincolor : Ti.Map.ANNOTATION_GREEN,
		animate : true,
		myid : $.mapa.annotations.length,
		rightButton : '/common/images/annotation_button.png',
		image : '/common/images/annotation.png'
	});

	$.mapa.addAnnotation(annotation);
	if (_cambioRegion) {
		cambiarRegion(_supermercado.coordinate.latitude,
_supermercado.coordinate.longitude);
	}

};

/**
 * Se aleja el mapa una unidad de medida.
 * @param {Event} e
 */
function zoomIn(e) {
	$.mapa.zoom(1);
};

/**
 * Se acerca el mapa una unidad de medida
 * @param {Event} e
 */
function zoomOut(e) {
	$.mapa.zoom(-1);
};

/**
 * Desplaza la visualización de región del mapa hasta la posición en la que se encuentra el
usuario según el receptor de GPS.
 * @param {Event} e
 */
function buscarmeEnMapa(e) {
	alert("Aquí estoy");
	posicionActual();
}

/**
 * Está a la escucha del evento para añadir un supermercado al mapa.
 */
Ti.App.addEventListener('addSupermercado', function(data) {
```

```
	addSupermercadoEnMapa(data.supermercado, data.cambioRegion);
});

/**
 * Está a la escucha del evento 'desapareceMapa' para hacer desaparecer el interfaz el mapa y
los botones de zoom.
 */
Ti.App.addEventListener('desapareceMapa', function(e) {
	$.mapa.setVisible(false);
	$.view_botones.setVisible(false);
});

/**
 * Está a la escucha del evento 'apareceMapa' para hacer aparcer en el interfaz el mapa y los
botones de zoom. Estos últimos sólo en SO IOS, ya que en GoogleMaps
 * incluye dichos botones por defecto.
 */
Ti.App.addEventListener('apareceMapa', function(e) {
	if (Ti.Platform.osname == "iphone") {
		$.view_botones.setVisible(true);
	}

	$.mapa.setVisible(true);

});

/**
 * Está a la esucha del evento 'limpiarMapa' para eliminar todas las anotaciones que se
encuentren actualmente en el mapa.
 */
Ti.App.addEventListener('limpiarMapa', function(e) {
	Ti.API.info("Limpiando mapa");
	$.mapa.removeAllAnnotations();
});

/**
 * Obtiene la latitud y longitud de dónde se encuentra actualmente el dispositivo móvil a
través de una librería de Titanium y cambia la región
 * actual que se muestra en el mapa a la posición actual.
 */
function posicionActual() {
	Ti.API.info("Posicion Actual");
	Titanium.Geolocation.getCurrentPosition(function(e) {
		if (e.error) {
			Ti.API.info("Error: " + JSON.stringify(e.error));
			return;
		}

		cambiarRegion(e.coords.latitude, e.coords.longitude);
	});
};

/**
 * Cambia la región que se muestra en el mapa dejándo en el centro la latitud y longitud
recibidas como parámetro.
 * @param {Object} _latitud
 * @param {Object} _longitud
 */
```

```
function cambiarRegion(_latitud, _longitud) {
	if (_latitud == null || _longitud == null || isNaN(_latitud) || isNaN(_longitud)) {
		Ti.API.info("Error: Región incorrecta");
	} else {
		var region = {
			latitude : _latitud,
			longitude : _longitud,
			animate : true,
			latitudeDelta : 0.1,
			longitudeDelta : 0.1
		}

		$.mapa.setLocation(region);
	}
};

/**
 * Evento que permanece a la espera que se pulse el botón derecho de una Annotation. En el momento que se puse se inicia un
 * control sobre la ventana con los detalles del establecimiento, pasándole estos como argumento. Posteriormente se abre la ventana.
 */
$.mapa.addEventListener('click', function(e) {
	if (e.clicksource == 'rightButton' || e.clicksource == 'rightPane') {
		Ti.API.info("click");
		var winEstablecimiento = Alloy.createController('establecimiento', {
			'nombre' : e.annotation.title,
			'telefono' : e.annotation.telefono,
			'direccion' : e.annotation.direccion,
			'servicios' : e.annotation.servicios,
			'horario' : e.annotation.horario,
			'id' : e.annotation.id
		}).getView();
		Alloy.Globals.TAB_LOCALIZACION.setWindow(winEstablecimiento);
		Alloy.Globals.TAB_LOCALIZACION.open(winEstablecimiento);
	}
});

/**
 * Evento que permanece a la espera de que se lance el evento 'cargaPosicion'. En dicho caso se llama al método posicionActual,
 * que sitúa al mapa en la posición que le envía el GPS de dónde se encuentra el dispositivo móvil actualmente.
 */
Ti.App.addEventListener('cargaPosicion', posicionActual);
```

FICHERO TSS

```
"#mapa": {
    top: '11%',
    animate: true,
    regionFit: true,
    userLocation: true,
    region: {
        latitude: Alloy.Globals.LATITUDE_BASE,
        longitude: Alloy.Globals.LONGITUDE_BASE,
        latitudeDelta: 0.1,
```

```
        longitudeDelta: 0.1
    }
}
/*
"#bt_zoom_in" : {
        top : '5dp',
        right : '5dp',
        width : '30dp',
        height : '30dp'
},
"#bt_zoom_out" : {
        top : '37dp',
        right : '5dp',
        width : '30dp',
        height : '30dp'
},
*/
```

FICHERO XML

```
<Alloy>
        <View ns="Ti.Map" id="mapa" mapType="Ti.Map.STANDARD_TYPE" scrollable="false">
        </View>
        <View id="view_botones" width="12%" height="25%" top="3%" left="87%" top="50dp"
scrollable="false">
                        <Button id="bt_zoom_in" onClick="zoomIn" title="+" width="95%"
height="33%" top="0%"/>
                        <Button id="bt_zoom_out" onClick="zoomOut" title="-" width="95%"
height="33%" top="34%"/>
                        <Button id="bt_miLocalizacion" onClick="buscarmeEnMapa" title="Yo"
width="95%" height="33%" top="68%">
        </View>
</Alloy>
```

7.3.8. PEDIDOS ANTERIORES

FICHERO JAVASCRIPT

```
/**
 * Funcioón que realiza una consulta a la BBDD y devuelve un JSON con todos los pedidos
realizados
 */
function getPedidosAnteriores() {
        Ti.API.info("Recibo evento carga pedidos anteriores");
        var tipo = "GET";
        var url = Alloy.Globals.SERVIDOR + "purchases";
        var getAllPedidosAnteriores = Ti.Network.createHTTPClient({
                username : "mobimarket",
                password : "mobimarket",
                timeout : 3000,
                onerror : function(e1) {
                        Ti.API.info("Error en getPedidosAnteriores: " + e1.error);
                        alert("Error: " + e1.error);
                },
                onload : function(e2) {
                        cargaPedidosAnteriores(JSON.parse(this.responseText));
                }
        });
```

```
	//Se prepara la conexión con el tipo (GET, PUT o POST) y la URL
	getAllPedidosAnteriores.open(tipo, url);
	//Se lanza la petición
	getAllPedidosAnteriores.send();

};

/**
 * Recibe como parámetro un JSON parseado con todos los pedidos de la BBDD. Busca
coincidencias con el usuario que está logeado.
 * Crea una fila por cada pedido del usuario y la muestra.
 * @param {String[]} _pedidosJSON
 */
function cargaPedidosAnteriores(_pedidosJSON) {
	var rows = [];
	for(var i=0; i < _pedidosJSON.length; i++) {
		if(_pedidosJSON[i].buyer == Alloy.Globals.USERNAME) {
			Ti.API.info("Mostrando pedido : " + i);
			var row = Ti.UI.createTableViewRow({
				height : '50dp'
			});

			var fechaFormateada = _pedidosJSON[i].created_at.split("T");
			var fecha = fechaFormateada[0];
			var hora = (fechaFormateada[1].split("."))[0];
			var labelFecha = Ti.UI.createLabel({
				left : '2dp',
				top : '2dp',
				text : fecha,
				font : {
					fontSize : Alloy.Globals.TAM_FUENTE
				},
				color : 'black'
			});
			var labelHora = Ti.UI.createLabel({
				left : '2dp',
				bottom : '2dp',
				text : hora,
				font : {
					fontSize : Alloy.Globals.TAM_FUENTE
				},
				color : 'black'
			});

			//El atributo backgroundImage a none hace que funcione correctamente el
atributo backgroundColor. Es un bug de Titanium
			var botonEstado = Ti.UI.createButton({
				left : '100dp',
				height : '40dp',
				width : '130dp',
				enabled : false,
				backgroundImage : 'none'
			});
			var estado = _pedidosJSON[i].status;
			if(estado == 'in_process') {
				botonEstado.setTitle(L('en_proceso'));
				botonEstado.setColor('black');
				botonEstado.setBackgroundColor('#DF3D5B');
```

```
                } else if(estado == 'served') {
                        botonEstado.setTitle(L('servido'));
                        botonEstado.setColor('white');
                        botonEstado.setBackgroundColor('#3DDF55');
                        row.setBackgroundColor("#A6D7AE");
                }

                var precio = _pedidosJSON[i].purchasePrice + "€";
                var labelPrecio = Ti.UI.createLabel({
                        text : precio,
                        right : '2dp',
                        font : {
                                fontSize : Alloy.Globals.TAM_FUENTE
                        },
                        color : 'black'
                });

                row.add(labelFecha);
                row.add(labelHora);
                row.add(botonEstado);
                row.add(labelPrecio);

                rows.push(row);
        }
    }

    $.tv_pedidosAnteriores.setData(rows);
}

getPedidosAnteriores();
```

FICHERO TSS

```
".container": {
        backgroundColor: "white"
}

"#win_pedidosAnteriores" : {
        title : L('bt_pedidosRealizados'

"#tv_pedidosAnteriores" : {
        borderRadius : '5dp'
}
```

FICHERO XML

```
<Alloy>
        <Window class="container" id="win_pedidosAnteriores">
                <TableView id="tv_pedidosAnteriores" />
        </Window>
</Alloy>
```

7.3.9. PRODUCTOCARRITO

FICHERO JAVASCRIPT

```
//Se obitnen los argumentos recibidos como parámetro.
var args = arguments[0] || {};
//Se utiliza una variable producto, para almacenar el producto del pedido del que se mostrarán
los detalles.
var producto = Alloy.Globals.CARRITO.getProducto(args.idProducto);

//Se rellenan las etiquetas y la imagen correspondiente con los datos del producto.
$.iv_producto.setImage(Alloy.Globals.SERVIDOR + producto.getId() + "/image");
$.lb_nombreProducto.setText(producto.getNombre());
$.lb_descripcionProducto.setText(producto.getDescripcion());
$.tf_cantidad.setValue(Alloy.Globals.CARRITO.getCantidad(producto.getId()).toString());
$.lb_precioPorUnidadValue.setText(producto.getPrecio().toFixed(2) + " €");

/**
 * Evento que escucha pulsaciones en el botón +. Cuando se pulsa, se añade una unidad más al
carrito y se actualizan las etiquetas
 * de número de unidades y precio total del producto.
 */
$.bt_plus.addEventListener('click', function(e) {
	var unidades = Alloy.Globals.CARRITO.getCantidad(producto.getId());
	Alloy.Globals.CARRITO.modificarCantidad(producto.getId(), unidades + 1);
	$.tf_cantidad.setValue(Alloy.Globals.CARRITO.getCantidad(producto.getId()).toString());
	var precioTotal = producto.getPrecio() *
Alloy.Globals.CARRITO.getCantidad(producto.getId());
	$.lb_precioTotalValue.setText(precioTotal.toFixed(2) + " €");
});

/**
 * Evento que permanece a la escucha de pulsaciones en el botón -. cuando se pulsa el botón se
comprueba que no sea la última unidad que
 * está incluída en el carrito. En ese caso se pregunta al usuario mediante un OptionDialog si
quiere eliminar el producto del carrito. Si el producto
 * es eliminado se cierra la ventana de detalles del producto y se vuelve a la ventana de
carrito. Si existe más de una unidad, entonces se
 * modifica la cantidad del carrito y se actualizan las etiquetas de número de productos y de
precio total del pedido.
 */
$.bt_menos.addEventListener('click', function(e) {
	Ti.API.info("--------------- click en boton menos ------------");
	var unidades = Alloy.Globals.CARRITO.getCantidad(producto.getId());
	if (unidades > 1) {
		Alloy.Globals.CARRITO.modificarCantidad(producto.getId(), unidades - 1);

	$.tf_cantidad.setValue(Alloy.Globals.CARRITO.getCantidad(producto.getId()).toString());
		$.lb_precioTotalValue.setText(producto.getPrecio() *
Alloy.Globals.CARRITO.getCantidad(producto.getId()).toFixed(2) + " €");
	} else {
		var dialog = Ti.UI.createOptionDialog({
			title : producto.getNombre(),
			options : [L('bt_borrar'), L('bt_cancelar')]
		});
		dialog.addEventListener('click', function(e2) {
			if (e2.index == 0) {
				Alloy.Globals.CARRITO.quitaProducto(producto.getId());
				$.win_productoCarrito.close();
```

```
                        }
                });
                dialog.show();

        $.tf_cantidad.setValue(Alloy.Globals.CARRITO.getCantidad(producto.getId()).toString());
        }
});

/**
 * Método que actualiza la etiqueta de precio total del pedido. Para ello se obtiene el precio
del producto y el número
 * de unidades que el carrito posee de dicho producto.
 */
function actualizarPrecios() {
        var cantidad = Alloy.Globals.CARRITO.getCantidad(producto.getId());
        var precio = producto.getPrecio();
        var total = cantidad * precio;

        $.lb_precioTotalValue.setText(total.toFixed(2) + " €");
}

/**
 * Evento que permanece a la escucha de que se abra la ventana de detalles del producto.
Interesa que cada vez que se abra se actualice la etiqueta de precio total.
 */
$.win_productoCarrito.addEventListener('open', actualizarPrecios);

/**
 * Evento que permanece a la escucha de que la ventana de detalles del producto se cierre.
Interesa que cuando se cierre la ventana y se vuelva a la del
 * carrito, este esté actualizado.
 */
$.win_productoCarrito.addEventListener('close', function(e) {
        Ti.App.fireEvent('actualizaCarrito');
});
```

FICHERO TSS

```
".container": {
        backgroundColor: "#F6EEE2"
}

"#win_productoCarrito" : {
        title : L('detallesProducto'
}

"Label" : {
        color : 'black',
        font : {
                fontSize : Alloy.Globals.TAM_FUENTE,
                fontFamily : 'Special Elite'
        }
}

"#tf_cantidad" : {
        font : {
                fontSize : '25dp'
        }
```

```
}

"#lb_descripcionProducto" : {
	font : {
		fontFamily : 'Georgia-Italic',
		fontSize : Alloy.Globals.TAM_FUENTE
	}
}

"#lb_nombreProducto" :  {
	color : 'blue'
}

"#lb_precioPorUnidad" : {
	text : L 'lb_precioUnitario'
}

"#lb_precioTotal" : {
	text : L 'lb_precioTotal'
}
```

FICHERO XML

```
<Alloy>
	<Window id="win_productoCarrito" layout="vertical" backgroundColor="#F6EEE2">
		<View height="10dp" />
		<Label id="lb_nombreProducto" />
		<View height="10dp" />
		<ImageView id="iv_producto" height="120dp" width="120dp" borderRadius="5dp"
borderWidth="2dp" />
		<View height="10dp" />
		<Label id="lb_descripcionProducto" />
		<View height="30dp" />
		<View class="container" id="view_botones">
			<View class="container" layout="vertical" left="5dp" width="48%"
top="0dp">
				<View class="container" layout="horizontal" top="10dp"
height="50dp">
					<Label id="lb_precioPorUnidad" left="5dp" width="75dp"
textAlign="center"/>
					<Label id="lb_precioPorUnidadValue" textAlign="right"/>
				</View>
				<View class="container" layout="horizontal" height="50dp">
					<Label id="lb_precioTotal" left="5dp" width="75dp"
textAlign="center" />
					<Label id="lb_precioTotalValue" textAlign="right" />
				</View>
			</View>
			<View class="container" layout="horizontal" right="5dp" width="48%"
height="100dp" top="10%">
				<Button id="bt_menos"
backgroundImage="/common/images/bt_minus.png" height="40dp" width="40dp" />
				<View width="5dp" />
				<TextField id="tf_cantidad" height="50dp" width="50dp"
borderWidth="1dp" borderRadius="5dp" editable="false" textAlign="center"  />
				<View width="5dp" />
				<Button id="bt_plus"
backgroundImage="/common/images/bt_plus.png" height="40dp" width="40dp" />
```

```
                    </View>
                </View>
        </Window>
</Alloy>
```

7.3.10. PRODUCTOS

FICHERO JAVASCRIPT

```
//CARGA DE CATEGORÍAS DESDE BBDD NOSQUL
/**
 * Se recibe un evento de foco sobre la pestaña productos, entonces, se realiza una consulta a
la BBDD para obtener un JSON con todas las categorías
 * existentes en el servidor. Dicho JSON parseado (convertido en array) se pasa a la función
cargaCategorias() para que las añada a la interfaz gráfica.
 * Se utiliza otro método auxiliar y no se realiza en el método onload, porque una vez termine
la consulta, se libera toda la memoria utilizada.
 */
Ti.App.addEventListener("cargaCategorias", function() {
        $.activityIndicator.show();
        Ti.API.info("Recibo evento carga categorias");
        var getAllCategories = Ti.Network.createHTTPClient({
                username : Alloy.Globals.USERNAME,
                password : Alloy.Globals.PASSWORD,
                timeout : 3000,
                onerror : function(e) {
                        $.activityIndicator.hide();
                        Ti.API.info("Error en cargaCategorias: " + e.error);
                        alert("Error en CargaCategorias");
                },
                onload : function(e) {
                        cargaCategorias(JSON.parse(this.responseText));
                }
        });
        //Se prepara la conexión con el tipo (GET, PUT o POST) y la URL
        getAllCategories.open("GET", Alloy.Globals.SERVIDOR + "categories");
        //Se lanza la petición
        getAllCategories.send();

});
/**
 * Se recibe un array de strings con los datos de las categorías. Son interesantes el _id y el
name para la representación gráfica.
 * Con el name y con la imagen correspondiente (sólo hay que añadir /image al _id para que el
servidor nos devuelva la imagen) es
 * suficiente para rellenar gráficamente las filas de las tabla.
 * @param {String[]} categorias
 */
function cargaCategorias(categorias) {
        var rows = [];

        for (var i = 0; i < categorias.length; i++) {
                var row = Ti.UI.createTableViewRow({
                        id : categorias[i]._id,
                        hasChild : true
                });
                var leftImage = Ti.UI.createImageView({
                        image : Alloy.Globals.SERVIDOR + categorias[i]._id + "/image",
                        left : '5dp',
```

```
                        height : '50dp',
                        width : '50dp'
                });

                var categoria = Ti.UI.createLabel({
                        text : categorias[i].name,
                        width : 'auto',
                        textAlign : 'left',
                        left : '60dp',
                        height : '50dp',
                        color : 'black',
                        font : {
                                fontSize : Alloy.Globals.TAM_FUENTE,
                                fontFamily : 'Special Elite'
                        }
                });

                row.add(leftImage);
                row.add(categoria);

                row.className = 'category_row';

                rows.push(row);
        }

        $.listaCategorias.setData(rows);
        $.activityIndicator.hide();
        Ti.App.fireEvent('cargaCategoriasOk');
}

//CARGA DE PRODCUTOS POR CATEGORÍA DESDE BBDD NOSQL
/**
 * Se recibe un evento de foco sobre la pestaña productos, entonces, se realiza una consulta a
la BBDD para obtener un JSON con todos los productos
 * existentes en el servidor. Dicho JSON parseado (convertido en array) se pasa a la función
cargaProductos() para que las añada a la interfaz gráfica.
 * Se utiliza otro método auxiliar y no se realiza en el método onload, porque una vez termine
la consulta, se libera toda la memoria utilizada.
 */
$.listaCategorias.addEventListener('click', function(e) {
        Ti.API.info("Recibo evento carga productos");
        var getAllCategories = Ti.Network.createHTTPClient({
                username : Alloy.Globals.USERNAME,
                password : Alloy.Globals.PASSWORD,
                timeout : 3000,
                onerror : function(e) {
                        alert("Error en CargaCategorias: " + e.error);
                },
                onload : function(e) {
                        cargaProductos(JSON.parse(this.responseText));
                },
                cache : true
        });
        //Se prepara la conexión con el tipo (GET, PUT o POST) y la URL, a la que se le añade
el id de la categoría a buscar.
        getAllCategories.open("GET", Alloy.Globals.SERVIDOR + "products/category/" +
e.rowData.id);
        //Se lanza la petición
```

```
	getAllCategories.send();
});

/**
 * Se recibe un array de strings con los datos de los productos de una categoría. Nos interesa
el _id y el name para la representación gráfica.
 * Con el name y con la imagen correspondiente (sólo hay que añadir /image al _id para que el
servidor nos devuelva la imagen) es
 * suficiente para rellenar gráficamente las filas de las tabla. Se añadirá a cada fila, la
descripción y el precio de cada producto que serán
 * útiles para ampliar información acerca de un producto.
 * @param {String[]} productos
 */
function cargaProductos(productos) {

	if (productos.length == 0) {
		alert(L('noProductos'));
	} else {
		var win = Ti.UI.createWindow({
			title : 'Productos',
			class : 'container',
			backgroundColor : '#F6EEE2'
		});

		var listaProductos = Ti.UI.createTableView({
			objName : 'tabla',
			backgroundColor : 'transparent'
		});

		var rows = [];

		for (var i = 0; i < productos.length; i++) {
			var row = Ti.UI.createTableViewRow({
				height : '52dp',
				producto : productos[i]
			});

			var leftImage = Ti.UI.createImageView({
				image : Alloy.Globals.SERVIDOR + productos[i]._id + "/image",
				left : '5dp',
				height : '50dp',
				width : '50dp'
			});

			var productoLabel = Ti.UI.createLabel({
				text : productos[i].name,
				width : 'auto',
				textAlign : 'left',
				left : '60dp',
				color : 'black',
				top : '2dp',
				font : {
					fontSize : Alloy.Globals.TAM_FUENTE,
					fontFamily : 'Special Elite'
				}
			});

			var precioLabel = Ti.UI.createLabel({
```

```
                                text : productos[i].price.toFixed(2) + " €",
                                width : 'auto',
                                textAlign : 'right',
                                right : '2dp',
                                color : 'green',
                                bottom : '2dp',
                                font : {
                                        fontSize : '15dp'
                                }
                        });

                        row.add(leftImage);
                        row.add(productoLabel);
                        row.add(precioLabel);

                        row.className = 'product_row';

                        rows.push(row);
                }

                listaProductos.setData(rows);
                Ti.API.info("Datos cambiados en tabla");

                listaProductos.addEventListener('click', function(e) {
                        var dialog = Ti.UI.createOptionDialog({
                                title : L('añadirProductoCarrito') + " " +
e.rowData.producto.name,
                                options : [L('bt_confirmar'), L('bt_detalles'),
L('bt_cancelar')]
                        });
                        dialog.addEventListener('click', function(e2) {
                                if (e2.index == 0) {
                                        var producto = new Producto(e.rowData.producto.name,
e.rowData.producto.description, e.rowData.producto.price, e.rowData.producto._id,
e.rowData.producto.type, e.rowData.producto.itemType);
                                        Alloy.Globals.CARRITO.addProducto(producto, 1);
                                        Ti.App.fireEvent('actualizarBadge');
                                        alert(producto.getNombre() + " " + L('addedCarrito'));
                                } else if (e2.index == 1) {
                                        var winDetalles =
Alloy.createController('productosDetalle', {
                                                'producto' : e.rowData.producto
                                        }).getView();

                                        //Abrimos la ventana recién creada en la TAB actual de
productos.
                                        Alloy.Globals.TAB_PRODUCTOS.setWindow(winDetalles);
                                        Alloy.Globals.TAB_PRODUCTOS.open(winDetalles);
                                }
                        });

                        dialog.show();
                });

                win.add(listaProductos);
                Ti.API.debug("Tabla añadida");

                //Abrimos la ventana creaa en la TAB actual de productos.
```

```
                Alloy.Globals.TAB_PRODUCTOS.setWindow(win);
                Alloy.Globals.TAB_PRODUCTOS.open(win);
        }

};
```

FICHERO TSS

```
".container": {
        backgroundImage : '/common/images/fondo.png'
}

"#listaCategorias" : {
        backgroundImage : '/common/images/shopping_fondo.png'
}
```

FICHERO XML

```
<Alloy>
        <TableView class="categorias" id="listaCategorias" backgroundColor="#F6EEE2"/>
        <ActivityIndicator id="activityIndicator" message="Loading..."/>
</Alloy>
```

7.3.11. PRODUCTOSDETALLE

FICHERO JAVASCRIPT

```
//Se obtienen los datos recibidos como parámetros. Es este caso un JSON parseado de producto.
var args = arguments[0] || {};
//Almacenamos el arguemnto en una variable para mayor comodidad.
var producto = args.producto;

//Asignamos las etiquetas y la imagen correspondiente a la ventana, con los argumentos
recibidos.
$.lb_nombreProducto.setText(producto.name);
$.iv_producto.setImage(Alloy.Globals.SERVIDOR + producto._id + "/image");
$.lb_descripcionProducto.setText(producto.description);
$.lb_precio.setText(producto.price.toFixed(2) + " €");

/**
 * El boton de añadir productos al carrito permanece a la escucha de que se pulse. Cuando
reciba una pulsación,
 * se añade el prodcuto al carrito y se lanza un Dialog anunciándolo.
 */
$.bt_addProductoCarrito.addEventListener('click', function(e) {
        var p = new Producto(producto.name, producto.description, producto.price, producto._id,
producto.type, producto.itemType);
        Alloy.Globals.CARRITO.addProducto(p, 1);
        Ti.App.fireEvent('actualizarBadge');
        $.win_productoDetalle.close();
        alert(p.getNombre() + " " + L('addedCarrito'));
});
```

FICHERO TSS

```
".container": {
        backgroundColor: "white"
```

```
}

"#win_productoDetalle" : {
	title : L('detallesProducto'
}

"Label" : {
	color : 'black',
	font : {
		fontSize : Alloy.Globals.TAM_FUENTE
	}
}

"#lb_descripcionProducto" : {
	font : {
		fontFamily : 'Georgia-Italic'
	},
	textAlign : 'center'
}

"#lb_nombreProducto" :  {
	color : 'blue'
}

"#bt_addProductoCarrito" : {
	title : L 'bt_addProductoCarrito'
}
```

FICHERO XML

```
<Alloy>
	<Window id="win_productoDetalle" layout="vertical" backgroundColor="#F6EEE2">
		<View height="10dp" />
		<Label id="lb_nombreProducto" />
		<View height="10dp" />
		<ImageView id="iv_producto" height="150dp" width="150dp" borderRadius="5dp"
borderWidth="2dp" />
		<View height="10dp" />
		<Label id="lb_descripcionProducto" />
		<View height="10dp" />
		<Label id="lb_precio" />
		<View height="30dp" />
		<View class="container" id="view_botones" backgroundColor="transparent">
			<Button id="bt_addProductoCarrito" width="300dp"></Button>
		</View>
	</Window>
</Alloy>
```

7.3.12. PROMOCIONES

FICHERO JAVASCRIPT

```
//CARGA DE PROMOCIONES DESDE BBDD NOSQUL
/**
 * Se recibe un evento de foco sobre la pestaña promociones, entonces, se realiza una consulta
a la BBDD para obtener un JSON con todas las promociones
 * existentes en el servidor que estén activas. Dicho JSON parseado (convertido en array) se
pasa a la función cargaPromociones() para que las añada a la interfaz gráfica.
```

```
 * Se utiliza otro método auxiliar y no se realiza en el método onload, porque una vez termine
la consulta, se libera toda la memoria utilizada.
 */
Ti.App.addEventListener('cargaPromociones', function(e) {
	Ti.API.info("Recibo evento carga promociones");
	$.activityIndicator.show();

	var getAllCategories = Ti.Network.createHTTPClient({
		username : Alloy.Globals.USERNAME,
		password : Alloy.Globals.PASSWORD,
		timeout : 3000,
		onerror : function(e) {
			alert("Error en CargaPromociones: " + e.error);
		},
		onload : function(e) {
			cargaPromociones(JSON.parse(this.responseText));
		}
	});
	//Se prepara la conexión con el tipo (GET, PUT o POST) y la URL
	getAllCategories.open("GET", Alloy.Globals.SERVIDOR + "promos/active");
	//Se lanza la petición
	getAllCategories.send();
});
/**
 * Se recibe un array de strings con los datos de las promociones. Es interesante el _id para
obtener la imagen para la representación gráfica.
 * (sólo hay que añadir /image al _id para que el servidor nos devuelva la imagen). A cada
imagen le añadimos un evento de 'click' y la información
 * necesaria para añadir la promo al carrito de la compra. Se añade como si fuera un producto
más.
 * @param {String[]} _promociones
 */
function cargaPromociones(_promociones) {
	var promociones = [];

	for (var i = 0; i < _promociones.length; i++) {
		var img = Ti.UI.createImageView({
			image : Alloy.Globals.SERVIDOR + _promociones[i]._id + "/image",
			backgroundColor : "black",
			touchEnabled : "true",
			promo : _promociones[i]
		});

		promociones.push(img);
	}
	if (_promociones.length > 0) {
		$.promocionesView.views = promociones;
		var index = $.promocionesView.currentPage;
		Ti.API.info("Indice: " + index);
		$.lb_nombrePromocion.text = _promociones[index].name;
		$.lb_descripcionPromocion.text = _promociones[index].description;
		$.lb_precioPromocion.text = _promociones[index].price + " €";

	}

	$.activityIndicator.hide();
	Ti.App.fireEvent('promocionesCargadasOK');
};
```

```
/**
 * Evento que está a la escucha de que se realice un scroll en la vista de promociones. Cuando
el scroll termine, actualiza el nombre, la descripción y
 * el precio en las etiquetas correspondientes.
 */
$.promocionesView.addEventListener('scrollend', function(e) {
	$.lb_nombrePromocion.setText(e.view.promo.name);
	$.lb_descripcionPromocion.setText(e.view.promo.description);
	$.lb_precioPromocion.setText(e.view.promo.price + " €");
});

$.bt_addPromocion.addEventListener('click', function(e) {
	if ($.promocionesView.views.length == 0) {
		alert(L('noPromo'));
	} else {
		var index = $.promocionesView.currentPage;
		var promo = $.promocionesView.views[index].promo;

		var dialog = Ti.UI.createOptionDialog({
			title : L('añadirProductoCarrito') + " " + promo.name,
			options : [L('bt_confirmar'), L('bt_cancelar')]
		});
		dialog.addEventListener('click', function(e2) {
			if (e2.index == 0) {
				var producto = new Producto(promo.name, promo.description,
promo.price, promo._id, promo.type, promo.itemType);
				Alloy.Globals.CARRITO.addProducto(producto, 1);
				Ti.App.fireEvent('actualizarBadge');
				alert(producto.getNombre() + " " + L('addedCarrito'));
			}
		});

		dialog.show();
	}
});
```

FICHERO TSS

```
".container": {
	backgroundColor : Alloy.Globals.BACKGROUND_COLOR
}

"Label" : {
	font : {
		fontSize : '15dp',
		fontFamily : 'Special Elite'
	},
	color : 'black',
	textAlign : 'center'
}

"#lb_precioPromocion" : {
	font : {
		fontSize : '20dp',
		fontFamily : 'Special Elite'
	},
```

```
        color : 'green'
}
```

FICHEROS XML

```
<Alloy>
        <View class="container" id="view_promociones">
                <View class="container" layout="vertical">
                        <View height="10dp" />
                        <Label id="lb_nombrePromocion" height="auto" width="auto"
clickable="false" textAlign="center"/>
                        <View height="10dp" />
                        <ScrollableView id="promocionesView" showPagingControl="true"
backgroundColor="black" width="250dp" height="250dp" clickable="false"/>
                        <View height="10dp" />
                        <Label id="lb_descripcionPromocion" height="auto" width="auto"
clickable="false" textAlign="center"/>
                        <Label id="lb_precioPromocion" height="auto" width="auto"
clickable="false" textAlign="center"/>
                </View>
                <Button id="bt_addPromocion" bottom="5dp" right="5dp"
backgroundImage="/common/images/addButton.png" height="50dp" width="50dp" />
        </View>
        <ActivityIndicator id="activityIndicator" message="Loading..."/>
</Alloy>
```

7.3.13. REGISTRO

FICHERO JAVASCRIPT

```
//Constantes para controlar errores
var ERROR_CAMPO_BLANCO = 1;
var ERROR_PASS_DISTINTAS = 2;
var ERROR_CORREO = 3;
var ERROR_TELEFONO = 4;
var ERROR_CP = 5;
var REGISTRO_OK = 0;
/*
 * Se generan dos ventanas para abrir de forma modal para visualizar los picker. Es la única
forma que se ha encontrado de representarlos
 * sin errores tanto en IOS como en Android.
 */
var windowPickerFecha = Ti.UI.createWindow({
        layout : 'vertical',
        height : '400dp',
        width : '400dp',
        value : new Date(2007,05,13),
        backgroundColor : 'transparent',
        title : L('fecha'),
        fullscreen : false
});

var windowPickerSexo = Ti.UI.createWindow({
        height : '50%',
        backgroundColor : 'transparent',
        title : L('sexo'),
        fullscreen : false
});
```

```
/*
 * Se crea un picker que nos permita seleccionar el sexo. Sólo funcionará en IOS. En Android
se seleccionará el sexo mediante un diálogo de opciones.
 */
var pickerSexo = Ti.UI.createPicker();
var pickerData = [Ti.UI.createPickerRow({
	title : ''
}), Ti.UI.createPickerRow({
	title : L('hombre')
}), Ti.UI.createPickerRow({
	title : L('mujer')
})];

pickerSexo.selectionIndicator = true;
pickerSexo.add(pickerData);

windowPickerSexo.add(pickerSexo);

/*
 * Se genera una barra con un botón y un picker de tipo fecha para seleccionar al fecha de
nacimiento.
 */
var btDone = Ti.UI.createButton({
	title : 'Done',
	style : 1
});
var barraBotones = Ti.UI.createView();
if (Ti.Platform.osname == 'android') {
	barraBotones.height = '20%';
} else if (Ti.Platform.osname == 'iphone') {
	barraBotones.height = 40;
}

barraBotones.add(btDone);

var pickerFecha = Ti.UI.createPicker({
	type : Ti.UI.PICKER_TYPE_DATE
});

windowPickerFecha.add(barraBotones);
windowPickerFecha.add(pickerFecha);

/**
 * Se permanece a la escucha de un cambio en el picker de sexo. Cuando se cambie de valor se
escribe en el campo de texto correspondiente
 * Male o Female, que son los campos aceptados el WebService para añadir usuarios a la BBDD.
Posteriormente se cierra la ventana modal.
 */
pickerSexo.addEventListener('change', function(e) {
	Ti.API.info("Click en: " + pickerData[e.rowIndex].getTitle());
	var sexo = "";
	if (e.rowIndex == 1) {
		sexo = 'Male';
	} else if (e.rowIndex == 2) {
		sexo = 'Female';
	}

	$.tf_sexoValue.setValue(sexo);
```

```
	windowPickerSexo.close();
});

/**
 * Se permanece a la escucha de que el botón Done de la ventana modal con el picker fecha sea
pulsado. En caso de Android se añade al campo de texto correspondiente
 * la fecha seleccionada en el picker. En IOS esta función no funciona correctamente y hay que
relizarla mediante el evento change.
 */
btDone.addEventListener('click', function(e) {
	if (Ti.Platform.osname == 'android') {
		$.tf_fechaValue.setValue(pickerFecha.value.getFullYear() + "-" + ("0" +
(pickerFecha.value.getMonth() + 1)).slice(-2) + "-" + ("0" +
pickerFecha.value.getDate()).slice(-2));
	}

	windowPickerFecha.close();
	$.tf_fechaValue.blur();
});

/**
 * Se escucha el evento de cambio en los valores del picker para actualizar el campo de texto
correspondiente. Sólo funciona correctamente en IOS.
 */
pickerFecha.addEventListener('change', function(e) {
	if (Ti.Platform.osname == 'iphone') {
		$.tf_fechaValue.setValue(pickerFecha.value.getFullYear() + "-" + ("0" +
(pickerFecha.value.getMonth() + 1)).slice(-2) + "-" + ("0" +
pickerFecha.value.getDate()).slice(-2));

	}
});

/**
 * Escucha del evento click en el campo de texto de fecha, para abrir la ventana modal con el
picker de fecha.
 */
$.tf_fechaValue.addEventListener('click', function(e) {
	Ti.API.info("Click en fecha");
	windowPickerFecha.open({
		modal : true,
		width : '300dp',
		height : '300dp'
	});
});

/**
 * Escucha del evento click en el campo de texto sexo, para abrir la ventana modal con el
picker de sexo (en caso de IOS) o
 * el diálogo de opciones en caso de Android. El picker de una columna no funciona
correctamente en Android.
 */
$.tf_sexoValue.addEventListener('click', function(e) {
	Ti.API.info("Click en sexo");
	if (Ti.Platform.osname == 'iphone') {
		windowPickerSexo.open({
			modal : true
		});
```

```
	} else if (Ti.Platform.osname == 'android') {
		var dialog = Ti.UI.createOptionDialog({
			title : L('sexo'),
			options : ["", L('hombre'), L('mujer')]
		});
		dialog.addEventListener('click', function(e) {
			if (e.index == 1) {
				$.tf_sexoValue.setValue("Male");
			} else if (e.index == 2) {
				$.tf_sexoValue.setValue("Female");
			}
		});
		dialog.show();
	}
});

/**
 * Limpia todos los campos de texto de la ventana.
 */
function limpiar() {
	$.tf_nombre.setValue("");
	$.tf_apellido.setValue("");
	$.tf_email.setValue("");
	$.tf_direccion.setValue("");
	$.tf_cp.setValue("");
	$.tf_telefono.setValue("");
	$.tf_usuario.setValue("");
	$.tf_pass.setValue("");
	$.tf_fechaValue.setValue("");
	$.tf_sexoValue.setValue("");
};

/**
 * Permanece a la espera del evento click en el botón de confirmar. Llama al método
comprobacion() para recibir un código de error o éxito.
 * Muestra el mensaje correspondiente dependiendo del código recibido o llama al método
registerUser() si todas las comprobaciones son correctas.
 */
function doClickConfirmar() {
	var comprobacion = comprobarDatos();
	if (comprobacion == REGISTRO_OK) {
		var nombre = $.tf_nombre.getValue();
		var apellido = $.tf_apellido.getValue();
		var email = $.tf_email.getValue();
		var direccion = $.tf_direccion.getValue();
		var cp = $.tf_cp.getValue();
		var telefono = $.tf_telefono.getValue();
		var usuario = $.tf_usuario.getValue();
		var pass = $.tf_pass.getValue();
		var fecha = $.tf_fechaValue.getValue();
		var sexo = $.tf_sexoValue.getValue();

		registrarUser(usuario, pass, nombre, apellido, direccion, cp, fecha, sexo,
telefono, email);
	} else {
		switch(comprobacion) {
			case ERROR_CAMPO_BLANCO :
				alert(L('error_camposObligatorios'));
```

```
                                break;
                        case ERROR_PASS_DISTINTAS:
                                alert(L('error_passDistintas'));
                                break;
                        case ERROR_CORREO:
                                alert(L('error_correo'));
                                break;
                        case ERROR_CP:
                                alert(L('error_cp'));
                                break;
                        case ERROR_TELEFONO:
                                alert(L('error_telefono'));
                                break;
                }
        }
}

/**
 * Cierra la ventana registro.
 */
function volverALogin() {
        $.win_registro.close();
};

/**
 * Realiza una serie de comprobaciones para que los campos estén correctos. Si pasa todos los
filtros devuelve REGISTRO OK y sino el código correspondiente al
 * error detectado.
 */
function comprobarDatos() {
        var nombre = $.tf_nombre.getValue();
        var apellido = $.tf_apellido.getValue();
        var email = $.tf_email.getValue();
        var direccion = $.tf_direccion.getValue();
        var cp = $.tf_cp.getValue();
        var telefono = $.tf_telefono.getValue();
        var usuario = $.tf_usuario.getValue();
        var pass = $.tf_pass.getValue();
        var passConfirmar = $.tf_confirmarPass.getValue();
        var fecha = $.tf_fechaValue.getValue();
        var sexo = $.tf_sexoValue.getValue();

        if (nombre == "" || apellido == "" || email == "" || direccion == "" || cp == "" ||
telefono == "" || usuario == "" || pass == "" || fecha == "" || sexo == "" || passConfirmar ==
"") {
                return ERROR_CAMPO_BLANCO;
        } else if (pass != passConfirmar) {
                return ERROR_PASS_DISTINTAS;
        } else if (cp.length != 5 || isNaN(cp)) {
                return ERROR_CP;
        } else if (telefono.length != 9 || isNaN(telefono)) {
                return ERROR_TELEFONO;
        } else if (email.split("@").length != 2 || email.indexOf(".") == -1) {
                return ERROR_CORREO;
        }

        return REGISTRO_OK;
}
```

```
/**
 * Se genera un fichero JSON con los datos necesarios (que se reciben como parámetro) y se
realiza una petición POST para que el nuevo usuario se almacene en la BBDD.
 * Para ello, se envían las cabeceras correspondientes y el JSON con los datos de usuario.
 *
 * @param {String} _usuario
 * @param {String} _password
 * @param {String} _nombre
 * @param {String} _apellido
 * @param {String} _direccion
 * @param {String} _cp
 * @param {String} _fechaDeNacimiento
 * @param {String} _sexo
 * @param {String} _telefono
 * @param {String} _email
 */
function registrarUser(_usuario, _password, _nombre, _apellido, _direccion, _cp,
_fechaDeNacimiento, _sexo, _telefono, _email) {
	var fechaFormato = _fechaDeNacimiento + "T12:00:00.000Z";
	var json = "{\"_id\": \"org.couchdb.user:" + _usuario + "\",\"type\": \"user\",
\"name\": \"" + _usuario + "\",\"roles\": [\"mobimarket_user\"],\"firstName\": \"" + _nombre +
"\",\"lastName\": \"" + _apellido

                    "JSON para Registrar: " + json);

	var url = Alloy.Globals.SERVIDOR + "register";
	var tipo = "POST";

	this.httpClient = Ti.Network.createHTTPClient({
		onerror : function(e) {
			Ti.API.info("Error en registrar user: " + e.error);
			alert(L('error_accesoBBDD'));
		},
		onload : function() {
			Ti.API.info("Estoy en onload");
			Ti.App.fireEvent('registroOk');
		},

		timeout : Alloy.Globals.TIEMPO_RESPUESTA
	});

	this.httpClient.open(tipo, url);
	//Se envía la cabecera indicando el tipo que será un JSON con los datos del usuario a
registrar
	this.httpClient.setRequestHeader('Content-Type', 'application/json; charset=utf-8');
	//Se realiza la petición enviando como parámentros el JSON
	this.httpClient.send(json);
};

/**
 * Permanece a la escucha del evento 'ventanaRegistro' para abrir la ventana de registro.
 */
Ti.App.addEventListener('ventanaRegistro', function(e) {
	$.win_registro.open();
});
```

FICHEROS TSS

```
".container": {
	backgroundColor: "white"
}

"#tvs_datosPersonales" : {
	headerTitle: 'Datos Personales'
}

"#tvs_datosUsuario" : {
	headerTitle: 'Datos de Usuario'
}

"#view_datosPersonales[platform=ios]" : {
	height: "400dp"
}

"#view_datosPersonales[platform=android]" : {
	height: "550dp"
}

"#view_datosUsuario[platform=ios]" : {
	height: "180dp"
}

"#view_datosUsuario[platform=android]" : {
	height: "200dp"
}

//ESTILO DE LABELS

"#lb_tituloDatosPersonales" : {
	text: L('datosPersonales'
		'white',
	font : {
		fontSize : '16dp'
	}
}

"#lb_tituloDatosUsuario" : {
	text: L 'datosUsuario'
		'white',
	font : {
		fontSize : '16dp'
	}
}


"#lb_nombre" : {
	text: L 'nombre'


"#lb_apellido" : {
	text: L 'apellidos'
```

```
"#lb_email" : {
	text: L 'email'
},

"#lb_cp" : {
	text: L 'cp'
},

"#lb_telefono" : {
	text: L 'telefono'
},

"#lb_direccion" : {
	text: L 'direccion'
},

"#lb_sexo" : {
	text: L 'sexo'
},

"#lb_fecha" : {
	text: L 'fecha'
},

"#lb_usuario" : {
	text: L 'usuario'
},

"#lb_pass" : {
	text: L 'pass'
},

"#lb_confirmarPass" : {
	text: L 'confirmarPass'
},

//ESTILOS DE TEXTFIELD
"#tf_usuario" : {
	autocapitalization: Titanium.UI.TEXT_AUTOCAPITALIZATION_NONE
},

"#tf_pass" : {
	passwordMask: true
},

"#tf_confirmarPass" : {
	passwordMask: true
},

//ESTILOS GLOBALES

"Label" : {
	left: '3%',
	font: {
		size: Alloy.Globals.TAM_FUENTE,
		fontFamily : 'Special Elite'
	},
	textAlign: 'center',
```

```
        color : 'black'
}

"TextField" : {
        left: '8%',
        width: '85%',
        value: "",
        backgroundColor: 'white',
        borderRadius: '5dp'
}

// BOTONES

"#bt_confirmar" : {
        title : L 'bt_confirmar'
}

"#bt_limpiar" : {
        title : L 'bt_limpiar'
}

"#bt_login" : {
        title : L 'bt_login'
}
```

FICHERO XML

```
<Alloy>
        <Window id="win_registro" class="container" backgroundColor="#F6EEE2"
layout="vertical">
                <ScrollView id="tv_registro" layout="vertical" height="80%" width="90%"
contentHeight="auto" contentWidth="auto">
                        <View id="view_datosPersonales" backgroundColor="transparent"
layout="vertical">
                                <Label id="lb_tituloDatosPersonales" />
                                <View height="8dp" />
                                <Label id="lb_nombre" />
                                <TextField id="tf_nombre" />
                                <View height="5dp" width="95%" />
                                <Label id="lb_apellido" />
                                <TextField id="tf_apellido" />
                                <View height="5dp" width="95%" />
                                <Label id="lb_email" />
                                <TextField id="tf_email" />
                                <View height="5dp" width="95%" />
                                <Label id="lb_direccion" />
                                <TextField id="tf_direccion" />
                                <View height="5dp" width="95%" />
                                <Label id="lb_cp" />
                                <TextField id="tf_cp" />
                                <View height="5dp" width="95%" />
                                <Label id="lb_telefono" />
                                <TextField id="tf_telefono" />
                                <View height="5dp" width="95%" />
                                <Label id="lb_fecha" />
                                <TextField id="tf_fechaValue" editable="false" />
                                <View height="5dp" width="95%" />
```

```
                            <Label id="lb_sexo" />
                            <TextField id="tf_sexoValue" editable="false" />
                        </View>
                        <View id="view_datosUsuario" backgroundColor="transparent"
layout="vertical">
                            <Label id="lb_tituloDatosUsuario" />
                            <View height="8dp" />
                            <Label id="lb_usuario" />
                            <TextField id="tf_usuario" />
                            <View height="5dp" width="95%" />
                            <Label id="lb_pass" />
                            <TextField id="tf_pass" />
                            <View height="5dp" width="95%" />
                            <Label id="lb_confirmarPass" />
                            <TextField id="tf_confirmarPass" />
                            <View height="5dp" width="95%" />
                        </View>
                </ScrollView>
                <View id="view_botton" height="20%">
                        <Button id="bt_confirmar" bottom="0%" left="3%" width="30%"
onClick="doClickConfirmar" />
                        <Button id="bt_limpiar" bottom="0%" left="35%" width="30%"
onClick="limpiar"/>
                        <Button id="bt_login" bottom="0%" left="67%" width="30%"
onClick="volverALogin" />
                </View>
        </Window>
</Alloy>
```

7.3.14. SEARCHBAR

FICHERO JAVASCRIPT

```
```

FICHERO TSS

```
"#searchBar" : {
        hintText : L('sb_buscaSupermercados'
```

FICHERO XML

```
<Alloy>
    <SearchBar id="searchBar" showCancel="false" width="10%" backgroundColor="gray"/>
</Alloy>
```

7.3.15. HELPERS

CARGASUPERMERCADOS.JS

```
var Supermercado = require('Supermercado');

function CargaSupermercados() {
```

```
	var jsonSuperMercados = '{"supermercados" : [{"nombre" : "MobiMarket Gijón",
"direccion" : "C Uria 22", "telefono" : "985 475 457", "horario" : "24H", "coordenadas" :
"43.6458,-6.2365"}, {"nombre" : "MobiMarket Oviedo", "direccion" : "C/Los Arcos 55",
"telefono" : "985 225 026", "horario" : "9h-21h", "coordenadas" : "43.2145,-6.9877"}]}'
	var superMercadosParse = JSON.parse(jsonSuperMercados);

	this.supermercados = [];

	for(var i=0; i<superMercadosParse.supermercados.length; i++) {
		var supermercado = new Supermercado(
			superMercadosParse.supermercados[i].nombre,
			superMercadosParse.supermercados[i].direccion,
			superMercadosParse.supermercados[i].telefono,
			superMercadosParse.supermercados[i].horario,
			superMercadosParse.supermercados[i].coordenadas,
			"" //Servicios
		);
		this.supermercados.push(supermercado);
	}

	this.getNumSupermercados = function() {
		return this.supermercados.length;
	};
	this.getSupermercado = function(_pos) {
		return this.supermercados[_pos];
	};
};

module.exports = CargaSupermercados;
```

CARRITO.JS

```
/**
 * Manejamos dos array uno con los productos y otro con las cantidades que se va a comprar.
 * @class Carrito
 */
function Carrito() {
	this.productos = [];
	this.cantidad = [];

	this.getProductos = getProductos;
	this.getCantidades = getCantidades;
	this.addProducto = addProducto;
	this.modificarCantidad = modificarCantidad;
	this.buscaProducto = buscaProducto;
	this.quitaProducto = quitaProducto;
	this.calculaTotal = calculaTotal;
	this.vaciaCarrito = vaciaCarrito;
	this.getNumProductos = getNumProductos;
	this.getProducto = getProducto;
	this.getCantidad = getCantidad;
	this.estaVacio = estaVacio;
};

/**
 * Devuelve el array de cantidades de los productos que están actualmente en el carrito.
 * @method getCantidades
 */
```

```
var getCantidades = function() {
	return this.cantidad;
};

/**
 * Devuelve el array de productos que están actualmente en el carrito
 */
var getProductos = function() {
	return this.productos;
};

/**
 * Recibe un parámetro y lo busca en la lista de productos. Si lo encuentra devuelve su
posición en el array, sino devuelve -1
 * @param {int} _id
 */
var buscaProducto = function(_id) {
	Ti.API.info("ID a buscar: " + _id);
	for (var i = 0; i < this.productos.length; i++) {
		if (this.productos[i].getId() == _id) {
			Ti.API.info('Producto encontrado en la posición ' + i);
			return i;
		}
	}

	Ti.API.info("Producto no encontrado");

	return -1;
};

/**
 * Busca si el producto recibido como parámetro está en el carrito. Si existe suma la cantidad
indicada. Si no añade el producto a la lista y la cantidad a la lista de cantidades.
 * @param {Producto} _producto
 * @param {int} _cantidad
 */
var addProducto = function(_producto, _cantidad) {
	var pos = this.buscaProducto(_producto.getId());
	Ti.API.info("ID producto a meter: " + _producto.getId());
	if (pos == -1) {
		this.productos.push(_producto);
		this.cantidad.push(_cantidad);
		Ti.API.info("Producto añadido al carrito");
	} else {
		this.cantidad[pos] += _cantidad;
		Ti.API.info("Producto " + _producto.getNombre() + " incrementado en " +
_cantidad + " unidades");
	}
};

/**
 * Busca el producto en el carrito y modifica la cantidad en la lista de cantidades. Si la
cantidad es cero o menor que cero, elimina el producto de la lista.
 * @param {int} _id
 * @param {int} _cantidad
 */
var modificarCantidad = function(_id, _cantidad) {
	var pos = this.buscaProducto(_id);
```

```
	if (pos == -1) {
		Ti.API.info("El producto no se encuentra en la lista");
	} else if (_cantidad > 0) {
		this.cantidad[pos] = _cantidad;
		Ti.API.info("Producto con id " + _id + " modificado en " + _cantidad + "
unidades");
	} else {
		quitaProducto(_id);
	}
};

/**
 * Busca el producto en la lista a través de su id. Si lo encuentra, su posición en el array
es sustituida por el último elemento de la lista. Luego se saca el último elemento.
 * @param {int} _id
 */
var quitaProducto = function(_id) {
	var pos = this.buscaProducto(_id);
	if (pos == -1) {
		Ti.API.info("El producto a borrar no está en la lista");
	} else {
		this.productos[pos] = this.productos[this.cantidad.length - 1];
		this.cantidad[pos] = this.cantidad[this.cantidad.length - 1];
		this.productos.pop();
		this.cantidad.pop();
		Ti.API.info("Producto borrado correctamente");
	}
};

var quitaProductoPos = function(_pos) {
	if (pos < this.productos.length) {
		Ti.API.info("El producto a borrar no esta en la lista");
	}
}
/**
 * Recorre todos los productos que tengamos en la lista y devuelve el precio total.
 */
var calculaTotal = function() {
	var total = 0.0;
	if (this.productos.length == 0) {
		Ti.API.info("No hay productos en el carrito");
	} else {
		for (var i = 0; i < this.productos.length; i++) {
			total += this.productos[i].getPrecio() * this.cantidad[i];
		}
	}

	total = total.toFixed(2);

	Ti.API.info("El total del carrito es: " + total);
	//El método toFixed() nos permite establecer el número de decimales
	return total;
};

/**
 * Vacía los arrays de productos y cantidades.
 */
var vaciaCarrito = function() {
```

```
	this.productos = [];
	this.cantidad = [];
	Ti.API.info("Carrito vaciado");
};

/**
 * Retorna el número de productos que hay en la lista del Carrito.
 */
var getNumProductos = function() {
	return this.productos.length;
};

/**
 * Busca la posición del producto a buscar en el array y lo devuelve.
 * @param {int} _pos
 */
var getProducto = function(_id) {
	var pos = this.buscaProducto(_id);
	if (pos == -1) {
		Ti.API.info("El producto a buscar no está en la lista");
	} else {
		return this.productos[pos];
	}
};

/**
 * Retorna la cantidad de la posición recibida como parámetro.
 * @param {int} _id
 */
var getCantidad = function(_id) {
	var pos = this.buscaProducto(_id);
	if (pos == -1) {
		Ti.API.info("El producto a buscar no está en la lista");
	} else {
		return this.cantidad[pos];
	}
};

/**
 * Devuelve un booleano con true si no hay productos y con false si los hay
 */
var estaVacio = function() {
	if (this.productos.length == 0) {
		return true;
	} else {
		return false;
	}
}
//Exportamos la clase carrito para que pueda ser utilizada por otros controladores de Alloy
module.exports = Carrito;
```

COORDENADAS.JS

```
function Coordenadas(_latitud, _longitud) {
	this.longitud = _longitud;
	this.latitud = _latitud;

	this.getLatitud = function() {
```

```
                return this.latitud;
        };
        this.getLongitud = function() {
                return this.longitud;
        };
};
```

PEDIDO.JS

```
/**
 * Clase muy simple que se encarga de almacenar los datos necesarios para la realizacion de un
pedido con sus getters y setters correspondientes.
 *
 * @param {String} _id
 * @param {String} _fecha
 * @param {String} _precio
 * @param {String} _estado
 * @param {String} _productos
 */
function Pedido(_id, _fecha, _precio, _estado, _productos) {
        this.id = _id;
        this.fecha = _fecha;
        this.precio = _precio;
        this.estado = _estado;
        this.productos = _productos;

        this.getId = function() {
                return this.id;
        };
        this.getFecha = function() {
                return this.fecha;
        };
        this.getPrecio = function() {
                return this.precio;
        };
        this.getEstado = function() {
                return this.estado;
        };

        this.setId = function(_id) {
                this.id = _id;
        };
        this.setFecha = function(_fecha) {
                this.fecha = _fecha;
        };
        this.setPrecio = function(_precio) {
                this.precio = _precio;
        };
        this.setEstado = function(_estado){
                this.estado = _estado;
        };
};
//Exporta la clase de forma que se pueda utilizar desde los controladores que lo incluyan
mediante la orden Require('Pedido')
module.exports = Pedido;
```

PRODUCTO.JS

```
/**
 * Clase simple con los datos que se necesitan de cada producto de un supermercado con sus
correspondientes Getters.
 * @param {String} _nombre
 * @param {String} _descripcion
 * @param {String} _precio
 * @param {String} _id
 * @param {String} _type
 * @param {String} _itemType
 */
function Producto(_nombre, _descripcion, _precio, _id, _type, _itemType) {
	this.nombre = _nombre;
	this.descripcion = _descripcion;
	this.precio = _precio;
	this.id = _id;
	this.type = _type;
	this.itemType = _itemType;

	this.getNombre = function() {
		return this.nombre;
	};
	this.getDescripcion = function() {
		return this.descripcion;
	};
	this.getPrecio = function() {
		return this.precio;
	};
	this.getId = function() {
		return this.id;
	};
	this.getType = function() {
		return this.type;
	};
	this.getItemType = function() {
		return this.itemType;
	};
};

//Se exporta la clase de forma que pueda ser incluída por otro controllador de la aplicación
mediante la orden Require('Producto')
module.exports = Producto;
```

PRUEBAPRODUCTOS.JS

```
var Producto = require('Producto');

function PruebaProductos() {
	this.productos = [];

	this.cargaProductos = function(_numProductos) {
		for(var i=0; i<_numProductos; i++) {
			var producto = new Producto("Producto: " + i, "Este es el producto " +
i, i*3, i);
			this.productos.push(producto);
		}
	};
```

```
	this.getProducto = function(_pos) {
		return this.productos[_pos];
	};
};

module.exports = PruebaProductos;
```

REQUEST.JS

```
var HEADER_TIPO = 0;
var HEADER_CONTENIDO = 1;

/**
 * Creamos una consulta para CouchDB con un nombre de usuario y una contraseña. La propia
librería de Titanium se encarga del codificar y decodificar usuario y contraseña en base64.
 * @param {String} _username
 * @param {String} _password
 */
function Request(_username, _password) {
	this.httpClient = Ti.Network.createHTTPClient({
		username : _username,
		password : _password,

		timeout : 3000
	});

	this.getCategorias = getCategorias;
	this.getProductosDeCategoria = getProductosDeCategoria;
	this.getPromociones = getPromociones;
	this.getItem = getItem;
	this.getSupermercados = getSupermercados;
	this.postUsuario = postUsuario;
};

/**
 * Realiza una consulta a la BBDD indicado tipo (que puede ser GET, PUT o POST); consulta (que
puede ser
 * login, categories, product/category, image, productos/active, items, restaurants y
register); el mensaje de error que se mostrará en caso de haber problemas;
 * un identificador para la búsqueda; una cabecera que es un array de tamaño dos y que consta
del tipo de cabecera en la primera posición y del contenido en la segunda.
 * Por ejemplo ["Content-type", "JSON/data"]; y por último los parámetros. El identificador,
la cabecera y los parámetros deberán pasarse con el valor null en caso de
 * no ser necesarios par una consulta determinada.
 * @param {String} _tipo
 * @param {String} _consulta
 * @param {String} _errMensaje
 * @param {String} _id
 * @param {String[]} _cabecera
 * @param {String} _parametros
 */
function consulta(_tipo, _consulta, _errMensaje, _id, _cabecera, _parametros) {
	var url = Alloy.Globals.SERVIDOR;
	if (_id != null) {
		url += _consulta + "/" + _id;
	} else {
		url += _consulta;
```

```
	}

	var data = [];

	Ti.API.info("========== CONSULTA ==========");
	Ti.API.info("User: " + this.httpClient.getUsername());
	Ti.API.info("Pass: " + this.httpClient.getPassword());
	Ti.API.info("URL: " + url);
	Ti.API.info("Tipo consulta: " + _tipo);
	Ti.API.info("==============================");

	this.httpClient.onerror = function() {
		alert(_errMensaje);
	};
	this.httpClient.onload = function() {
		switch(_consulta) {
			case "login":
				if (this.status == 200) {
					Alloy.Globals.USERNAME = userValue;
					Alloy.Globals.PASSWORD = passValue;

					alert("Login OK");
				} else {
					alert("Usuario o contraseña incorrectos");
				}
				break;

			case "categories":
				categoriasParse = JSON.parse(this.responseText);

				Ti.App.fireEvent("consultaDatosCategorias", {
					categorias : categoriasParse
				});
				break;

			case "products/category":
				var productosByCategoryIdParse = JSON.parse(this.responseText);

				Ti.App.fireEvent("consultaDatosProductosByCategoryId", {
					productos : productosByCategoryIdParse
				});
				break;

			case "promos/active":
				var promosActivasParse = JSON.parse(this.responseText);

				Ti.App.fireEvent("consultaCargaPromociones", {
					promociones : promosActivasParse
				});
				break;

			case "items":
				var itemParse = JSON.parse(this.responseText);

				Ti.App.fireEvent("consultaItem", {
					item : itemParse
				});
				break;
```

```
                        case "restaurants":
                                var supermercadosParse = JSON.parse(this.responseText);

                                Ti.App.eventFire("consultaSupermercados", {
                                        supermercados : supermercadosParse
                                });
                                break;
                        case "register":
                                Ti.App.fireEvent("registroOk", {
                                        registro : true
                                });
                                break;

                }
        };

        this.httpClient.open(_tipo, url);
        Ti.API.info("HTTPClient abierto");
        if (_cabecera != null && cabecera.length == 2) {
                httpClient.setRequestHeader(_cabecera[HEADER_TIPO],
_cabecera[HEADER_CONTENIDO]);
                Ti.API.info("HTTPClient cabecera enviada");
        }
        if (_parametros != null) {
                this.httpClient.send(_parametros);
        } else {
                this.httpClient.send();
        }

};

/**
 * Realiza una consulta a la BBDD que devuelve un JSON con todas las categorías mediante el
evento consultaDatosCategorias
 */
var getCategorias = function() {
        consulta("GET", "categories", L('err_accessoBBDD'), null, null, null);
};
/**
 * Realiza una consulta a la BBDD devolviendo un fichero JSON con todos los productos de una
categoría determinada (cuyo id es recibido como parámetro)
 * mediante el evento consultaDatosProductosByCategoryId
 * @param {String} _idCategoria
 */
var getProductosDeCategoria = function(_idCategoria) {
        consulta("GET", "products/category", L('err_accesoBBDD'), _idCategoria, null, null);
};
/**
 * Realiza una consulta a la BBDD que devuelve un JSON con todas las promociones activas
mediante el evento consultaCargaPromociones
 */
var getPromociones = function() {
        consulta("GET", "promos/active", L('err_accesoBBDD'), null, null, null);
}
var getItem = function(_idItem) {
        consulta("GET", "items", L('err_accesoBBDD'), _idItem, null, null);
}
```

```
var getSupermercados = function() {
	consulta("GET", "restaurants", L('err_accesoBBDD'), null, null, null);
};
var postUsuario = function(_usuario, _password, _nombre, _apellido, _direccion, _cp,
_fechaDeNacimiento, _sexo, _telefono, _email) {
	var json = "{\"_id\": \"org.couchdb.user:" + _usuario + "\",\"type\": \"user\",
\"name\": \"" + _usuario + "\",\"roles\": [\"mobimarket_user\"],\"firstName\": \"" + _nombre +
"\",\"lastName\": \"" + _apellido

		"POST", "register", "No se puede registrar al usuario", null, ["Content-Type",
"application/json; charset=utf-8"], json);

};

module.exports = Request;
```

SUPERMERCADO.JS

```
var Coordenadas = require('Coordenadas');

var Supermercado = function(_nombre, _direccion, _telefono, _horario, _coordenadas,
_servicios) {
	this.nombre = _nombre;
	this.direccion = _direccion;
	this.telefono = _telefono;
	this.horario = _horario;
	this.coordenadas = _coordenadas;
	this.servicios = _servicios;

	this.getNombre = getNombre;
	this.getDireccion = getDireccion;
	this.getTelefono = getTelefono;
	this.getCoordenadas = getCoordenadas;
	this.getServicios = getServicios;
};

var getNombre = function() {
	return this.nombre;
};

var getDireccion = function() {
	return this.direccion;
};

var getTelefono = function() {
	return this.telefono;
};

var getHorario = function() {
	return this.horario;
};

var getCoordenadas = function() {
	return this.coordenadas;
```

```
};

var getServicios = function() {
	return this.servicios;
};

module.exports = Supermercado;
```

7.3.16. TEST DE ROBOTIUM PARA MOBIMARKET

```
package com.example.mapping.test;

import com.jayway.android.robotium.solo.Solo;

@SuppressWarnings("unchecked")
public class Test extends ActivityInstrumentationTestCase2 {

	// OPC 1: Titanium APK
	private static final String TARGET_PACKAGE_ID = "com.example.mapping";
	private static final String LAUNCHER_ACTIVITY_FULL_CLASSNAME =
"com.example.mapping.MobimarketActivity";

	// private static final String TARGET_PACKAGE_ID="com.ctic.burgerking";
	// private static final String
	// LAUNCHER_ACTIVITY_FULL_CLASSNAME="com.ctic.burgerking.LoginActivity";

	// private static final String TARGET_PACKAGE_ID="com.example.holamundo";
	// private static final String
	// LAUNCHER_ACTIVITY_FULL_CLASSNAME="com.example.holamundo.MainActivity";
	private static Class launcherActivityClass;
	static {

		try {
			launcherActivityClass = Class
					.forName(LAUNCHER_ACTIVITY_FULL_CLASSNAME);
		} catch (ClassNotFoundException e) {
			throw new RuntimeException(e);
		}
	}

	public Test() throws ClassNotFoundException {
		super(launcherActivityClass);
	}

	private Solo solo;

	@Override
	protected void setUp() throws Exception {
		solo = new Solo(getInstrumentation(), getActivity());
	}

	public void testDisplayBlackBox() {

		// MobiMarket: titanium apk
		solo.clearEditText(0);
		solo.clearEditText(1);
		solo.enterText(0, "cris");
		solo.enterText(1, "cris");
```

```
            solo.clickOnButton("LOGIN");
            solo.clickOnText("Productos");
            solo.clickOnText("CarnicerÌa");
            solo.clickOnText("ChuletÛn de buey");
            solo.clickOnText("Confirmar");
            solo.clickOnButton("Aceptar");
            solo.goBack();
            solo.clickOnText("PanaderÌa");
            solo.clickOnText("Pan artesano");
            solo.clickOnText("Confirmar");
            solo.clickOnButton("Aceptar");
            solo.goBack();
            solo.clickOnText("Pedidos");
            solo.clickInList(1);
            solo.clearEditText(0);
            solo.enterText(0, "2");
            solo.goBack();
            solo.clickOnButton("Pedido");
            solo.clickOnText("Confirmar");
            solo.clickOnButton("Aceptar");
            solo.clickOnMenuItem("Logout");

            // solo.clickOnButton("OK");
            // solo.clickOnActionBarItem(1);
            // solo.clickOnActionBarItem(1);
            // solo.clickOnImageButton(1);
            // solo.clickOnMenuItem("Logout");
            // solo.clickOnActionBarHomeButton();
            // solo.clickOnButton("OK");
            // solo.clickOnActionBarItem(2);

            // OPCION 2:
            /*
             * solo.enterText(0, "mobirest"); solo.enterText(1, "mobirest");
             * solo.clickOnButton("Entrar"); //solo.pressMenuItem(0);
             * solo.clickOnImageButton(2); solo.clickOnButton("Desloguearse");
             * solo.clickOnButton("Aceptar");
             */

            // OPCION 3:
            /*
             * boolean actual = solo.searchText("Hola Mundo"); assertEquals(true,
             * actual);
             */
    }

    @Override
    public void tearDown() throws Exception {
            solo.finishOpenedActivities();
    }
```

7.4. DIARIO DE DESARROLLO DEL PROYECTO

En este apéndice se incluirá el diario con las anotaciones del autor. Se describen las tareas realizadas con una aproximación del tiempo invertido en horas. No se incluyen por ejemplo, las anotaciones realizadas a lo largo de la codificación o las tutorías con ninguno de los dos tutores.

7.4.1. MARZO

DÍA	TAREAS	TIEMPO (en horas)
11	Presentación CTIC	1
	Preparación del equipo	6
	Instalación del entorno de desarrollo	1
12	Instalación del entorno de desarrollo	5
	Formación: Titanium Studio	3
13	Formación: Titanium Studio	6
14	Formación: Titanium Studio	1
	Formación: JavaScript	7
15	Formación: JavaScript	6
16	Formación: Titanium Studio	2
18	Formación: Alloy	6
	Formación: CSS, XML	3
19	Formación: Alloy	3
	Estudio de la aplicación MobiRestaurant	3
20	Creación de la interfaz principal	9
21	Codificación subsistema: Localización	6
	Documentación: Planificación del proyecto	2
22	Formación: Alloy (Barra de búsqueda y mapa)	6
23	Formación: Alloy (Barra de búsqueda y mapa)	3
24	Formación: Herramienta Pencil	2
	Creación y documentación de prototipos de IU	3
25	Codificación subsistema: Localización	6
26	Formación: Alloy (Tablas y Eventos)	6
	Codificación subsistema: Productos	2
27	Formación: BBDD noSQL	3
	Formación: Web Services para el proyecto	3
28	Formación: Herramientas SoapUI y SourceTree	3
	Codificación subsistema: Productos	3
	Documentación: Requisitos y pruebas	3
29	Codificación subsistema: Productos	6
	TOTAL HORAS	119

7.4.2. ABRIL

DÍA	TAREAS	TIEMPO (en horas)
1	Prubas de subsistema: Productos	3
	Formación: Vista de imágenes en scroll	2
	Codificación: Subsistema Promociones	1
	Documentación: Requisitos y pruebas	2
2	Codificación: Subsistema Promociones	2
	Codificación: Subsistema Carrito	4
3	Pruebas: Subsistema Promociones	1
	Codificación: Subsistema Carrito	5
	Documentación: Requisitos y pruebas	2
8	Documentación: Requisitos y pruebas	1
	Codificación: Subsistema Carrito	5
9	Documentación: Requisitos y pruebas	1
	Creación de iconos	3
8	Formación: Autentificación en Alloy	5
	Codificación: Subsistema de registro y login	1
10	Codificación: Subsistema de registro y login	6
11	Pruebas: Carrito y login	2
	Corrección de errores	4
12	Pruebas: Sistema compelto IOS	2
	Formación: Desplegar aplicaciones en terminal IoS	5
15	Corrección de errores	6
16	Corrección de errores	1
	Pruebas: Sistema completo Android	3
	Cambio en el sistema de consulta a BBDD	2
17	Cambio en el sistema de consulta a BBDD	5
	Cambio de los tamaños de letra en Android	1
18	Cambio de los tamaños de letra en Android	1
	Cambio de las tablas en android	5
	Formación: Errores en Android de IU	3
19	Formación: Errores en Android de IU	2
	Cambio en los layouts Android de Registro y Login	4
22	Cambio en los layouts Android de Registro y Login	6
	Cambio en los layouts para Android de producto	2
23	Cambio en los layouts para Android de Supermercado	1
	Corrección de eventos para Andoid	5
24	Corrección de tamaños de IU para Android	6
	Documentación: Manuales de Desarrollador	2
25	Corrección de errores de IU Android	6

26	Documentación: Manuales de Desarrollador	2
29	Pruebas y correción sistema Android	6
	TOTAL HORAS	126

7.4.3. MAYO

DÍA	TAREAS	TIEMPO (en horas)
2	Correción de conexiones a la BBDD	6
	Documentación: Manuales del desarrollador	2
3	Documentación: Manuales del desarrollador	1
	Cambio de tablas en sistema Android	5
	Documentación: Diseño de pruebas	3
5	Documentación: Maquetación	2
6	Cambio de tablas en sistema Android	6
7	Cambio de tablas en sistema Android	2
	Modificación de JSON	4
8	Modificación de JSON	3
	Inclusión de Badge en iOS	3
	Documentación: Manual de desarrollador	2
9	Inclusión de Badge en iOS	1
	Eliminación del apaisado en iOS	2
	Eliminación del apaisado en Android	3
	Eliminación del apaisado en Android	2
10	Eliminación del apaisado en Android	3
	Impedir que una aplicación se cierre cuando está en backgrond	6
11	Documentación: Manual de usuario	2
12	Documentación: Manual de usuario	3
13	Correcciones en la aplicación Android	6
14	Correcciones en la aplicación Android	6
	Documentación: Referencias	2
15	Correcciones de la aplicación en Android	4
	Uso de properties para almacenaje de datos	2
16	Uso de properties para el almacenaje de dato	5
	Investigación a cerca de SysTrace	3
17	Funcionamiento de Robotium	6
20	Refinamiento de interfaces de usuario: Pickers	6
21	Refinamiento de interfaces de usuario: Pickers	4
	Documentación: JavaDoc	2
22	Documentación: JavaDoc	6

	Funcionamiento de Robotium	1
23	Medicciones de SysTrace	6
	Investigación a cerca de SysTrace	2
24	Medicciones de SysTrace	3
	Creación de mediciones en Robotium	3
	Investigación a cerca de SysTrace	3
27	Comparación: Mediciones con Robotium	4
	Codificación: Pruebas del servidor	2
28	Ventana de carga	6
	Documentación: papeles del proyecto	2
29	Ventana de carga	2
	Realización de pruebas	4
	TOTAL HORAS	151

7.4.4. JUNIO

DÍA	TAREAS	TIEMPO (en horas)
10	Pruebas: Realización de pruebas	3
11	Documentación: documentar pruebas	2
12	Evaluación: pruebas con Systrace	2
13	Evaluación: pruebas con Systrace	3
14	Evaluación: pruebas con Systrace	2
15	Evaluación: pruebas con Robotium	5
16	Documentación: pruebas con Robotium	4
17	Documentación: revisión de la especificación de requisitos	2
18	Documentación: revisión de la especificación de requisitos	2
19	Documentación: revisión de la especificación de requisitos	3
20	Documentación: evaluación del entorno	3
21	Documentación: evaluación del entorno	4
22	Documentación: evaluación del entorno	6
	TOTAL HORAS	41

7.4.5. JULIO

DÍA	TAREAS	TIEMPO (en horas)
1	Documentación: Introducción	4
2	Codificación: Refinamiento de interfaces	4
3	Documentación: Juntar documentación	5
4	Documentación: Apéndices	6

5	Documentación: Correcciones	3
6	Documentación: Correcciones	10
7	Evaluación: uso de SysTrace	7
8	Documentación: Planificación	4
10	Documentación: cambios en evaluación	4
11	Documentación: Correcciones	3
12	Documentación: Correcciones	5

7.5. CONSULTAS DE EJEMPLO A LOS WEB SERVICES DE LA BBDD

En todas las consultas, al principio de la url aparece la palabra "servidor" que identifica la dirección pública o privada del servidor en el que se ubican los *web services*.

Hacer Login	
TIPO	POST
URL	servidor/mobimarket/mobimarket/api/login
DESCRIPCIÓN *Se le pasará a la consulta como parámetro la cadena name=usuario&pass="contraseña", sustituyendo usuario y contraseña por los credenciales que se quieran comprobar. Si el login es correcto se recibe un STATUS 200. Además la consulta tendrá que realizarse bajo la cabecera de tipo "application/x-www-form-urlencoded".*	
JSON EJEMPLO No se utiliza ningún JSON en esta consulta.	

Registrar Usuario	
TIPO	POST
URL	servidor/mobimarket/mobimarket/api/register
DESCRIPCIÓN *Se le pasará a la consulta como parámetro un fichero JSON con los datos del usuario que se quiere registrar en la base de datos. La consulta tendrá que realizarse bajo la cabecera de tipo "application/json" y "charset=utf-8".*	
JSON EJEMPLO { "_id" : "org.couchdb.user:nombre", "address" : "dirección", "birthday" : "fecha", "email" : "correo@email.com", "firstName" : "nombre", "lastName" : "apellido", "name" : "usuario", "password" : "contraseña", "phoneNumber" : "666777888",	

```
  "roles" : [ "mobimarket_user" ],
  "serveOnTable" : false,
  "sex" : "sexo",
  "type" : "user",
  "userCategory" : "normal",
  "zipcode" : "32321"
}
```


Consultar Supermercados	
TIPO	GET
URL	servidor/mobimarket/mobimarket/api/restaurants

DESCRIPCIÓN

La base de datos devuelve un fichero JSON con la información de los supermercados que tiene almacenados.

JSON EJEMPLO


```
[ { "_attachments" : { "image" : { "content_type" : "image/png",
          "digest" : "md5-MBMahB/43lOLNcTQ2Ck4Mg==",
          "revpos" : 8,
          "stub" : true
        } },
   "_id" : "16d9f0b7ff0ffe2f202ff55e97008ff2",
   "_rev" : "10-d97c477cd3938d41dc151871c91dde96",
   "address" : "C Uria 22",
   "bussinessHours" : "24H",
   "coordinate" : { "latitude" : 43.538200000000003,
      "longitude" : -5.6546000000000003
     },
   "created_at" : "2013-03-19T16:55:38.612Z",
   "name" : "MobiMarket Gijón",
   "phone" : "9854754578",
   "services" : [  ],
   "type" : "restaurant"
  },
```

```
{ "_attachments" : { "image" : { "content_type" : "image/png",
        "digest" : "md5-5V3dHsN8cg1EynWxXHs76A==",
        "revpos" : 3,
        "stub" : true
      } },
  "_id" : "16d9f0b7ff0ffe2f202ff55e97010150",
  "_rev" : "5-4a990ff6b1caf850d667d4580c020f6a",
  "address" : "C/ Los Arcos 55",
  "bussinessHours" : "9h - 21h",
  "coordinate" : { "latitude" : 43.368099999999998,
    "longitude" : -5.859
   },
  "created_at" : "2013-03-20T15:30:24.605Z",
  "name" : "MobiMarket Oviedo",
  "phone" : "985225026",
  "services" : [ ],
  "type" : "restaurant"
 }
]
```


Consultar Supermercado	
TIPO	GET
URL	servidor/mobimarket/mobimarket/api/restaurants/{idSupermercado}

DESCRIPCIÓN

La base de datos devuelve un fichero JSON con la información de un supermercado en concreto especificado por su _id.

JSON EJEMPLO


```
{
  "type": "restaurant",
  "coordinate":    {
    "longitude": -5.6546,
    "latitude": 43.5382
  },
```

```
  "name": "MobiMarket Gijón",
  "address": "C Uria 22",
  "phone": "9854754578",
  "bussinessHours": "24H",
  "services": [],
  "_id": "16d9f0b7ff0ffe2f202ff55e97008ff2",
  "_rev": "10-d97c477cd3938d41dc151871c91dde96",
  "_attachments": {"image":    {
    "stub": true,
    "revpos": 8,
    "digest": "md5-MBMahB/43lOLNcTQ2Ck4Mg==",
    "content_type": "image/png"
  }},
  "created_at": "2013-03-19T16:55:38.612Z"
}
```


Consultar categorías	
TIPO	GET
URL	servidor/mobimarket/mobimarket/api/categories
DESCRIPCIÓN *La base de datos devuelve un fichero JSON con la información de todas las categorías que tiene un supermercado.*	
JSON EJEMPLO	


```
[
    {
    "type": "category",
    "name": "Bebidas alcohólicas",
    "orderPosition": 0,
    "_id": "16d9f0b7ff0ffe2f202ff55e970149b4",
    "_rev": "2-9d461cd6c57b3937f82005d4036a8f1d",
    "created_at": "2013-03-21T14:54:03.602Z",
    "_attachments": {"image":       {
```

```
        "stub": true,
        "revpos": 2,
        "digest": "md5-oAriv8hmipakYzKYGovoxA==",
        "content_type": "image/png"
      }}
   },
     {
     "type": "category",
     "name": "Bebidas no alcohólicas",
     "orderPosition": 0,
     "_id": "16d9f0b7ff0ffe2f202ff55e97013bc2",
     "_rev": "3-40829f748943c8c2aa51dd80c5cd7bf5",
     "created_at": "2013-03-21T14:50:47.783Z",
     "_attachments": {"image":       {
        "stub": true,
        "revpos": 2,
        "digest": "md5-emhfIrNawOCx9jcUm4V/OQ==",
        "content_type": "image/png"
      }}
   }
]
```


Consultar productos	
TIPO	GET
URL	servidor/mobimarket/mobimarket/api/category/{categoryId}
DESCRIPCIÓN *La base de datos devuelve un fichero JSON con la información de todos los productos que tiene una categoría concreta.*	
JSON EJEMPLO [{ "itemType": "product", "type": "item",	

```
    "name": "Cerveza",
    "price": 0.47,
    "description": "Cerveza Mahou sin alcohol",
    "categoryIds": ["16d9f0b7ff0ffe2f202ff55e97013bc2"],
    "_id": "16d9f0b7ff0ffe2f202ff55e9701c57e",
    "_rev": "2-ce6c556b75f0b6f4bcae05aedcb7aae0",
    "created_at": "2013-05-07T09:47:44.596Z",
    "_attachments": {"image":       {
      "stub": true,
      "revpos": 2,
      "digest": "md5-y5BgnUEGZXT8IFlfDbWOKw==",
      "content_type": "image/png"
    }}
  },
    {
    "itemType": "product",
    "type": "item",
    "name": "Agua Mineral",
    "price": 0.33,
    "description": "50cl de agua mineral procedente de Asturias",
    "categoryIds": ["16d9f0b7ff0ffe2f202ff55e97013bc2"],
    "_id": "16d9f0b7ff0ffe2f202ff55e9701dfb1",
    "_rev": "2-6f1034429c06dbb4930b0c6426c6caa9",
    "created_at": "2013-05-07T09:50:36.814Z",
    "_attachments": {"image":       {
      "stub": true,
      "revpos": 2,
      "digest": "md5-SGgbMwsrD9zMYFhtaww+4Q==",
      "content_type": "image/png"
    }}
  }
]
```

Consultar promociones	
TIPO	GET
URL	servidor/mobimarket/mobimarket/api/promos/active

DESCRIPCIÓN

La base de datos devuelve un fichero JSON con la información de todas las promociones que están activas en los supermercados.

JSON EJEMPLO


```
[
    {
    "itemType": "promo",
    "type": "item",
    "name": "Promo Bocadillo de Jamón",
    "price": 5,
    "description": "Pan de leña de cuarto más 100g de jamón ibérico",
    "categoryIds": [],
    "from": "2013-03-20T00:00:00.000Z",
    "to": "2014-03-20T00:00:00.000Z",
    "itemsIds": [],
    "_id": "16d9f0b7ff0ffe2f202ff55e9701160f",
    "_rev": "8-82c88eb7ddd6af9a4a9ae6470c4d2cbe",
    "created_at": "2013-03-20T16:53:21.407Z",
    "_attachments": {"image":       {
      "stub": true,
      "revpos": 7,
      "digest": "md5-jm1ZroRD4QUMAMHg9eZBsg==",
      "content_type": "image/png"
    }}
  },
    {
    "itemType": "promo",
    "type": "item",
    "name": "Promo Cerveza",
    "price": 0.8,
```

```
    "description": "3x2 en Cervezas",
    "categoryIds": [],
    "from": "2013-04-29T00:00:00.000Z",
    "to": "2013-06-26T00:00:00.000Z",
    "itemsIds": [],
    "_id": "16d9f0b7ff0ffe2f202ff55e9701c4f8",
    "_rev": "4-f59c2c1435a0ac5dc6e641b6563ec480",
    "created_at": "2013-04-29T06:54:22.400Z",
    "_attachments": {"image":      {
      "stub": true,
      "revpos": 3,
      "digest": "md5-GIFDzx8ZEh8Dgw6/DTnI+w==",
      "content_type": "image/png"
    }}
  }
]
```


Consultar pedidos realizados	
TIPO	GET
URL	servidor/mobimarket/mobimarket/api/purchases

DESCRIPCIÓN

La base de datos devuelve un fichero JSON con la información de todos los pedidos que se encuentran almacenados en la persistencia.

JSON EJEMPLO


```
[
    {
    "type": "purchase",
    "purchaseItems": [     {
      "item":         {
        "itemType": "product",
        "type": "item",
        "price": 6.24,
        "_id": "16d9f0b7ff0ffe2f202ff55e9701807e"
```

```
      },
      "quantity": 1,
      "recipientUser": null
    }],
    "purchasePrice": 6.24,
    "buyer": "agus",
    "status": "in_process",
    "_id": "16d9f0b7ff0ffe2f202ff55e97018bc2",
    "_rev": "2-69aa2f12cec1490feb3c08694e28d9fd",
    "created_at": "2013-04-23T10:45:48.703Z"
  },
    {
    "type": "purchase",
    "purchaseItems": [     {
      "item":         {
        "itemType": "product",
        "type": "item",
        "price": 6.24,
        "_id": "16d9f0b7ff0ffe2f202ff55e9701807e"
      },
      "quantity": 1,
      "recipientUser": null
    }],
    "purchasePrice": 6.24,
    "buyer": "mobimarket",
    "status": "in_process",
    "_id": "16d9f0b7ff0ffe2f202ff55e970194b9",
    "_rev": "1-d878bf51de56ba027ae0ef28d17ac156",
    "created_at": "2013-04-23T12:47:59.050Z"
  }
]
```


Realizar pedido

TIPO	GET
URL	servidor/mobimarket/mobimarket/api/purchases

DESCRIPCIÓN

Se envía a la base de datos un fichero JSON con la información de un nuevo pedido para que sea almacenado. La consulta tendrá que realizarse bajo la cabecera de tipo "application/json" y "charset=utf-8".

JSON EJEMPLO

```
{
        "type" : "purchase",
        "purchaseItems" : [
        {
                "item" : {
                        "itemType" : "product",
                        "type" : "item",
                        "price" : 0.53,
                        "_id" : "16d9f0b7ff0ffe2f202ff55e97017ffc"
                },
                "quantity" : 1
                },
        {
                "item" : {
                        "itemType" : "product",
                        "type" : "item",
                        "price" : 5,
                        "_id" : "16d9f0b7ff0ffe2f202ff55e97010bac"
                },
                "quantity" : 1
        }],
        "purchasePrice" : 5.53,
        "buyer" : "agustin",
        "created_at" : "2013-04-24T10:30:26.359Z",
        "status" : "in_process"
}
```

Obtener imagen	
TIPO	GET
URL	servidor/mobimarket/mobimarket/api/{idDoc}/image
DESCRIPCIÓN *La base de datos devuelve un fichero de imagen del documento que la solicita. Es decir si queremos obtener la imagen de un producto con id 6d9f0b7ff0ffe2f202ff55e97010bac sólo se tendrá que realizar la consulta a la url que aquí se expone sustituyendo idDoc, por el identificador del producto. Será el mismo proceso para la obtención de una imagen de cualquier tipo de documento.*	
JSON EJEMPLO No se utilizan ficheros JSON de entrada ni de salida para la consulta a este Web Service.	