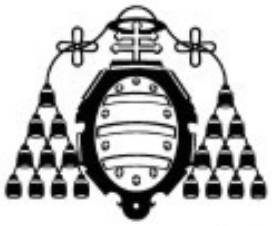
UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

"DESARROLLO DE APLICACIÓN WEB PARA EL RECONOCIMIENTO DE ENTIDADES MUSICALES"

DIRECTOR: JOSE EMILIO LABRA GAYO

AUTOR: BERNARDO MÁRTINEZ GARRIDO

Vº Bº del Director del Proyecto

Agradecimientos

Se agradece a BMAT su ayuda e información a la hora de desarrollar los aspectos técnicos relacionados con el standard CWR.

Se agrade a WESO, a Daniel Fernández Álvarez y a Cristian González García el apoyo y ayuda durante el desarrollo del proyecto.

Resumen

El presente proyecto busca desarrollar el prototipo para un sistema de conciliación de entidades musicales, las cuales pueden provenir de diversas fuentes.

Este proyecto pertenece a un proyecto mayor, desarrollado por el grupo de investigación WESO, como parte de un contrato con la empresa BMAT, y englobado en el plan Avanza. El resto del proyecto de WESO incluye el algoritmo de conciliación MERA, con el cual se integra la aplicación, haciendo uso de él durante el proceso de conciliación.

La arquitectura y diseño de la aplicación se ha elaborado teniendo en cuenta las necesidades de la empresa, que una vez tenga el producto procederá a integrarlo en su infraestructura.

Esto se debe a que BMAT pretende cubrir una necesidad con este proyecto. Esta empresa trabaja para la industria discográfica, localizando a través de diversos medios usos fraudulentos de las licencias de sus clientes. Para esto periódicamente recibe información de sus clientes acerca de que licencias poseen, por medio de ficheros de transacciones, y los contrasta contra los datos que han surgido de las búsquedas.

El problema es que los datos de estas búsquedas son datos en sucio, que rara vez coinciden con precisión o claridad con los datos de las transacciones. Y BMAT busca una solución que permita unir ambas fuentes de datos.

Para esto será necesario procesar los ficheros, y con sus datos iniciar un proceso de conciliación.

Esto se llevará a cabo a través de un cliente web, a través del cual se procesarán los ficheros de entrada y se controlará el sistema de conciliación. Permitiendo además labores de consulta y actualización de datos, y generación de informes.

Este cliente se conectará a una serie de servicios web, los cuales contendrán la mayor parte de la lógica de negocio. De esta manera se consigue un diseño modular, el cual será más fácil de integrar en la estructura del cliente.

Palabras Clave

Conciliación entidades musicales, ficheros CWR, servicios web, Python, Flask, parsing

Abstract

The aim of this project is developing a prototype for a matching system for musical entities which may come from various sources.

This project is part of a bigger one, developed by the research group WESO, and part of a contract between the group and the company BMAT, which is also under the scope of the Avanza plan. The other part of the main project includes the MERA matching algorithm, which is integrated into the application, making use of it for the matching process.

The arquitecture and design of the application have taken into account the needs of the company, which will integrate the final product into its infrastructure.

This is because BMAT wants to fix a deficiency with this project. This company works with the discographic industry, locating unauthorized uses of their clients licenses through several mediums. For this they periodically receive information from their clients about which licenses they own, through the use of transaction files, and compare them with the data which they got from the searches.

The problem is that the data they get from said searches is dirty data, which rarely equals to the data from the transactions. BMAT looks for a way to join both data sources.

For this it will be needed to parse the files, and use their data to start a matching process.

This will be done through a web client, which will be used to process the data files and handle the matching process. Additionally allowing querying and updating data, and generating reports.

The client will connect to a series of web services, which will contain most of the business logic. This allows a modular design, which will be easier to integrate in the client's systems.

Keywords

Musical entities matching, CWR files, web services, Python, Flask, parsing

Índice General

Índice de Figuras

Memoria del Proyecto

1.1 Resumen de la Motivación

El proyecto ha surgido como parte de un contrato entre el grupo WESO y la empresa BMAT, con el que esta empresa pretendía cubrir una cadencia muy específica en su infraestructura: poder vincular entre si los datos provenientes de diversas fuentes.

BMAT trabaja para clientes de la industria musical, y su labor consiste en localizar usos no autorizados de melodías y canciones de sus clientes. Para esto trabaja con dos tipos de datos, por un lados los datos en sucio generados por sus búsquedas, y por otro los datos en limpio recibidos de los clientes.

El problema es que no poseen ningún método para vincular automáticamente ambas fuentes de datos, de manera que se puedan contrastar las búsquedas con los datos de los clientes.

Por eso, se ha desarrollado una aplicación que permite recibir esos datos y vincularlos por medio de un sistema de conciliación.

Hay que tener en cuenta que este proyecto no es una labor independiente, si no que forma parte de un proyecto mayor del que se ha encargado el cliente, y que está englobado en el plan Avanza.

1.2 Resumen de Objetivos

Los principales objetivos son los siguientes:

- Procesado de ficheros de transacciones CWR
- Procesado de ficheros en sucio USO
- Generación de informes del procesamiento de ficheros
- Conciliación de los datos de ficheros CWR y USO
- Generación de informes de los resultados de conciliación
- Corrección de datos de conciliación

1.3 Resumen de Alcance del Proyecto

El proyecto forma parte de un contrato para la empresa BMAT, desarrollado por el grupo de investigación CWR, y las principales limitaciones a su alcance serán las necesidades del cliente y la organización del grupo.

Entre ellas está el echo de que la labor de conciliación la llevará a cabo un algoritmo desarrollado por WESO, el algoritmo de conciliación MERA, que deberá ser integrado al proyecto.

Otra limitación importante es el echo de que el resultado del proyecto será integrado en la infraestructura del cliente. Lo que significa que no será necesario encargarse de labores horizontales con el control de acceso y seguridad. Estos serán gestionados por el propio cliente.

Entre lo que la aplicación sí deberá realizar, los aspectos más importantes se resumen en la necesidad de procesar lo ficheros de entrada, inicialmente ficheros USO y CWR, para generar datos que el sistema de conciliación sea capaz de utilizar.

Adicionalmente, se deberán ofrecer medios para consultar, y generar informes de estos datos, tanto de los datos de conciliación como de los datos de los ficheros de entrada.

1.4 Proyecto Avanza

Este proyecto forma parte de otro proyecto mayor, englobado en el plan Avanza, con los siguientes datos:

- **Referencia:** TSI-020602-2012-195
- **Título del proyecto:** TagFlow, Sistema Avanzado de Monitoreo y Tagging de Flujos de datos multimedia
- **Entidad financiadora:** Ministerio de Industria, Turismo y Comercio, Subprograma Avanza I+D
- **Entidades participantes:** BMAT Licensing, S. L., Weso Research Group
- **Investigador responsable:** BMAT Licensing S.L.

Introducción

1.5 Justificación del Proyecto

El proyecto ha surgido como necesidad al preparar un contrato entre el grupo de investigación WESO y la empresa BMAT.

Esta empresa ha desarrollado un proyecto de mayor envergadura, englobado en el plan Avanza, y el presente proyecto es una solución a medida para conectar varios componentes del sistema final, y que será integrado a este tras a su termino.

Por esto se ha optado por una solución que sea lo más ligera posible, cuyo objetivo general es integrar una serie de pasos (recepción de ficheros, procesamiento de estos, conciliación con otros datos, etc), utilizando una arquitectura modular y fácil de re-adaptar, que no se encargue de labores horizontales, como el sistema de seguridad, ya que estas, una vez se finalice el proyecto, serán aplicada por el el sistema del cliente, al que se integrará la aplicación.

Para asegurar esta modularidad el proyecto se basará en servicios web, que será imprescindible crearlos desde cero. Esto es porque las dos labores principales de las que se encargarán, la conciliación y el procesamiento de ficheros, no ofrecen otra alternativa.

El sistema de conciliación deberá usar el algoritmo MERA, ya que este se ha desarrollado paralelamente a este proyecto, y es un requisito del contrato, y los ficheros de entrada CWR y USO requieren de un parser específico, sobre todo los ficheros CWR, ya que a pesar de seguir un standard no existen librerías públicas para procesarlos.

1.6 Objetivos del Proyecto

1.6.1 Lectura y procesamiento de ficheros CWR

Es necesario leer ficheros CWR (Common Works Registrations) y procesar su contenido. Teniendo en cuenta que estos ficheros aunque siguen un standard tienden a ser heterogéneos, ya que son comunes las variantes en su implementación y los errores en su uso.

1.6.2 Lectura y procesamiento de ficheros USO

Los ficheros USO contienen datos sucios provenientes de numerosas fuentes y de contenido impredecible, aunque seguirán una estructura de DSV delimitados por tabulador.

Es necesario leerlos y procesar su contenido.

1.6.3 Conciliación entre datos de ficheros CWR y USO

Los datos recibidos han de ser contrastados como parte de un proceso de conciliación. Para esto es imprescindible usar el algoritmo MERA desarrollado por WESO.

Esta conciliación generará un informe, y será posible corregir sus resultados.

1.6.4 Generación de informes

Será posible generar informes tanto del proceso de conciliación, como del procesamiento de ficheros.

1.6.5 Histórico de datos

Se han de almacenar los datos de los ficheros usados, y que conciliaciones se han realizado con ellos.

1.6.6 Unificable e integrable

El sistema se ha de poder integrar en la infraestructura del cliente, ya que este lo quiere para unificar diferentes fuentes de datos.

1.7 Estudio de la Situación Actual

La búsqueda de alternativas está limitada por el hecho de que hay que usar el algoritmo de conciliación creado por WESO, por lo que no es posible buscar alternativas para este aspecto del proyecto.

El resto de aspectos en su mayor parte sirven para integrar los distintos componentes de la aplicación, y son específicos para el problema tratado. Pero existe un punto donde se pueden buscar alternativas, y este es el procesamiento de ficheros.

Los ficheros CWR siguen un standard usado por numerosas empresas, y se sabe que existen aplicaciones para su procesamiento y gestión. Pero existe un gran problema a la hora de compararlos con la solución dada por este proyecto, y es que no son accesibles.

Las empresas que manejan estos ficheros usan sus propias aplicaciones internas, que no comparten, como mucho ofrecen, y cobran, servicios basados en ellas. Esto hace que en la práctica no existan alternativas al procesamiento de ficheros CWR.

Aspectos Teóricos

1.8 Conceptos

1.8.1 Conciliación/Matching

Comparación y búsqueda de coincidencias entre dos conjuntos de datos.

Se utilizará para contrastar los datos de los ficheros de entrada, de manera que se localicen coincidencias entre ellos.

1.8.2 Procesamiento de ficheros / Parsing

Procesamiento de ficheros a partir de una serie de reglas léxicas y gramaticales. También puede considerarse como un método para transformar una estructura de datos en otra.

Se utilizará para procesar los ficheros CWR, poblando un grafo a partir de este.

1.8.3 DSL

Domain Specific Language. Un lenguaje creado para trabajar con un problema en concreto, y pensado para poder definirlo y hacer referencia a sus componentes con claridad.

En el caso del proyecto se utiliza para la configuración del parser.

1.9 Herramientas

1.9.1 Python

Python es un lenguaje de programación dinámico, que resulta perfecto para la creación rápida de prototipos.

1.9.2 Flask

El framework Flask permite crear aplicaciones web ligeras, incluyendo tanto el interfaz como, por medio de extensiones, servicios web.

1.9.3 Pyparsing

Pyparsing permite generar un parser basado puramente en objetos. Con este se procesan los ficheros CWR, y el DSL usado para la configuración de este parser.

Planificación del Proyecto y Resumen de Presupuestos

1.10 Planificación

El proyecto ha comenzado el 23 de Enero, y se espera que termine a finales de Julio. Dando un total de seis meses.

Durante este tiempo se han dedicado cuatro horas diarias al proyecto, de Lunes a Viernes. Lo que da un total de veinte horas semanales.

El trabajo del proyecto se ha dividido en cinco etapas: análisis, diseño, desarrollo, despliegue y corrección.

Toda la planificación está más detallada en el fichero adjunto de planificación.

1.10.1 General

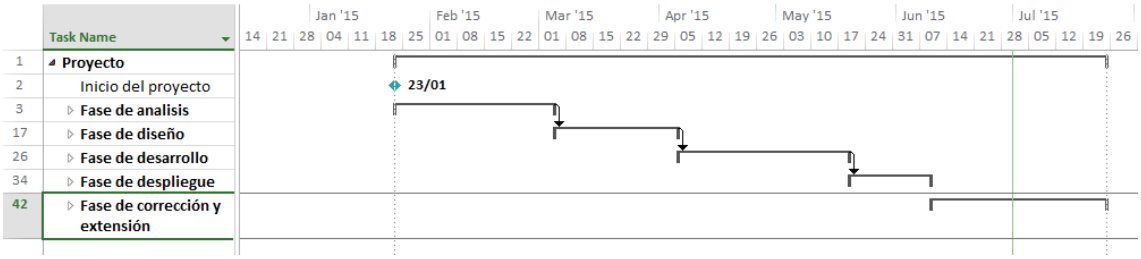


Figura 9.1. Planificación general del proyecto

1.10.2 Fase de Análisis

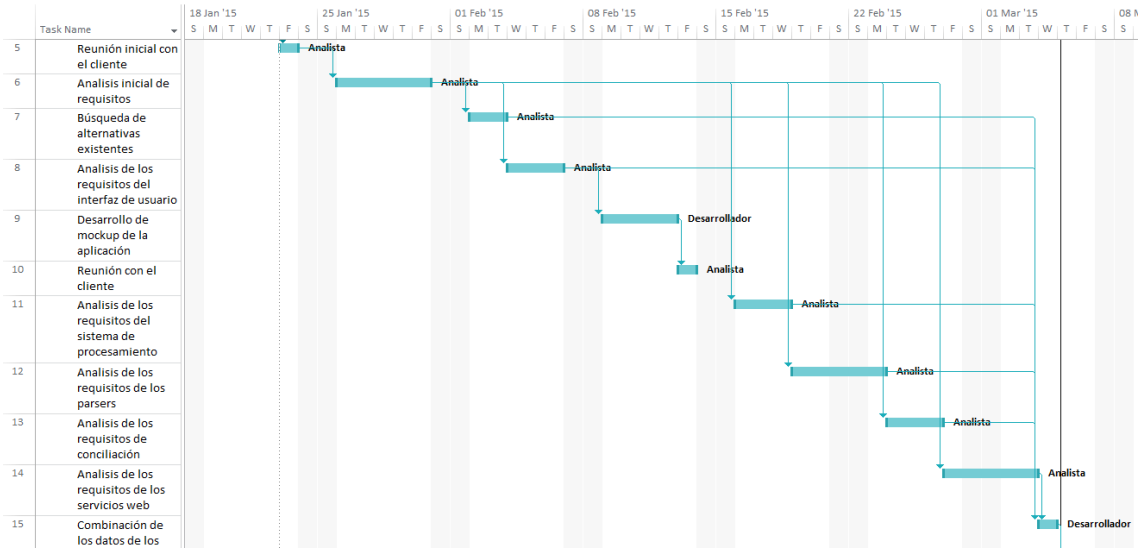


Figura 9.2. Planificación de la fase de análisis

1.10.3 Fase de Diseño

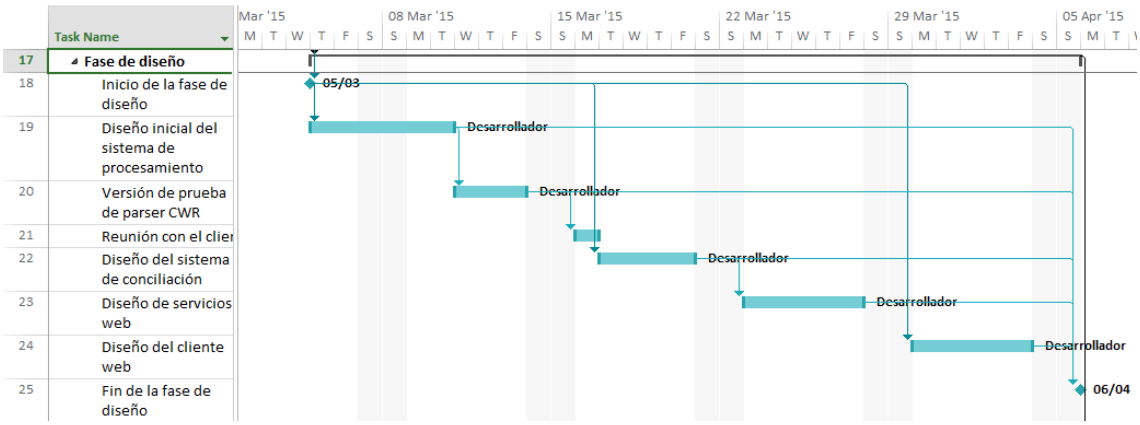


Figura 9.3. Planificación de la fase de diseño

1.10.4 Fase de Desarrollo

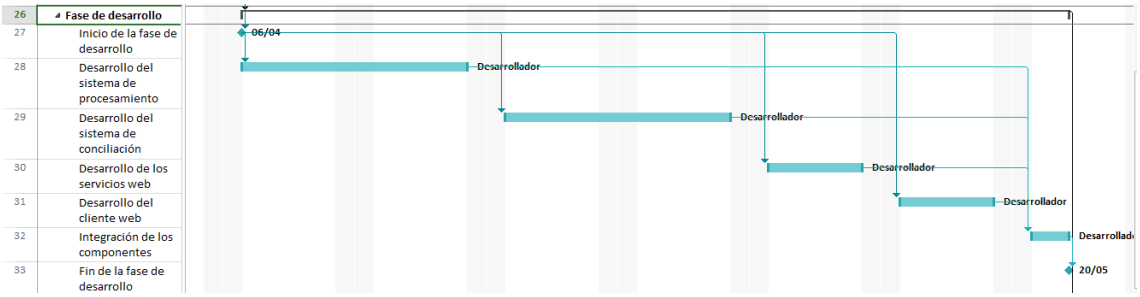


Figura 9.4. Planificación de la fase de desarrollo

1.10.5 Fase de Despliegue

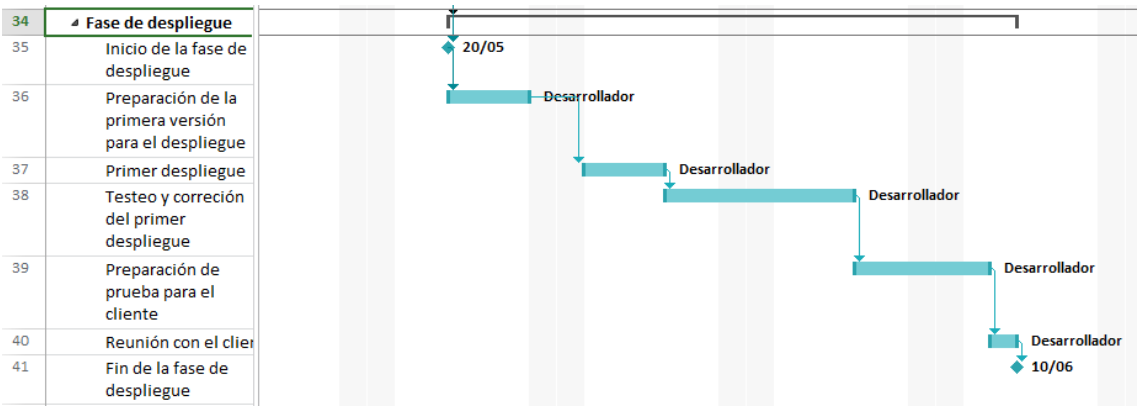


Figura 9.5. Planificación de la fase de despliegue

1.10.6 Fase de Corrección

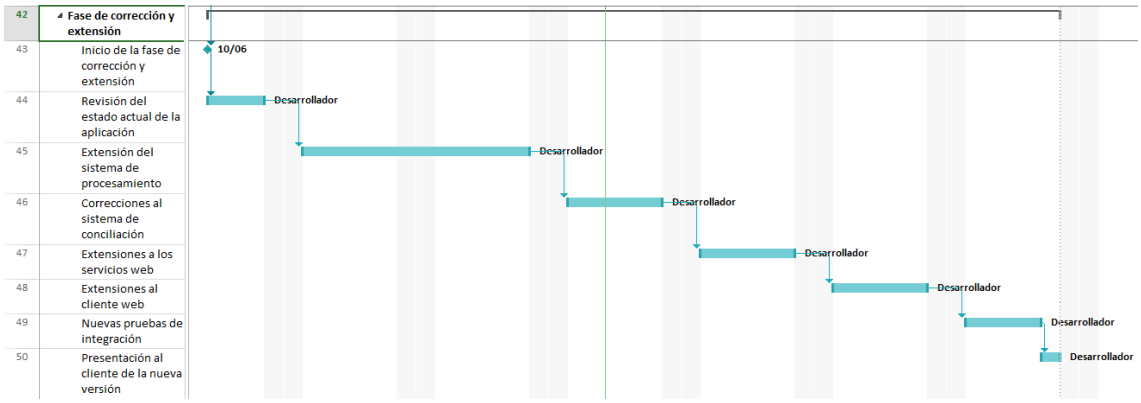


Figura 9.6. Planificación de la fase de corrección

1.11 Resumen del Presupuesto

El proyecto tiene una duración de cinco meses, con cuatro horas de trabajo diarias, dando una duración total de seiscientas horas de trabajo.

Meses	Trabajo diario	Horas totales
5	4h/día	600h

Recurso	Coste
Analista	50€/h
Desarrollador	30€/h

El desglose de coste es el siguiente:

Concepto	Cantidad	Total
Recursos humanos		
Analisis	96h	4.800,00 €
Desarrollo	424h	12.720,00 €
Otros recursos		
Costes de infraestructura		425,00 €
Alquiler oficina		3.200,00 €
	Subtotal	21.145,00 €
	Total (IVA 21%)	4.440,45 €
	Total	25.585,45 €

Análisis

1.12 Definición del Sistema

1.12.1 Determinación del Alcance del Sistema

El proyecto busca crear un sistema que integre la lectura de datos a partir de ficheros con un sistema de conciliación de datos. Esto se ha de realizar por medio de un sistema web modular, que permita a un usuario subir los datos y consultar los resultados del proceso.

Este sistema ha de ser modular y ligero, ya que posteriormente a la finalización del proyecto será adaptado por el cliente a sus sistemas.

Además, el servicio de conciliación ha de utilizar forzosamente el algoritmo MERA, desarrollado paralelamente a este proyecto.

Por tanto, el proyecto se deberá encargar de cuatro tareas principales:

- Cliente web
- Servicios web y su integración
- Sistema de conciliación, utilizando el algoritmo MERA
- Sistema de procesamiento de ficheros de entrada

Los servicios web englobarán el sistema de conciliación y procesamiento, y el cliente web se comunicará con estos por medio de mensajes REST.

Todo esto se realizará utilizando el lenguaje Python, y para los aspectos web se hará uso del framework Flask.

1.13 Requisitos del Sistema

1.13.1 Obtención de los Requisitos del Sistema

1.13.1.1 Requisitos del sistema de procesamiento de ficheros

1.13.1.1.1 Requisitos generales de ficheros

Código	Nombre Requisito	Descripción del Requisito
R1.1	Procesamiento ficheros CWR	Debe ser posible procesar y representar en la aplicación ficheros CWR
R1.2	Procesamiento ficheros USO	Debe ser posible procesar y representar en la aplicación ficheros CWR
R1.3	Consultar listado de ficheros	Se deben poder consultar los datos de los ficheros procesados

1.13.1.1.2 Requisitos de ficheros CWR

Código	Nombre Requisito	Descripción del Requisito
R2.1	Validación automática de ficheros CWR	Se debe validar el contenido de los ficheros CWR durante su procesamiento, comprobando que se ajustan a la especificación
R2.2	Validación manual de ficheros CWR	Se debe permitir validar manualmente el contenido del fichero CWR mediante un interfaz gráfico
R2.3	Generación de informes a partir de ficheros CWR	Se debe poder generar informes a partir de los datos de los ficheros CWR
R2.4	Generación de ficheros acknowledgment a partir de ficheros CWR	Se debe poder generar ficheros acknowledgement (parte del estandard CWR) a partir de los ficheros CWR
R2.5	Modelado de ficheros CWR	Ha de ofrecerse un modelo que represente los ficheros CWR

1.13.1.1.3 Requisitos de ficheros USO

Código	Nombre Requisito	Descripción del Requisito
R3.1	Modelado de ficheros USO	Ha de ofrecerse un modelo que represente los ficheros USO

1.13.1.2 Requisitos de servicios web

Código	Nombre Requisito	Descripción del Requisito
R4.1	Servicio web de procesamiento CWR	Todas las labores vinculadas a los ficheros CWR han de estar en un servicio web exclusivo
R4.2	Servicio web de conciliación	Todas las labores vinculadas al proceso de conciliación han de estar en un servicio web exclusivo
R4.3	Servicio web administador	Ha de existir un servicio web que centralice el uso del resto de servicios, y que será usado por el cliente

1.13.1.3 Requisitos de conciliación

Código	Nombre Requisito	Descripción del Requisito
R5.1	Algoritmo de conciliación	Se ha de utilizar el algoritmo de conciliación MERA para el proceso de conciliación
R5.2	Realimentación del algoritmo	Los resultados del algoritmo han de servir para enriquecer y mejorar futuras consultas de conciliación
R5.3	Corrección de resultados	Se ha de poder corregir los resultados de la conciliación, antes de usarlos en la realimentación, de manera que no se repitan los errores
R5.4	Generación de informes de conciliación	Se han de generar informes de los resultados del proceso de conciliación

1.13.1.4 Requisitos de cliente web

Código	Nombre Requisito	Descripción del Requisito
R6.1	Cliente web	El usuario ha de acceder a la aplicación por medio de un servicio web
R6.2	Uso de servicios web	El cliente web ha de estar con la menor cantidad de lógica posible, delegando sus labores a los servicios web

1.13.2 Identificación de Actores del Sistema

1.13.2.1 Usuario

El usuario accederá al sistema por medio del interfaz web, y se encargará de enviar datos para el servicio de conciliación, e introducir datos.

1.13.3 Especificación de Casos de Uso

A continuación se muestran los casos de uso generales:

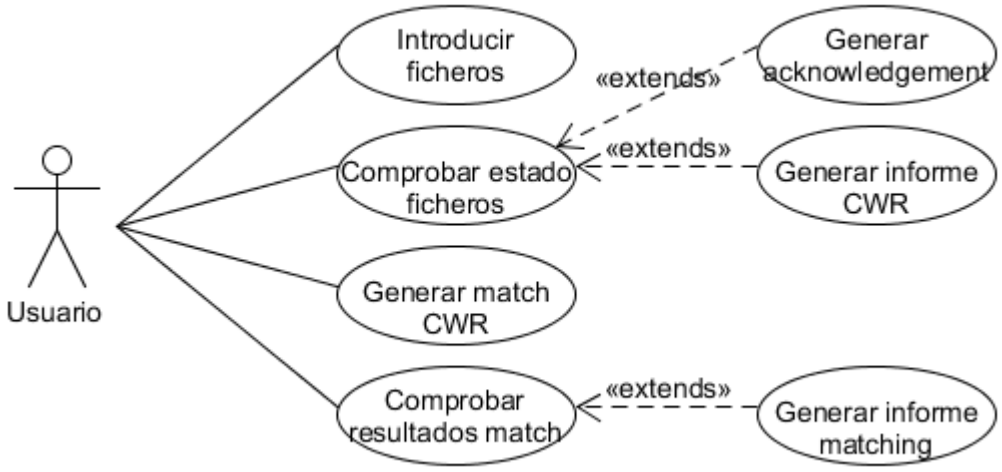


Figura 12.1. Diagrama de casos de uso general

Los diferentes casos de uso generales se detallan más abajo.

1.13.3.1 Introducir ficheros de datos

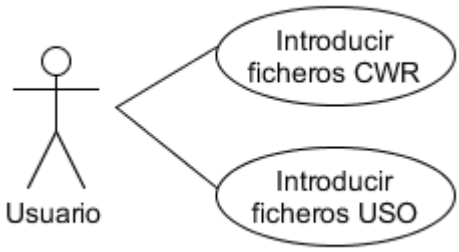


Figura 12.2. Diagrama de casos de uso de introducir ficheros de datos

Nombre del Caso de Uso
Introducir ficheros CWR
Descripción
A través del cliente web, la aplicación recibirá uno o varios ficheros CWR, que procesará y almacenará para su posterior uso.

Nombre del Caso de Uso
Introducir ficheros USO
Descripción
A través del cliente web, la aplicación recibirá uno o varios ficheros USO, que procesará y almacenará para su posterior uso.

1.13.3.2 Consultar datos de ficheros CWR

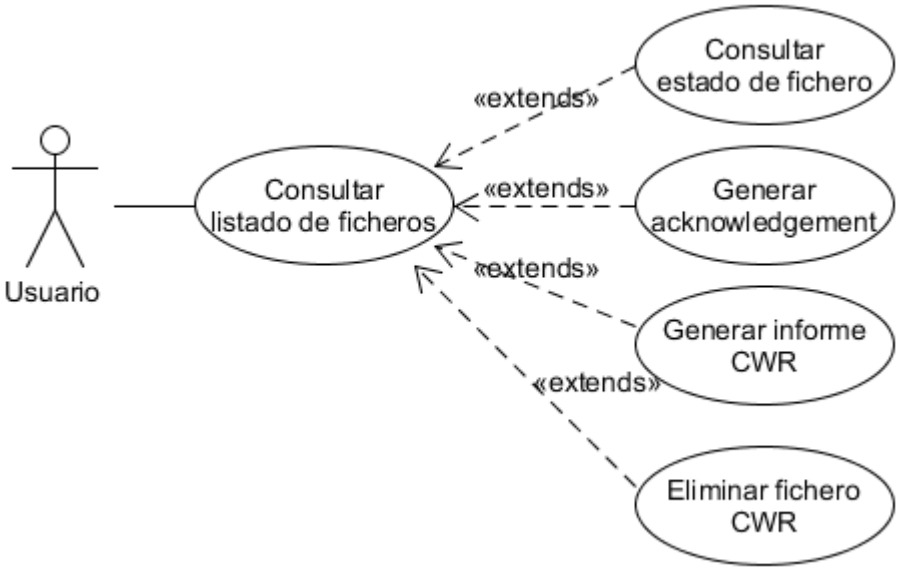


Figura 12.3. Diagrama de casos de uso de consultar datos de ficheros CWR

Nombre del Caso de Uso
Consultar listado de ficheros
Descripción
El usuario consulta la lista de ficheros que han sido procesados. Pudiendo acceder a diversos detalles sobre estos.

Nombre del Caso de Uso
Consultar estado de fichero
Descripción
El usuario consulta el estado actual de un fichero, pudiendo comprobar si hubo fallos durante su procesamiento, o si está actualmente siendo procesado.

Nombre del Caso de Uso
Generar acknowledgement
Descripción
El usuario genera un informe acknowledgement (parte de la especificación CWR) a partir de los datos del fichero

Nombre del Caso de Uso

Generar informe CWR	
Descripción	
El usuario genera un informe con los datos del fichero CWR	

Nombre del Caso de Uso
Eliminar fichero CWR
Descripción
El usuario elimina un fichero CWR y todos los datos vinculados a este

1.13.3.3 Generar conciliación CWR

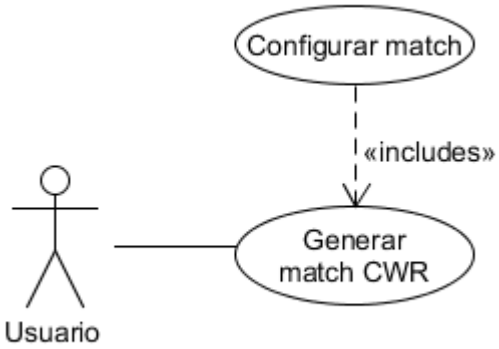


Figura 12.4. Diagrama de casos de uso de generar conciliación CWR

Nombre del Caso de Uso
Generar conciliación CWR
Descripción
El usuario pide a la aplicación que envié un fichero CWR al proceso de conciliación.

Nombre del Caso de Uso
Configurar conciliación
Descripción
El usuario configura el proceso de conciliación, pudiendo usar una configuración por defecto

1.13.3.4 Comprobar resultados conciliación

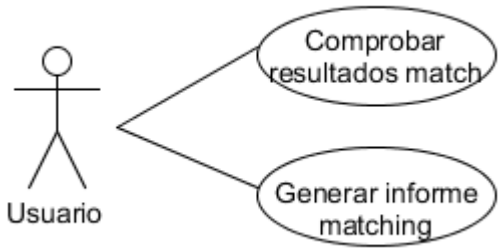


Figura 12.5. Diagrama de casos de uso de comprobar resultados de conciliación

Nombre del Caso de Uso
Comprobar resultados conciliación
Descripción
El usuario pide a la aplicación que le muestre los resultados de la conciliación

Nombre del Caso de Uso
Generar informe conciliación
Descripción
El usuario pide el informe sobre el resultado de un proceso de conciliación

1.14 Identificación de los Subsistemas en la Fase de Análisis

1.14.1 Descripción de los Subsistemas

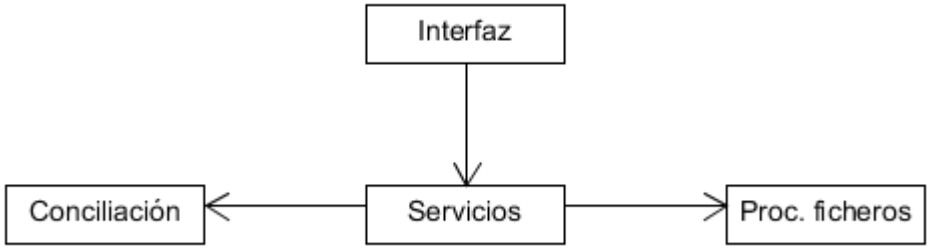


Figura 13.1. Diagrama de los subsistemas

Los subsistemas existentes durante la fase de analisis son:

- Subsistema de interfaz
- Subsistema de conciliación
- Subsistema de procesamiento de ficheros
- Subsistema de servicios

1.14.1.1 Subsistema de interfaz (cliente web)

El subsistema de interfaz proporciona al usuario un método para acceder a la aplicación y sus funcionalidades, comunicandose con el subsistema de servicios.

Esta interfaz será el cliente web

1.14.1.2 Subsistema de administración de servicios

Este subsistema se encarga de integrar los diferentes subsistemas, y ofrecer funcionalidades extra como el almacenamiento de datos, o la realización de consultas sobre estos.

El subsistema estará integrado en un servicio web.

1.14.1.3 Subsistema de conciliación

El subsistema de conciliación recibe datos con los que realizar la conciliación, y permite acceder a estos resultados.

Este subsistema estará integrado en un servicio web.

1.14.1.4 Subsistema de procesamiento de ficheros

El subsistema de procesamiento de ficheros ofrece modelos y parsers para gestionar estos.

El grupo más importante será el modelo y parser del fichero CWR.

Este subsistema estará integrado en un servicio web.

1.14.2 Descripción de los Interfaces entre Subsistemas

Los susbsistemas están incrustados en una serie de servicios web, menos la interfaz, que forma parte del cliente web. Debido a esto será posible tenerlos desplegados en localizaciones independientes.

La comunicación entre ellos, aun en el caso de que se encontrasen localmente en el mismo servidor, se realizará por medio de mensajes JSON.

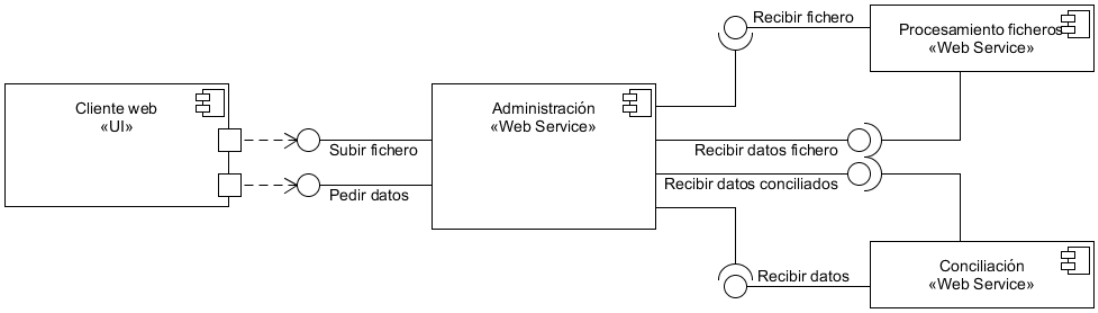


Figura 13.2. Diagrama de comunicaciones de los subsistemas

1.15 Diagrama de Clases Preliminar del Análisis

1.15.1 Diagramas de Clases del Subsistema de interfaz

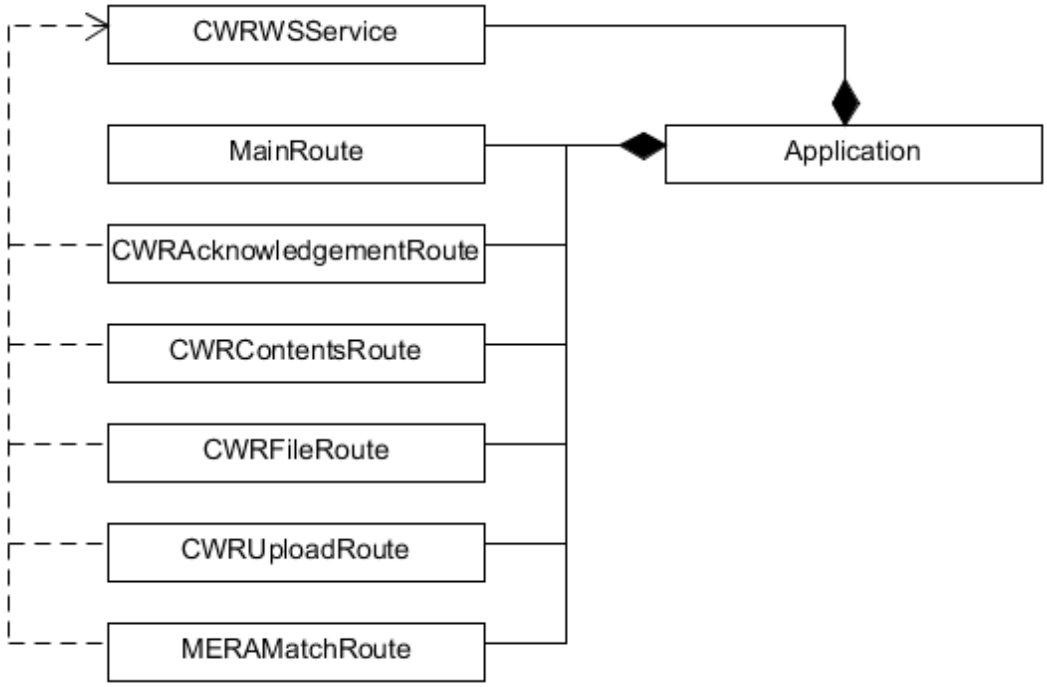


Figura 14.1. Diagrama de clases del subsistema de interfaz

El cliente web está compuesto de una serie de rutas. Estas son clases que gestionan las distintas URLs internas del cliente.

Además, posee un servicio que sirve de intermediario con los clientes web. Este forma parte de la aplicación principal, y estará accesible a las clases ruta, que harán uso de él.

Nombre de la Clase	
Application	
Descripción	
Clase ejecutable, encargada de poner la configuración inicial y ejecutar el cliente web.	
Responsabilidades	
Configuración del cliente. Ejecución del cliente. Integración de los diferentes componentes del cliente.	

Nombre de la Clase	

MainRoute	
Descripción	
Gestor de las URLs internas generales, tales como el índice principal o la página de información.	
Responsabilidades	
Gestión de URLs internas generales. Creación y gestión de interfaz web para las URLs internas generales	

Nombre de la Clase	
CWRAcknowledgementRoute	
Descripción	
Gestor de las URLs dedicadas a la generación de informes acknowledgement	
Responsabilidades	
Gestión de URLs para la generación de informes acknowledgement. Creación y gestión de interfaz web para las URLs de generación de informes acknowledgement	

Nombre de la Clase	
CWRContentsRoute	
Descripción	
Gestor de las URLs dedicadas a la muestra de los datos de ficheros CWR	
Responsabilidades	
Gestión de URLs para la muestra de los datos de ficheros CWR Creación y gestión de interfaz web para las URLs de muestra de los datos de ficheros CWR	

Nombre de la Clase	
CWRFileRoute	
Descripción	
Gestor de las URLs dedicadas al listado de ficheros CWR	
Responsabilidades	
Gestión de URLs para el listado de ficheros CWR Creación y gestión de interfaz web para las URLs de listado de ficheros CWR	

Nombre de la Clase	
CWRUploadRoute	
Descripción	
Gestor de las URLs dedicadas a la recepción de ficheros	
Responsabilidades	
Gestión de URLs para la recepción de ficheros Creación y gestión de interfaz web para las URLs de recepción de ficheros	

Nombre de la Clase	
MERAMatchRoute	
Descripción	
Gestor de las URLs dedicadas al proceso de conciliación con MERA	
Responsabilidades	
Gestión de URLs para el proceso de conciliación con MERA Creación y gestión de interfaz web para las URLs del proceso de conciliación con MERA	

Nombre de la Clase	
CWRWSService	
Descripción	
Intermedia entre el cliente web y los diferentes servicios web	
Responsabilidades	
Adaptar datos y peticiones entre el cliente y el servicio web Ocultar peticiones a los servicios web	

1.15.2 Diagramas de Clases del Subsistema de conciliación

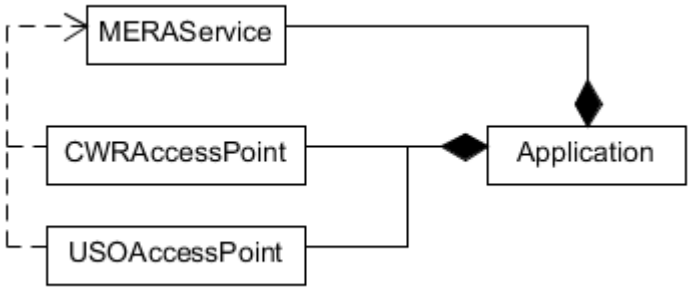


Figura 14.2. Diagrama de clases del subsistema de conciliación

El subsistema de conciliación está preparado para ser parte de un servicio web, y por eso se compone de una aplicación central, con una serie de clases gestionando los puntos de acceso REST.

Un servicio sirve a la vez para envolver al algoritmo de conciliación, y para gestionar el envio del resultado al sistema de integración.

Nombre de la Clase
Application
Descripción
Clase ejecutable, encargada de poner la configuración inicial y ejecutar el servicio web.
Responsabilidades
Configuración del servicio web. Ejecución del servicio web. Integración de los diferentes componentes del cliente.

Nombre de la Clase
CWRAccessPoint
Descripción
Gestor del punto de acceso REST para datos procesados de ficheros CWR
Responsabilidades
Procesar mensajes REST. Recibir datos procesados de ficheros CWR. Enviar los datos procesados al servicio de conciliación.

Nombre de la Clase
USOAccessPoint

Descripción
Gestor del punto de acceso REST para datos procesados de ficheros USO
Responsabilidades
Procesar mensajes REST. Recibir ficheros USO. Enviar los datos procesados al servicio de conciliación.

Nombre de la Clase
MERAService
Descripción
Gestiona el proceso de conciliación
Responsabilidades
Adaptar datos y peticiones al sistema de conciliación Enviar los resultados al sistema integrador

Nombre de la Clase
MERA
Descripción
Contiene el algoritmo de conciliación
Responsabilidades
Realizar la labor de conciliación

1.15.3 Diagramas de Clases del Subsistema de procesamiento de ficheros CWR

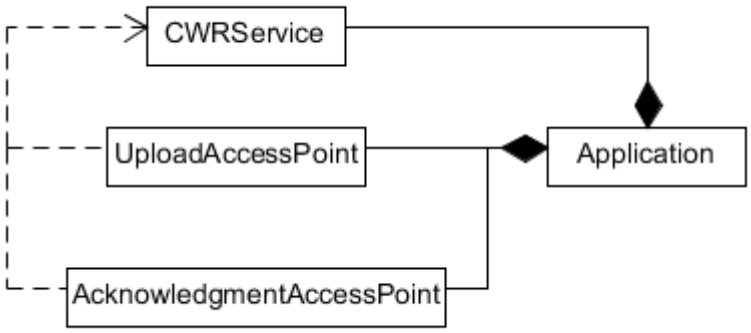


Figura 14.3. Diagrama de clases del subsistema de procesamiento de ficheros CWR

El subsistema de procesamiento de ficheros CWR recibe un fichero y lo transforma en una estructura de datos que el resto de sistemas pueda procesar. Esta estructura será generada por medio del subsistema de API para ficheros CWR.

Además ofrece sistemas para generar los ficheros acknowledgement.

Nombre de la Clase
Application
Descripción
Clase ejecutable, encargada de poner la configuración inicial y ejecutar el servicio web.
Responsabilidades
Configuración del servicio web. Ejecución del servicio web. Integración de los diferentes componentes del cliente.

Nombre de la Clase
UploadAccessPoint
Descripción
Gestor del punto de acceso REST para ficheros CWR
Responsabilidades
Procesar mensajes REST. Recibir ficheros CWR y procesarlos. Enviar los datos procesados al servicio de conciliación.

Nombre de la Clase

AcknowledgementAccessPoint	
Descripción	
Gestor del punto de acceso REST para informes acknowledgement de ficheros CWR	
Responsabilidades	
Procesar mensajes REST. Recibir datos de ficheros CWR. Generar acknowledgement. Enviar los datos del informe al servicio de conciliación.	

Nombre de la Clase	
CWRService	
Descripción	
Servicio encargado de procesar ficheros CWR	
Responsabilidades	
Envolver y ocultar el API para ficheros CWR Procesar ficheros CWR Enviar resultados al sistema integrador	

1.15.4 Diagramas de Clases del Subsistema de procesamiento de ficheros USO

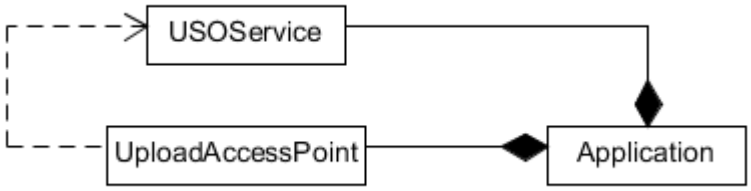


Figura 14.4. Diagrama de clases del subsistema de de procesamiento de ficheros USO

Nombre de la Clase
Application
Descripción
Clase ejecutable, encargada de poner la configuración inicial y ejecutar el servicio web.
Responsabilidades
Configuración del servicio web. Ejecución del servicio web. Integración de los diferentes componentes del cliente.

Nombre de la Clase
UploadAccessPoint
Descripción
Gestor del punto de acceso REST para ficheros USO
Responsabilidades
Procesar mensajes REST. Recibir ficheros USO y procesarlos. Enviar los datos procesados al servicio de conciliación.

Nombre de la Clase
USOService
Descripción
Servicio encargado de procesar ficheros USO
Responsabilidades
Envolver y ocultar el API para ficheros USO Procesar ficheros USO Enviar resultados al sistema integrador

1.15.5 Diagramas de Clases del Subsistema integrador

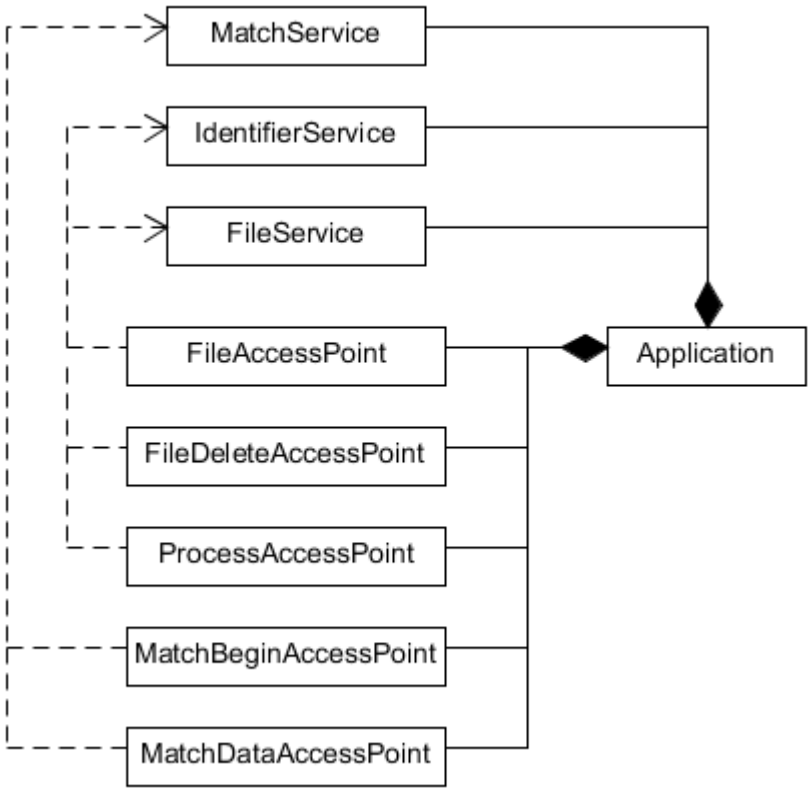


Figura 14.5. Diagrama de clases del subsistema integrador

El subsistema integrador se encarga de dar un punto de acceso al cliente web, a través del cual se comunica con el resto de servicios web.

Nombre de la Clase	
Application	
Descripción	
Clase ejecutable, encargada de poner la configuración inicial y ejecutar el servicio web.	
Responsabilidades	
Configuración del servicio web. Ejecución del servicio web. Integración de los diferentes componentes del cliente.	

Nombre de la Clase	
FileAccessPoint	
Descripción	

Gestor del punto de acceso REST para ficheros
Responsabilidades
Procesar mensajes REST. Ofrecer listados de datos de ficheros CWR.

Nombre de la Clase
FileDeleteAccessPoint
Descripción
Gestor del punto de acceso REST eliminación de ficheros
Responsabilidades
Procesar mensajes REST. Eliminar ficheros CWR.

Nombre de la Clase
ProcessAccessPoint
Descripción
Gestor del punto de acceso REST para procesamiento de ficheros
Responsabilidades
Procesar mensajes REST. Recibir ficheros CWR y USO y reenviarlos para su procesamiento. Recibir resultados de procesar ficheros CWR Recibir resultados de conciliación

Nombre de la Clase
MatchAccessPoint
Descripción
Gestor del punto de acceso REST para conciliación
Responsabilidades
Procesar mensajes REST. Recibir ficheros CWR y USO y reenviarlos al sistema de conciliación Recibir resultados de conciliaciones

Nombre de la Clase
MatchBeginService
Descripción
Servicio para inicial el proceso de conciliación
Responsabilidades
Ocultar conexiones con el servicio web de conciliación Enviar peticiones de conciliación al servicio de conciliación

Nombre de la Clase	
MatchDataService	
Descripción	
Servicio para recibir conexiones del sistema de conciliación	
Responsabilidades	
Ocultar conexiones con el servicio web de conciliación Recibir resultados del proceso de conciliación	

Nombre de la Clase	
IdentifierService	
Descripción	
Servicio para generar identificadores para los ficheros usados	
Responsabilidades	
Identificar univocamente los ficheros procesados	

Nombre de la Clase	
FileService	
Descripción	
Servicio para gestionar los datos y listados de ficheros	
Responsabilidades	
Ofrecer el listado de ficheros procesados Ofrecer datos de ficheros concretos	

1.15.6 Diagramas de Clases del Subsistema de Procesamiento de Ficheros

1.15.6.1 Subsistema de API para ficheros CWR

El API para ficheros CWR se compone de dos componentes principales: un modelo que representa la estructura interna del fichero, y un conjunto de parsers.

El modelo se puede subdividir en diferentes secciones, para facilitar su comprensión.

1.15.6.1.1 Modelo transmisión

Una transmisión posee una transacción de encabezado, una transacción de cola y una serie de grupos, que se guardan en listas de transacciones, todas las cuales comienzan con un encabezado de grupo, y terminan en una cola de grupo.

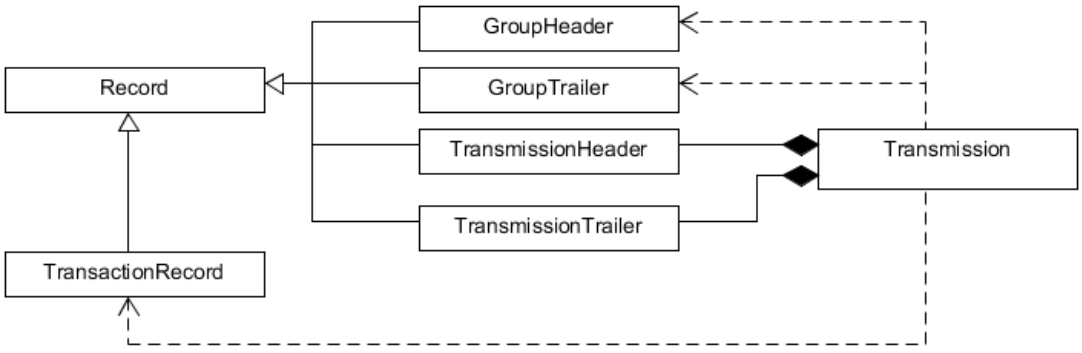


Figura 14.6. Diagrama de clases del modelo de transmisión de fichero CWR

1.15.6.1.2 Modelo de fichero

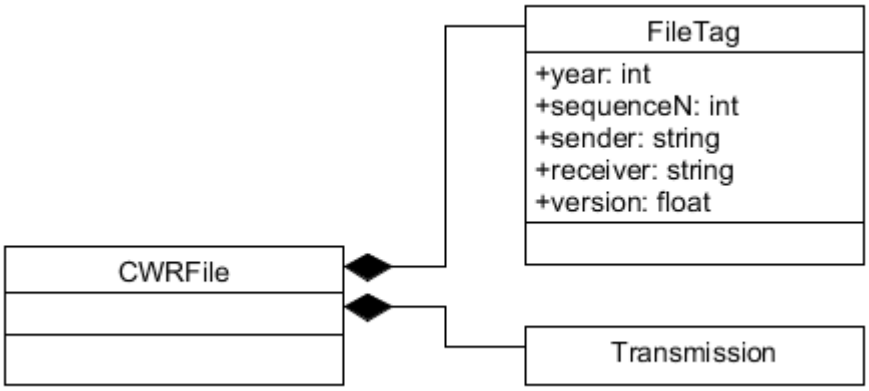


Figura 14.7. Diagrama de clases del modelo de fichero CWR

1.15.6.1.3 Modelo de registro

Existen en total alrededor de una treintena de registros diferentes. Por lo que no se va a entrar en detalle. Todos ellos sirven para almacenar los datos en cada registro del fichero, y se han de hacer a medida para cubrir cada caso.

Los registros son todas aquellas lineas del fichero que no sirven como datos de control.

Como ejemplo, se muestra el diagrama de clases para los registros de las transacciones de acuerdos.

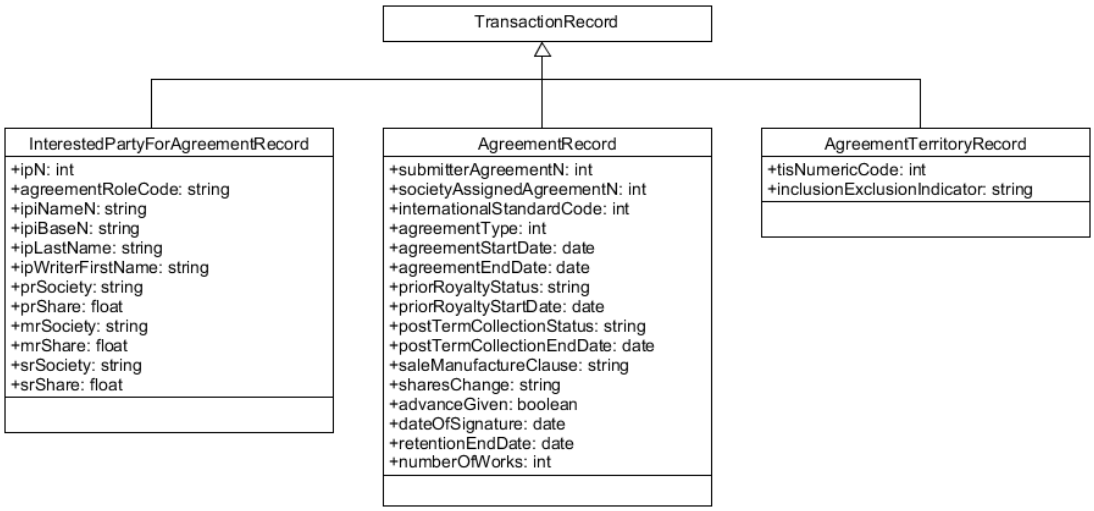


Figura 14.8. Diagrama de clases del modelo de registro del fichero CWR

1.16 Análisis de Casos de Uso y Escenarios

1.16.1 Caso de Uso: Introducir ficheros de datos

Figura 15.1. Diagrama de robustez de introducir ficheros de datos

Introduccir ficheros de datos	
Precondiciones	No hay
Poscondiciones	Se ha introducido un nuevo fichero en el sistema para ser procesado
Actores	Usuario
Descripción	1. El usuario accede a la sección de subida de ficheros y elige el tipo de fichero a añadir 2. El usuario elige los ficheros que serán añadidos 3. El usuario los envía y puede consultar su estado
Variaciones (escenarios secundarios)	
Excepciones	• **El fichero contiene errores y no se puede procesar.** En la pantalla de estado se avisará de esto, pero no afectará directamente al proceso.
Notas	

1.16.2 Caso de Uso: Consultar Datos de Fichero CWR

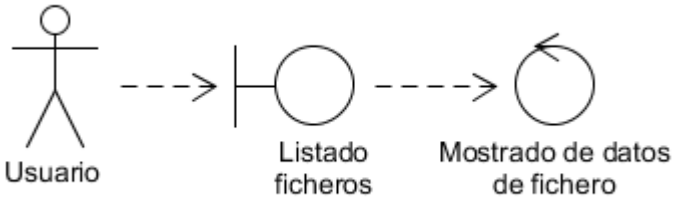


Figura 15.2. Diagrama de robustez de datos de ficheros CWR

Consultar Datos de Fichero CWR	
Precondiciones	Han de haberse introducido ficheros en el sistema para su procesamiento
Poscondiciones	Se ha consultado el estado de los ficheros
Actores	Usuario
Descripción	1. El usuario accede a la sección de listado de ficheros 2. Consulta el estado actual de los ficheros • Si el fichero ha sido procesado, se puede consultar su contenido • También es posible generar un informe del fichero
Variaciones (escenarios secundarios)	Si no se ha procesado el fichero no será posible consultar sus datos ni generar el informe. Esto se le indicará al usuario en la misma pantalla.
Excepciones	
Notas	

1.16.3 Caso de Uso: Generar Conciliación de Fichero CWR

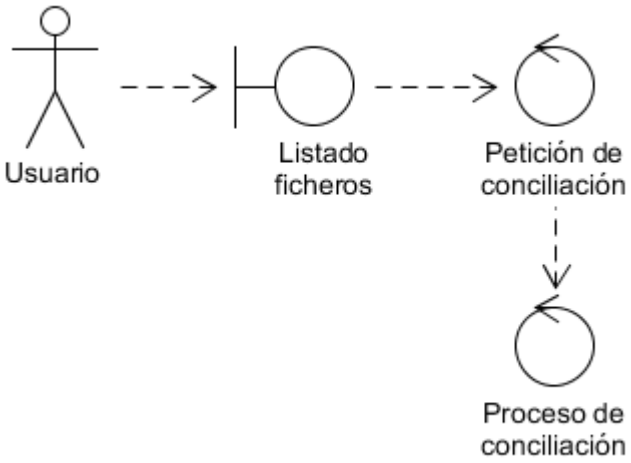


Figura 15.3. Diagrama de robustez de generar conciliación de ficheros CWR

Generar Conciliación de Fichero CWR	
Precondiciones	Ha de existir al menos un fichero correctamente procesado en el sistema.
Poscondiciones	Se ha consultado el estado de los ficheros
Actores	Usuario
Descripción	1. El usuario accede a la sección de datos de ficheros 2. El usuario elige un fichero 3. El usuario configura las opciones de conciliación 4. El usuario envía los datos
Variaciones (escenarios secundarios)	Si el fichero ya ha pasado por el proceso de conciliación este comienza de nuevo.
Excepciones	
Notas	Hasta que el proceso de conciliación termine, no será posible reiniciarlo.

1.16.4 Caso de Uso: Comprobar Resultados Conciliación

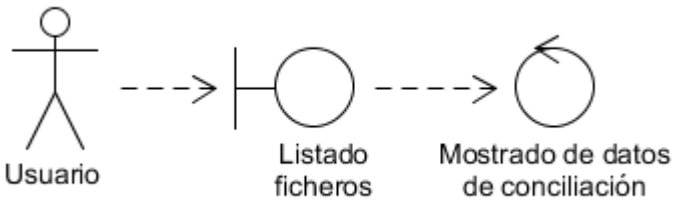


Figura 15.4. Diagrama de robustez de comprobar resultados de conciliación

Comprobar Resultados Conciliación	
Precondiciones	Ha de haberse finalizado el proceso de conciliación para por lo menos uno de los ficheros
Poscondiciones	Se ha consultado el estado del proceso de conciliación
Actores	Usuario
Descripción	1. El usuario accede a la sección de consulta de ficheros 2. El usuario elige un fichero del que consultar el resultado de conciliación ○ También es posible generar un informe
Variaciones (escenarios secundarios)	
Excepciones	
Notas	

1.17 Análisis de Interfaces de Usuario

1.17.1 Descripción de la Interfaz

1.17.1.1 Pantalla de índice

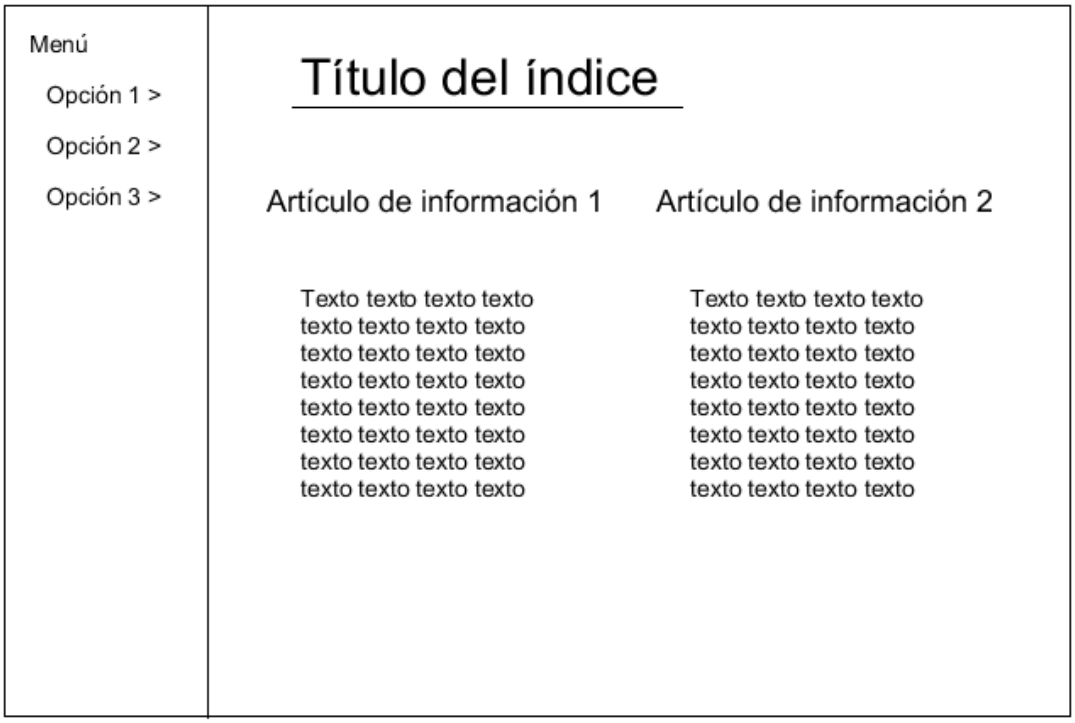


Figura 16.1. Esquema de la interfaz del índice

1.17.1.2 Pantalla de subida de ficheros

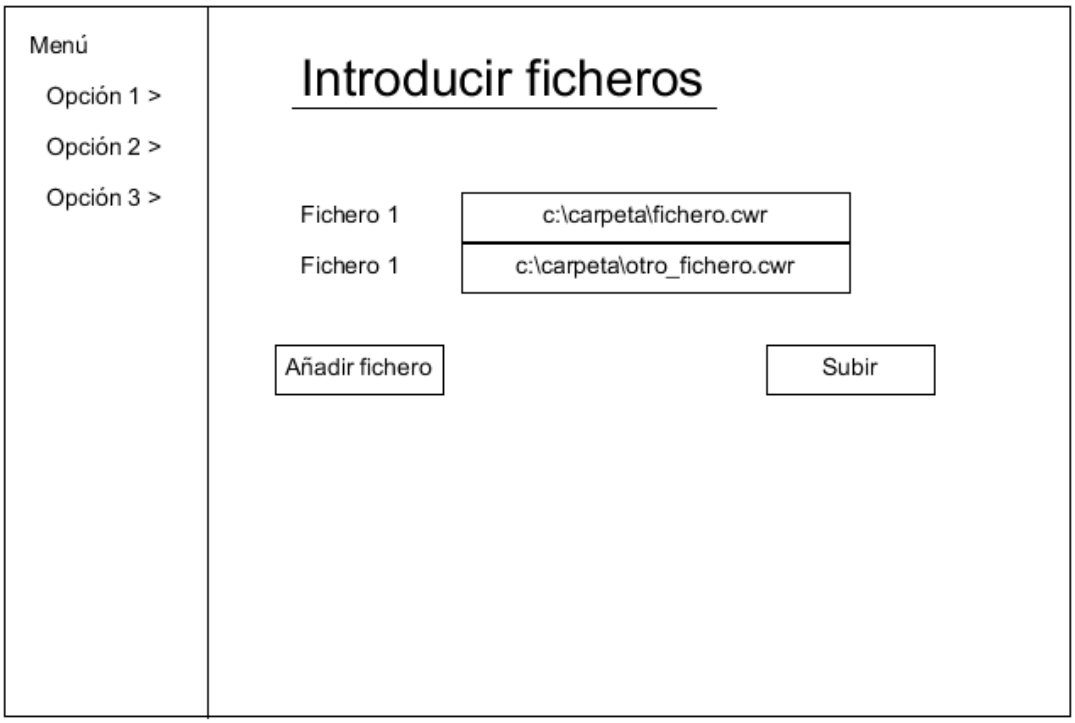


Figura 16.2. Esquema de la interfaz de subida de ficheros

1.17.1.3 Pantalla de listado de ficheros

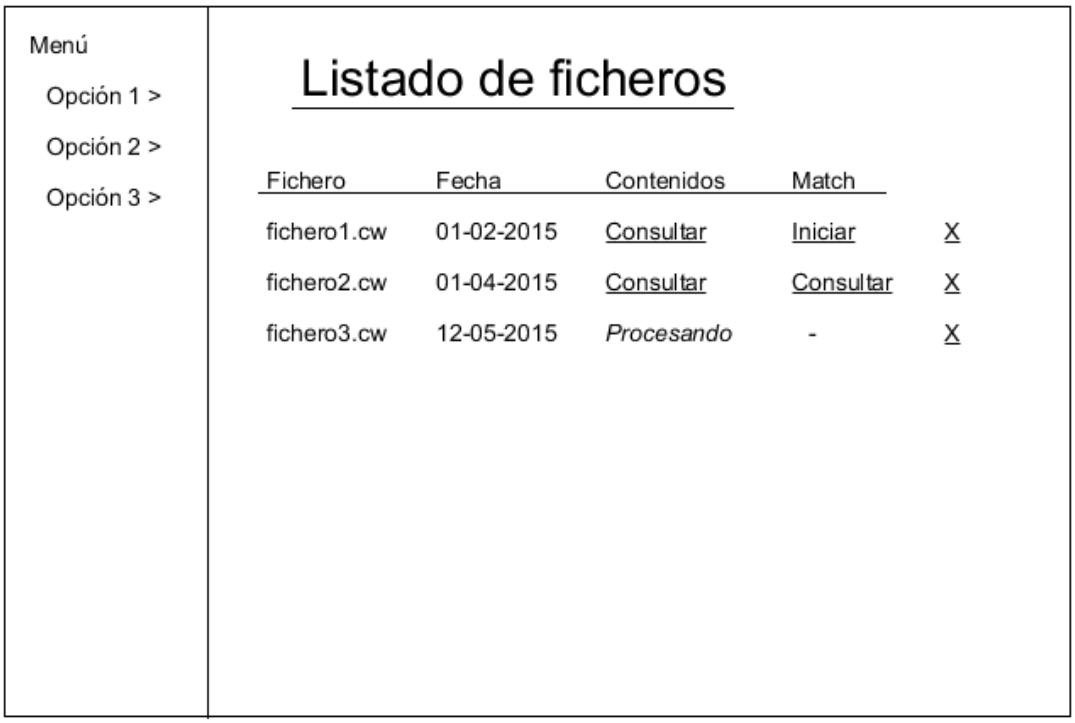


Figura 16.3. Esquema de la interfaz de la pantalla de listado de ficheros

1.17.1.4 Pantallas de datos de fichero CWR

1.17.1.4.1 Resumen

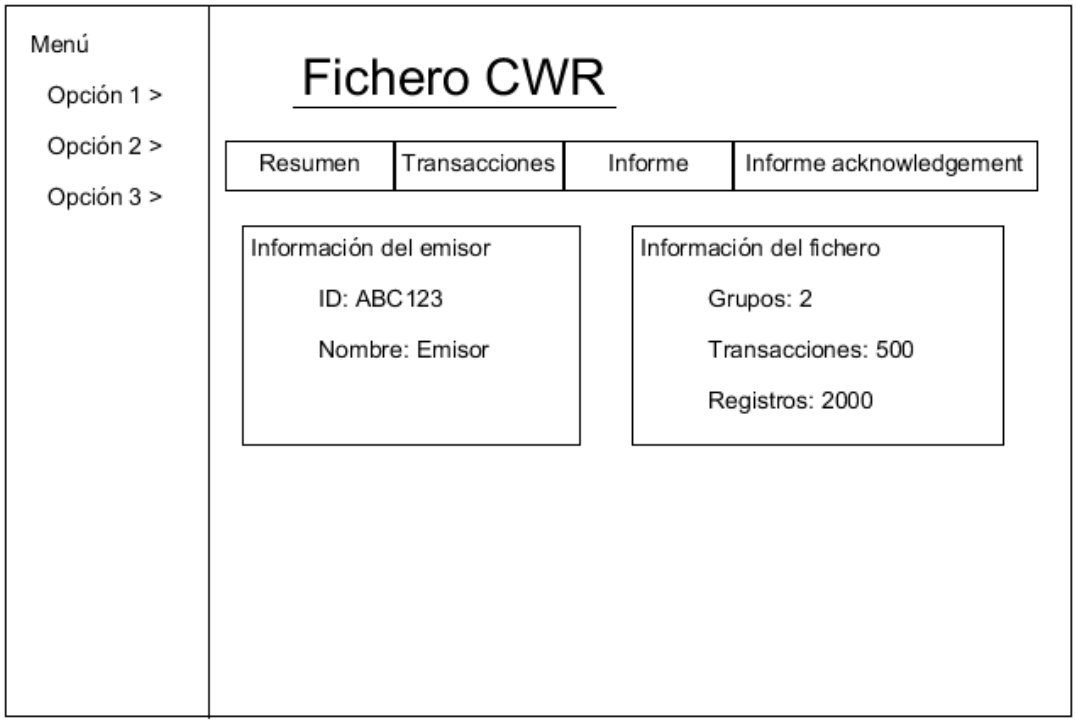


Figura 16.4. Esquema de la interfaz de la pantalla de resumen del fichero CWR

1.17.1.4.2 Grupo

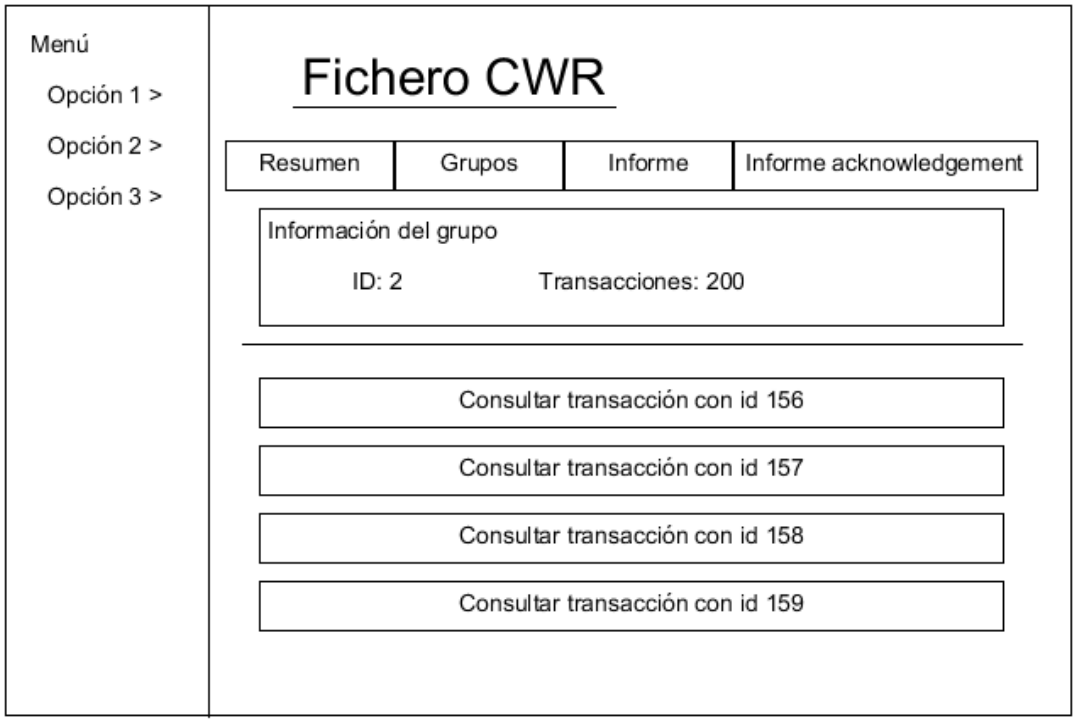


Figura 16.5. Esquema de la interfaz de la pantalla de grupo del fichero CWR

1.17.1.5 Pantallas de conciliación de datos

1.17.1.5.1 Configuración y envío

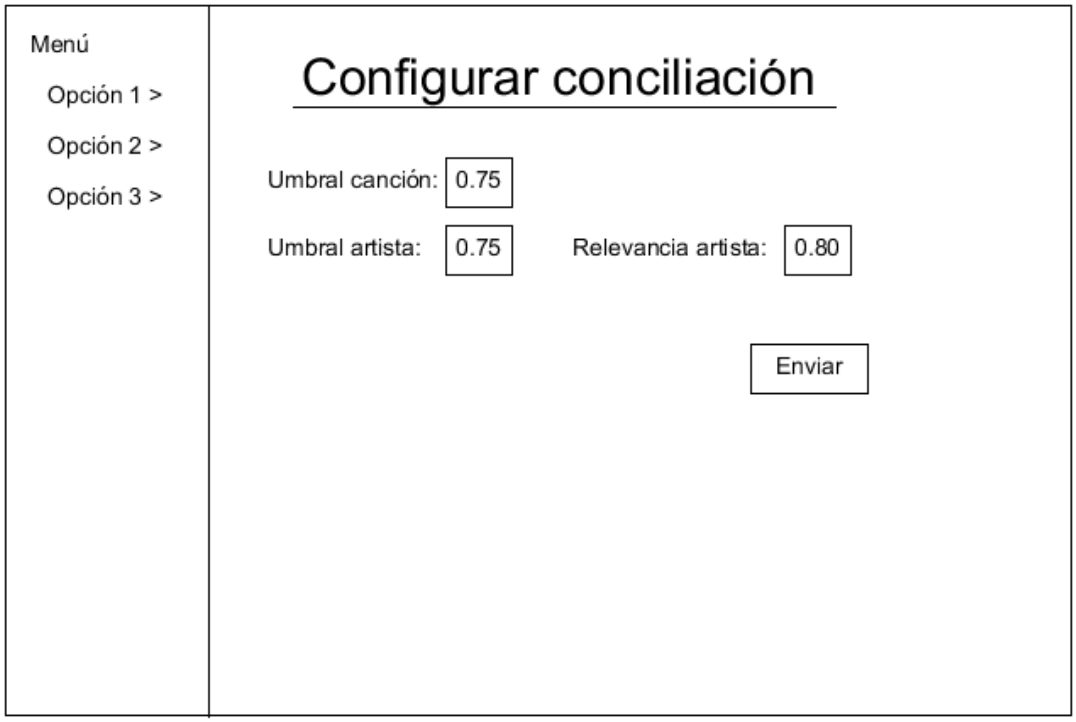


Figura 16.6. Esquema de la interfaz de la pantalla de configuración y envío para la conciliación

1.17.1.5.2 Resumen

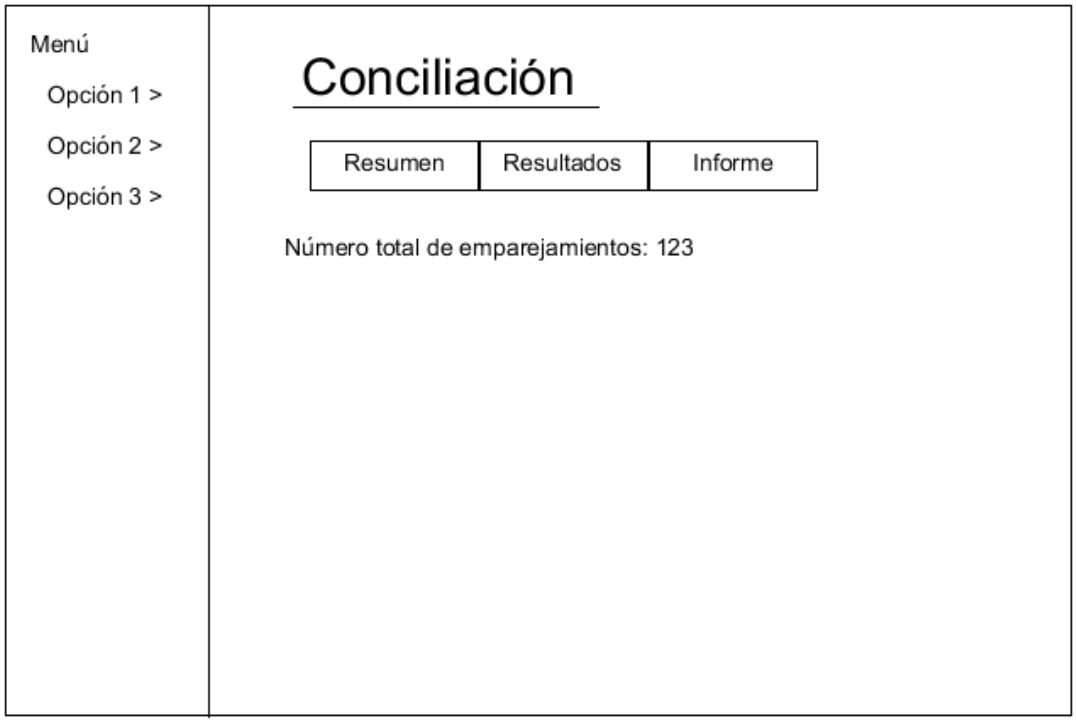


Figura 16.7. Esquema de la interfaz de la pantalla de resumen de conciliación

1.17.1.5.3 Resultados

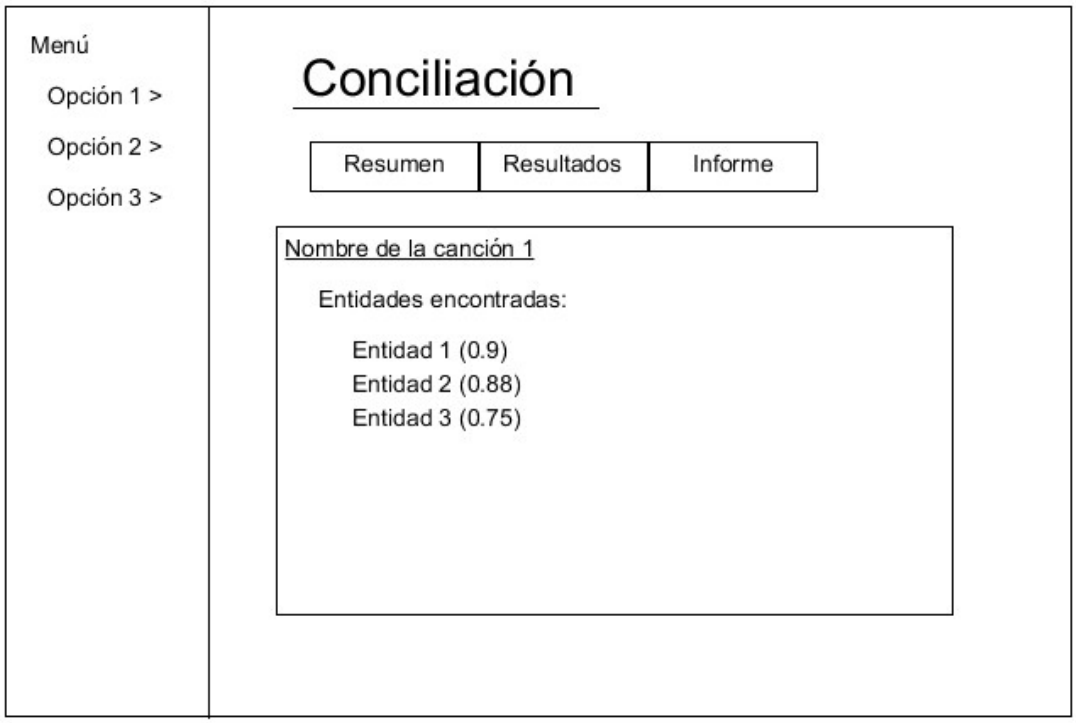


Figura 16.8. Esquema de la interfaz de la pantalla de resultados de conciliación

1.17.2 Descripción del Comportamiento de la Interfaz

Solo existirá interfaz gráfica en el cliente web, que mantendrá un estilo homogeneo, con una barra de menú con las opciones más importantes siempre visibles.

Además se ha orientado su uso entorno a los ficheros de entrada, ya que estos guiarán el flujo de trabajo de la aplicación. Por esto todas las opciones, tales como añadir nuevos datos o iniciar el proceso de conciliación, gravitarán entorno al listado de ficheros.

1.17.3 Diagrama de Navegabilidad

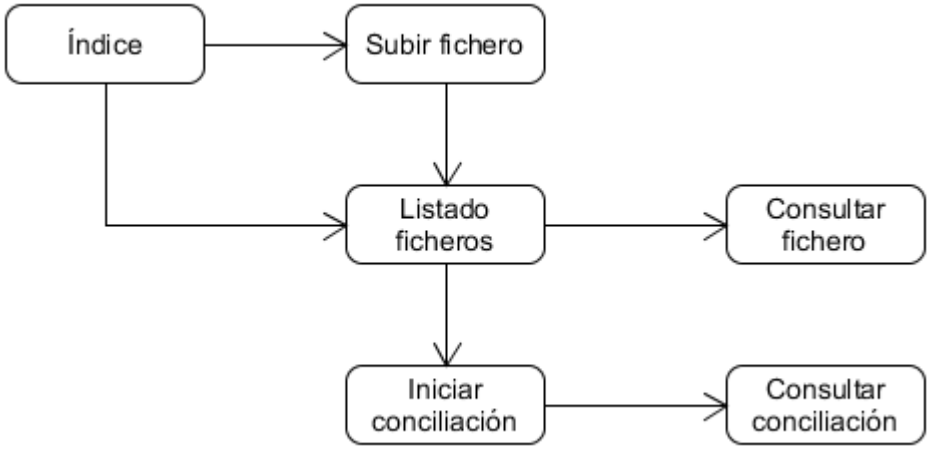


Figura 16.9. Diagrama de navegabilidad

1.18 Especificación del Plan de Pruebas

1.18.1 Pruebas unitarias

Existirá un conjunto de tests unitarios para cada subsistema, de manera que se valide el funcionamiento de cada uno de sus componentes.

En estas pruebas todas las dependencias del componente serán eliminadas por medio del uso de stubbing y mocks.

Aunque existirá un conjunto inicial de casos de prueba, muchos otros tests serán añadidos a la vez que se desarrolla el proceso, como resultado de aplicar la metodología de Test Driven Design, de manera que todas las funcionalidades con un mínimo de importancia serán testeadas.

Además, se utilizará un sistema de integración continua (Travis para los repositorios públicos, y Codeship para los privados) que ejecutará estos tests tras cada modificación al código, con lo que estos tests servirán también como pruebas de regresión.

1.18.2 Pruebas de integración

Para comprobar que los diferentes componentes de los distintos subsistemas funcionan correctamente en conjunto se han realizado pruebas de integración. Estas pruebas pueden parecer similares a las unitarias, pero la principal diferencia es que usará dependencias reales para su funcionamiento.

Esto no significa necesariamente que sean pruebas conectando distintos subsistemas. Un subsistema puede tener pruebas de integración solo para él, utilizando los distintos componentes que posee.

Al igual que las pruebas unitarias, estas se automatizarán por medio de un servicio de integración continua.

1.18.3 Pruebas de usabilidad

Debido a que es un prototipo, que posteriormente será adaptado al sistema del cliente, no se han realizado pruebas de usabilidad.

1.18.4 Pruebas de código

La calidad del código se ha comprobado principalmente por medio del servicio Landscape, que comprueba cada modificación al repositorio principal, generando un informe de calidad del código.

Además, se realiza periódicamente un testeo para comprobar que el código se adapta a los estándares de Python.

Se usará también un sistema de cobertura

1.18.5 Plan de pruebas

1.18.5.1 Pruebas unitarias

1.18.5.1.1 Caso de uso: Introducir ficheros de datos

Caso de Uso: Introducir ficheros de datos	
Prueba	**Resultado Esperado**
Se intenta subir un fichero de texto con extensión valida	El fichero es aceptado
Prueba	**Resultado Esperado**
Se intenta subir un fichero de texto cuya extensión indica que puede ser peligroso (ficheros ejecutables, por ejemplo)	El fichero es rechazado
Prueba	**Resultado Esperado**
Se intenta subir un fichero binario	El fichero es rechazado

1.18.5.1.2 Caso de uso: Consultar datos de ficheros CWR

Caso de Uso: Consultar datos de ficheros CWR	
Prueba	**Resultado Esperado**
Se piden datos de un fichero existente	Se muestra el estado del fichero
Prueba	**Resultado Esperado**
Se piden datos de un fichero inexistente	Se rechaza la petición
Prueba	**Resultado Esperado**
Se pide acknowledgement de un fichero CWR inexistente	Se rechaza la petición
Prueba	**Resultado Esperado**
Se pide informe de un fichero CWR inexistente	Se rechaza la petición
Prueba	**Resultado Esperado**
Se pide eliminar un fichero existente	Se elimina el fichero
Prueba	**Resultado Esperado**
Se pide eliminar un fichero inexistente	Se rechaza la petición

1.18.5.1.3 Caso de uso: Generar Conciliación de Ficheros CWR

Caso de Uso: Generar Conciliación de Ficheros CWR	
Prueba	**Resultado Esperado**
Se reciben datos de configuración inválidos	Se rechaza el proceso de conciliación

Prueba	**Resultado Esperado**
Se reciben datos de configuración inválidos	Se usa la configuración por defecto

1.18.5.1.4 Caso de uso: Comprobar Resultados de Conciliación

Caso de Uso: Comprobar Resultados de Conciliación	
Prueba	**Resultado Esperado**
Se piden resultados de fichero inexistente	Se rechaza la petición
Prueba	**Resultado Esperado**
Se piden resultados de fichero existente pero no procesado por el sistema de conciliación	Se rechaza la petición
Prueba	**Resultado Esperado**
Se piden resultados de fichero existente y procesado por el sistema de conciliación	Se muestran los datos
Prueba	**Resultado Esperado**
Se pide informe de fichero inexistente	Se rechaza la petición
Prueba	**Resultado Esperado**
Se pide informe de fichero existente pero no procesado por el sistema de conciliación	Se rechaza la petición

1.18.5.2 Pruebas de integración

1.18.5.2.1 Caso de uso: Introducir ficheros de datos

Caso de Uso: Introducir ficheros de datos	
Prueba	**Resultado Esperado**
Introducir un fichero CWR valido en la sección de CWRs	El fichero es aceptado y procesado
Prueba	**Resultado Esperado**
Introducir un fichero CWR con errores	El fichero es aceptado pero al procesarlo se rechaza
Prueba	**Resultado Esperado**
Introducir un fichero USO valido en la sección de CWRs	El fichero es aceptado y procesado
Prueba	**Resultado Esperado**
Introducir un fichero USO con errores	El fichero es aceptado pero al procesarlo se rechaza

1.18.5.2.2 Caso de uso: Consultar datos de ficheros CWR

Caso de Uso: Consultar datos de ficheros CWR	
Prueba	**Resultado Esperado**
Se pide acknowledgement de un fichero CWR existente	Se genera el fichero

1.18.5.2.3 Caso de uso: Generar Conciliación de Ficheros CWR

Caso de Uso: Generar Conciliación de Ficheros CWR	
Prueba	**Resultado Esperado**
Se pide conciliación de un fichero CWR procesado	Se genera el resultado de conciliación

1.18.5.2.4 Caso de uso: Comprobar Resultados de Conciliación

Caso de Uso: Comprobar Resultados de Conciliación	
Prueba	**Resultado Esperado**
Se pide informe de fichero existente y procesado	Se genera el informe

Diseño del Sistema

1.19 Arquitectura del Sistema

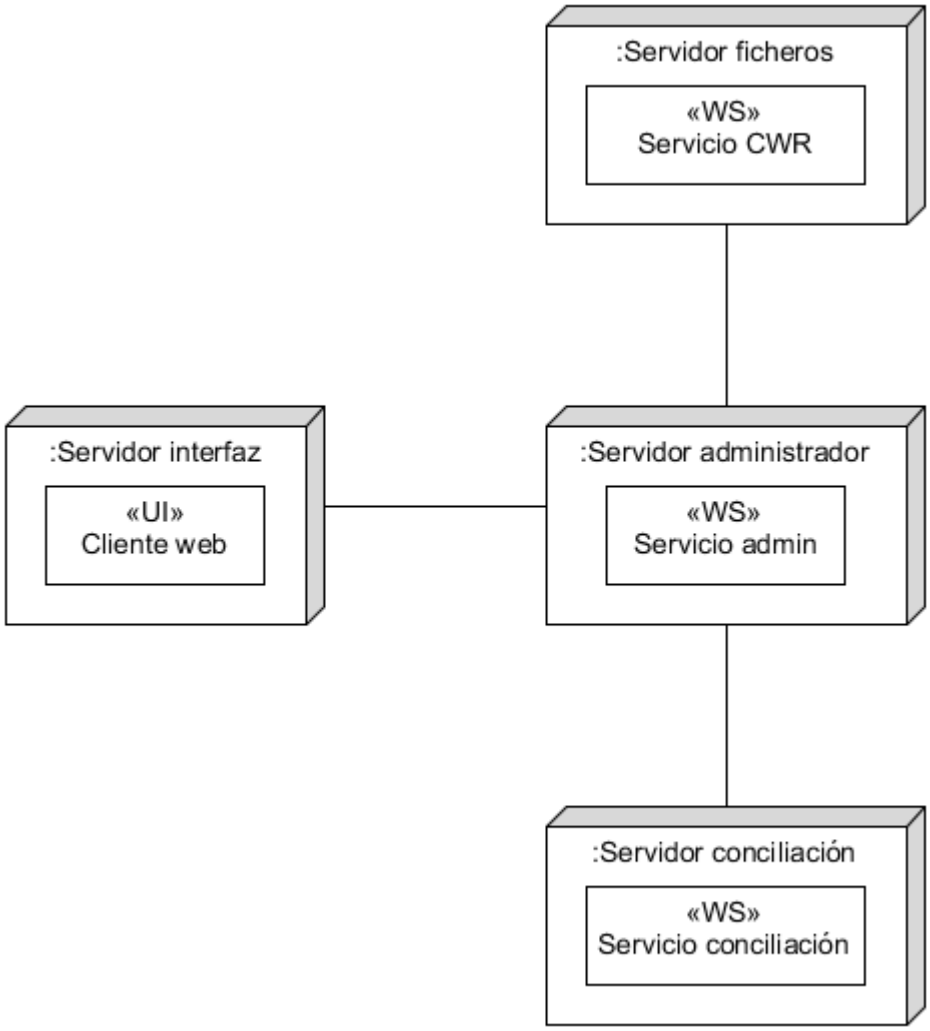


Figura 18.1. Diagrama de la arquitectura del sistema

El sistema se compone de cuatro piezas: un cliente web y tres servicios web. Todos estos están preparados para funcionar en servidores distintos, aunque es posible desplegarlos todos en el mismo, siempre y cuando este tenga la suficiente potencia.

Cada componente tiene el siguiente uso:

- Cliente web: la interfaz con la que interactuará el usuario
- Servicio administrador: el servicio web central que gestionará el procesamiento de ficheros y la conciliación

- Servicio CWR: encargado de procesar los ficheros CWR
- Servicio de conciliación: encargado de realizar el proceso de conciliación

1.19.1 Diagramas de Paquetes

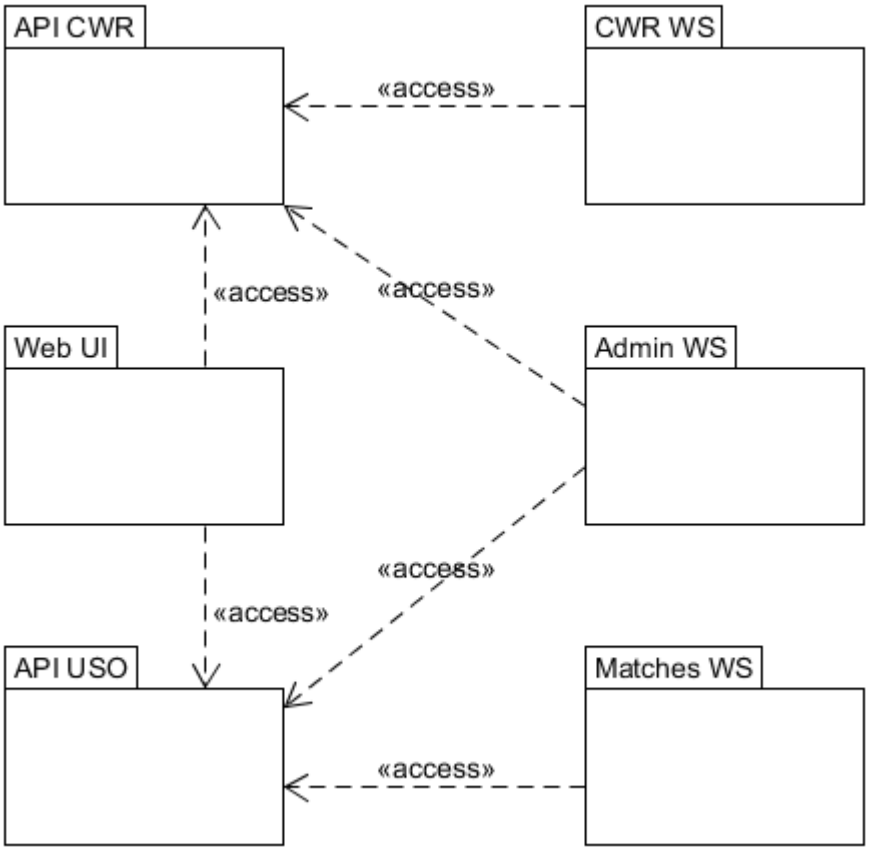


Figura 19.2. Diagrama de paquetes

La aplicación se compone de seis módulos distintos. Estos se corresponden cada uno a un subproyecto distinto.

Solamente los APIs son completamente independientes del resto de los subproyectos, mientras que el resto comparte depende de estos.

No existen dependencias entre los servicios web, ya que aunque estos comparten datos el proceso de comunicación entre ellos se basa en los módulos de API y mensajes JSON.

El objetivo de cada módulo es el siguiente:

- API CWR: modelo y parser para ficheros CWR
- API USO: modelo y parser para ficheros USO
- Web UI: interfaz para el usuario
- CWR WS: servicio web para el procesamiento de ficheros CWR
- Matches WS: servicio web para el proceso de conciliación

1.20 Diseño de Clases

1.20.1.1 Paquete API CWR

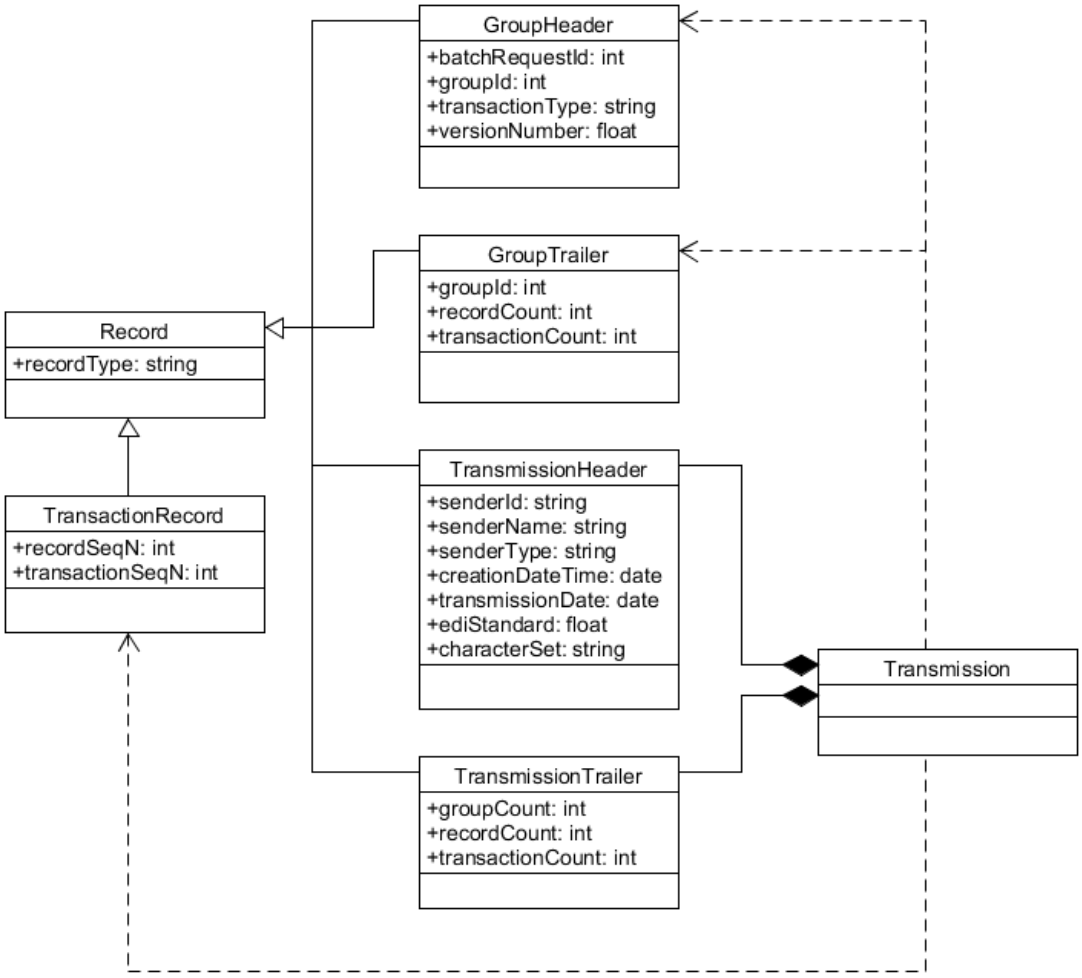


Figura 19.3. Diagrama de clases de transmisión del fichero CWR

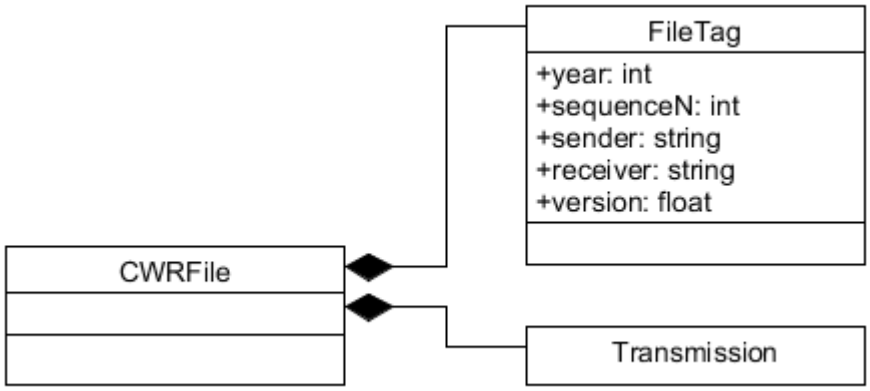


Figura 19.4. Diagrama de clases de modelo del fichero CWR

1.20.1.2 Paquete API USO

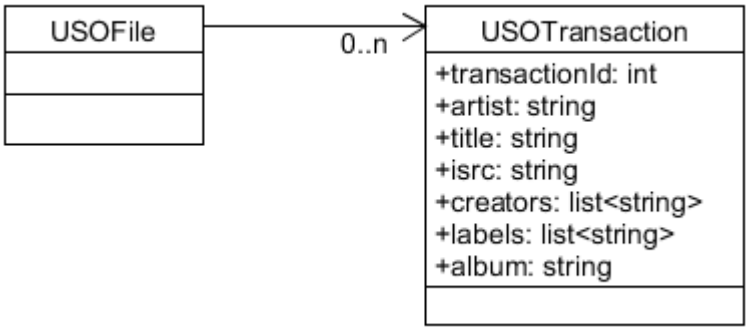


Figura 19.5. Diagrama de clases de modelo del fichero USO

1.20.1.3 Paquete Web UI

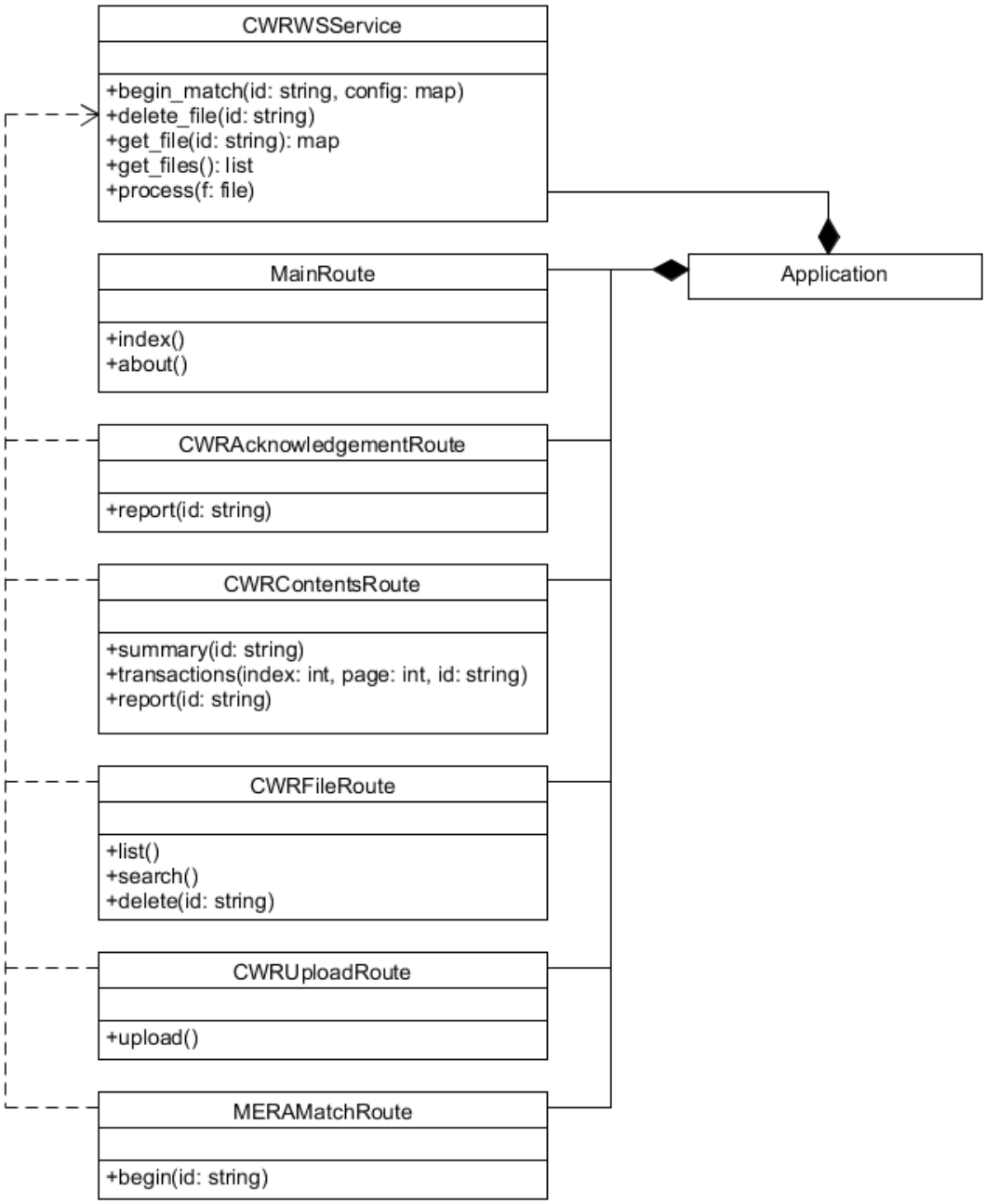


Figura 19.6. Diagrama de clases del paquete de Web UI

1.20.1.4 Paquete Admin WS

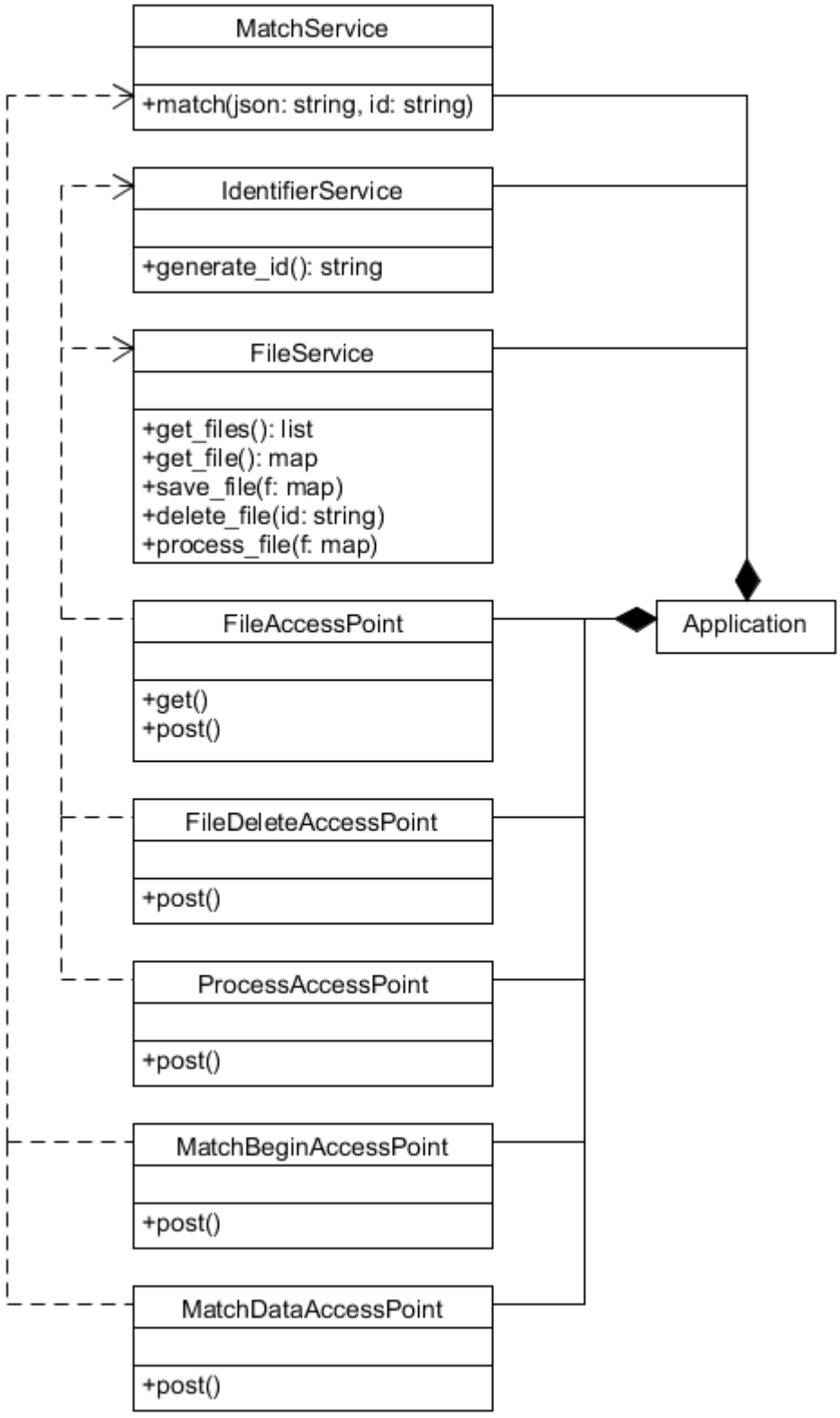


Figura 19.7. Diagrama de clases del paquete de Admin WS

1.20.1.5 Paquete CWR WS

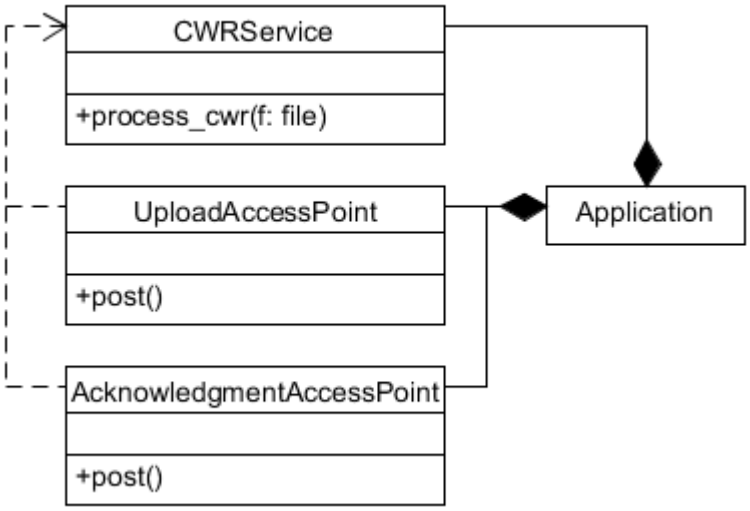


Figura 19.8. Diagrama de clases del paquete de CWR WS

1.20.1.6 Paquete Matches WS

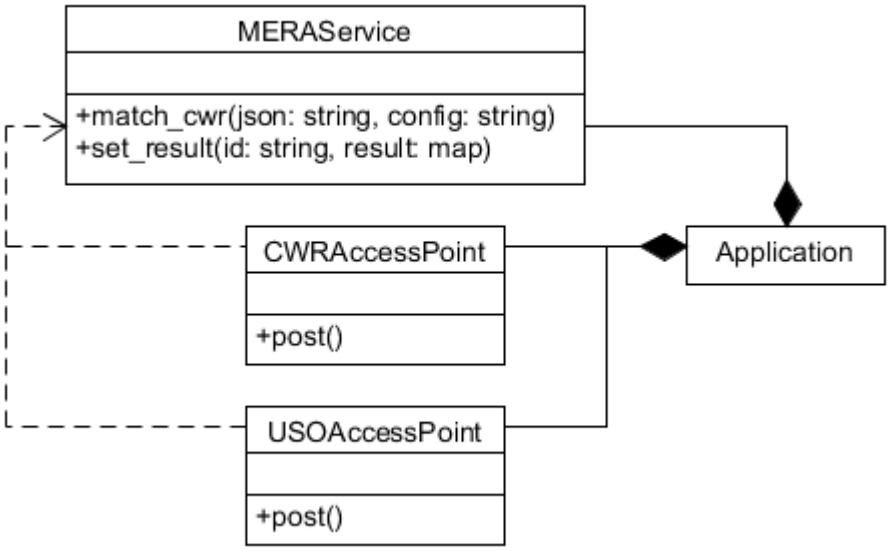


Figura 19.9. Diagrama de clases del paquete de Matches WS

1.21 Diagramas de Interacción y Estados

1.21.1 Caso de Uso: Introducir Ficheros de Datos

1.21.1.1 Diagramas de Interacción (Comunicación y Secuencia)

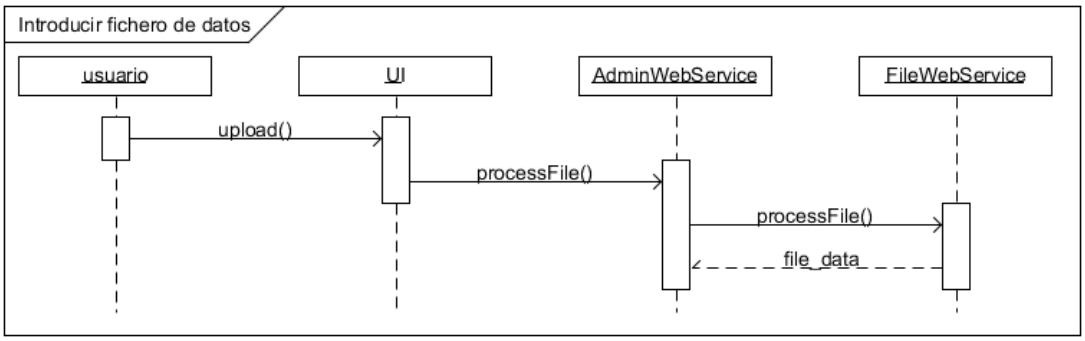


Figura 20.1. Diagrama de interacción de introducir ficheros de datos

El usuario pide subir un fichero a través de la interfaz web. Esta lo envía al servicio de administración, que lo reenvía al servicio de procesamiento de ficheros, que una vez lo ha procesado envía los datos de vuelva al servicio de administración.

1.21.2 Caso de Uso: Consultar Datos de Ficheros CWR

1.21.2.1 Diagramas de Interacción (Comunicación y Secuencia)

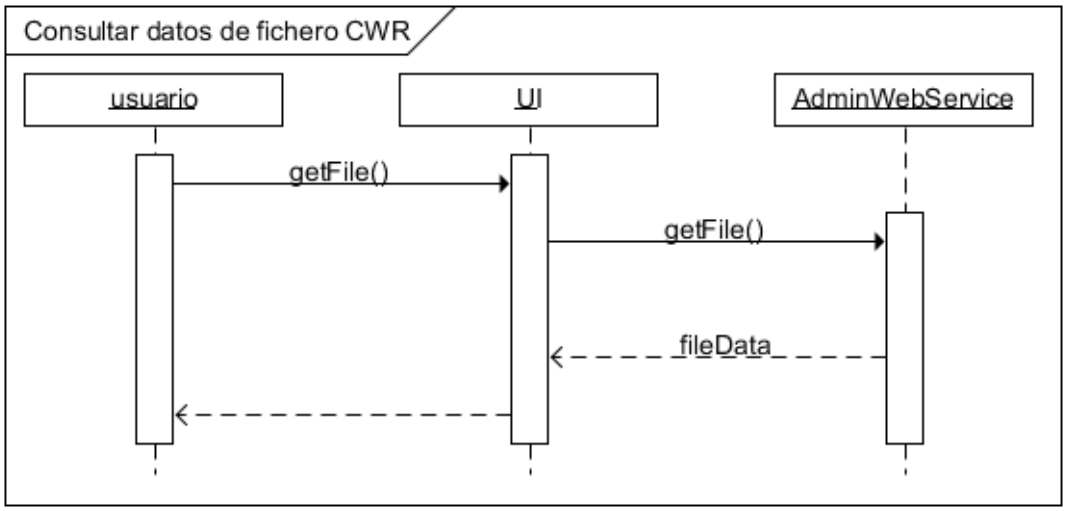


Figura 20.2. Diagrama de interacción de consultar datos de ficheros CWR

El usuario realiza la petición de consulta de un fichero al interfaz, que a su vez realiza esta petición al servicio web de administración. Tras esto los datos son enviados hacia el usuario.

1.21.3 Caso de Uso: Generar Conciliación de CWR

1.21.3.1 Diagramas de Interacción (Comunicación y Secuencia)

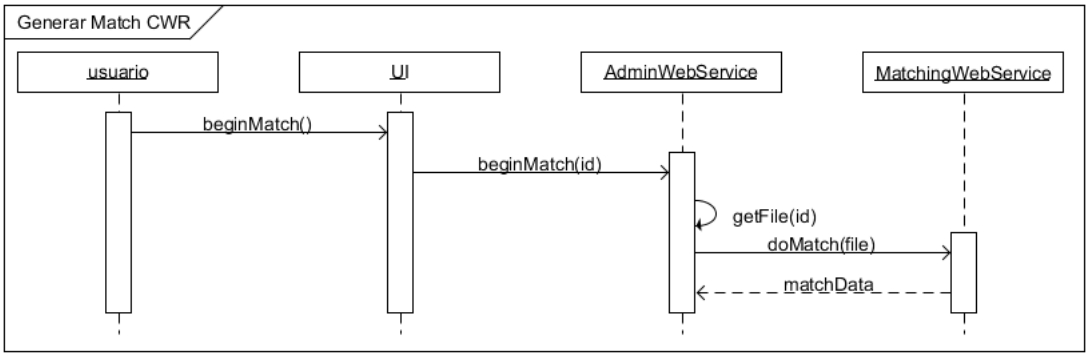


Figura 20.3. Diagrama de interacción de generar conciliación de CWR

El usuario realiza la petición de inicio de conciliación a la interfaz, que se la envía al servicio web de administración. Este adquiere los datos del fichero con el que realizar el proceso de conciliación y los envía al servicio web de conciliación.

Una vez este finaliza envía el resultado de vuelta al servicio web de administración.

Todo el proceso es asíncrono.

1.21.4 Caso de Uso: Comprobar Resultados de Conciliación

1.21.4.1 Diagramas de Interacción (Comunicación y Secuencia)

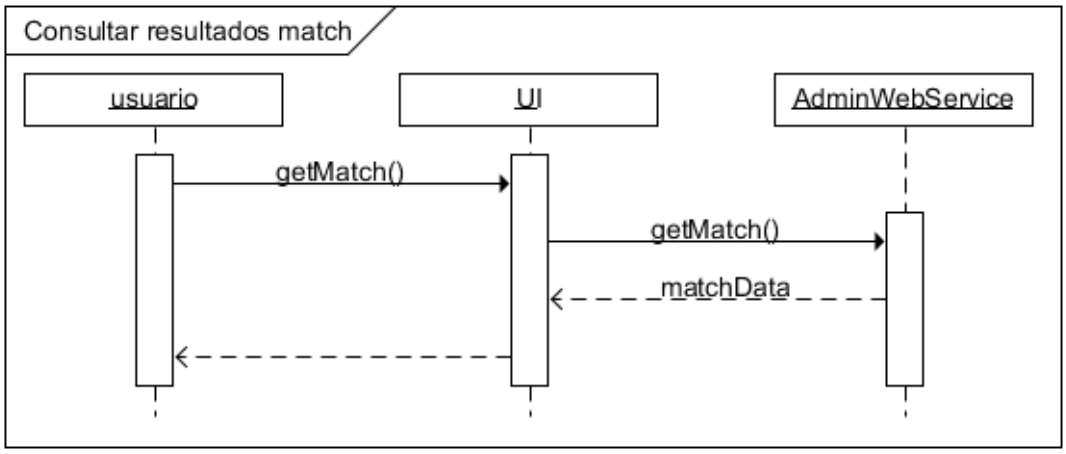


Figura 20.4. Diagrama de interacción de comprobar resultados de conciliación

El cliente pide los datos de una conciliación a la interfaz, que a su vez se los pide al servicio web de administración. Tras esto los datos se envían hacia el cliente.

1.22 Diagramas de Actividades

1.22.1 Diagramas de Actividad del Proceso de Conciliación

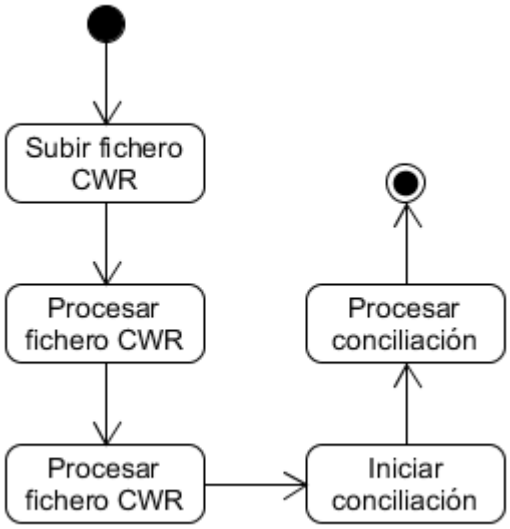


Figura 21.1. Diagrama de actividad del proceso de conciliación

Se incluye un diagrama de actividad para el proceso de conciliación.

1.23 Diseño de la Interfaz

1.23.1 Índice

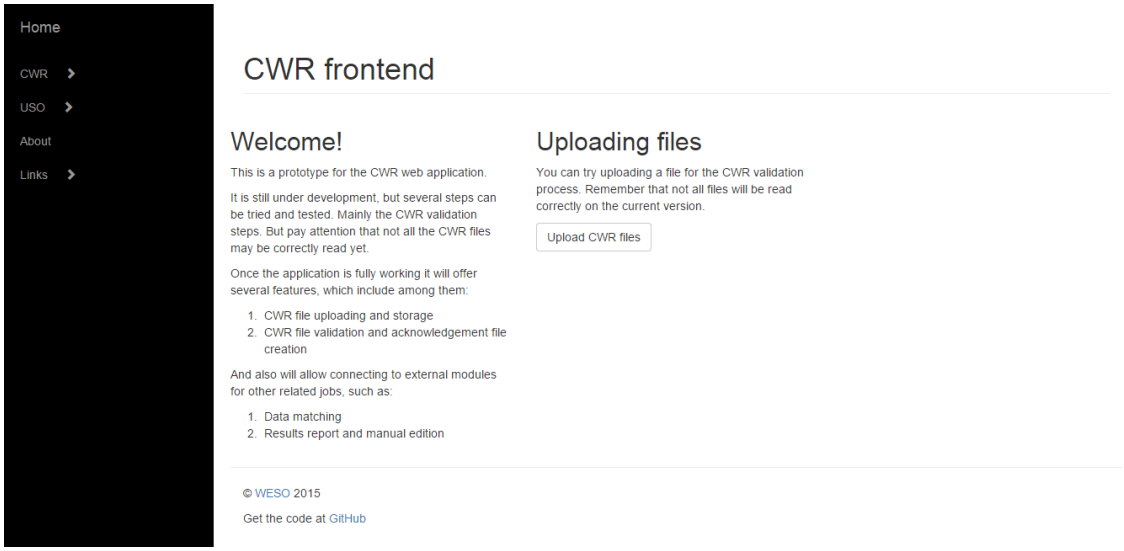


Figura 22.1. Interfaz del índice

1.23.2 Añadir ficheros

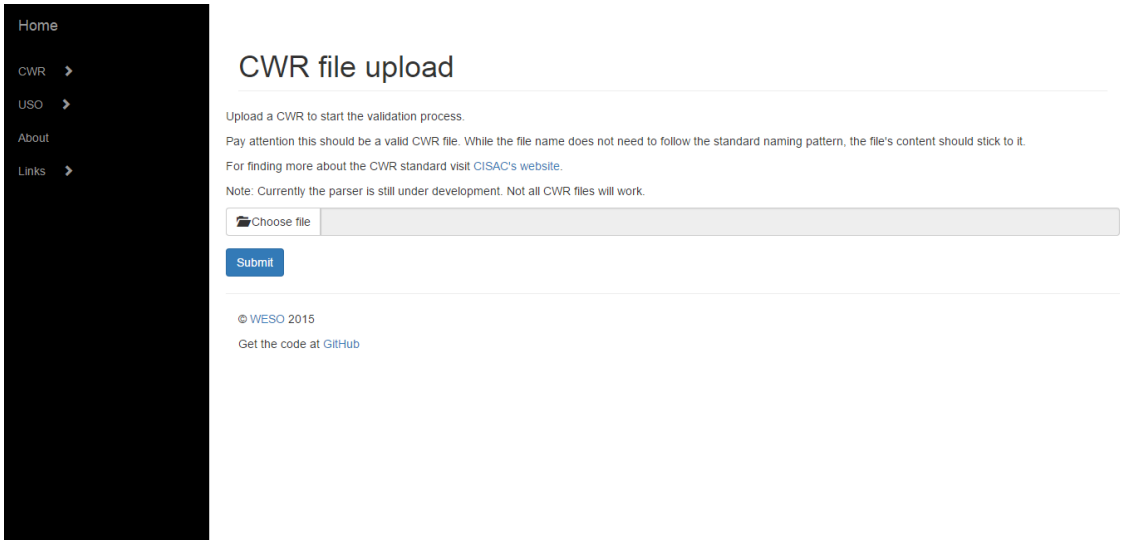


Figura 22.2. Interfaz de añadir ficheros

1.23.3 Listado de ficheros

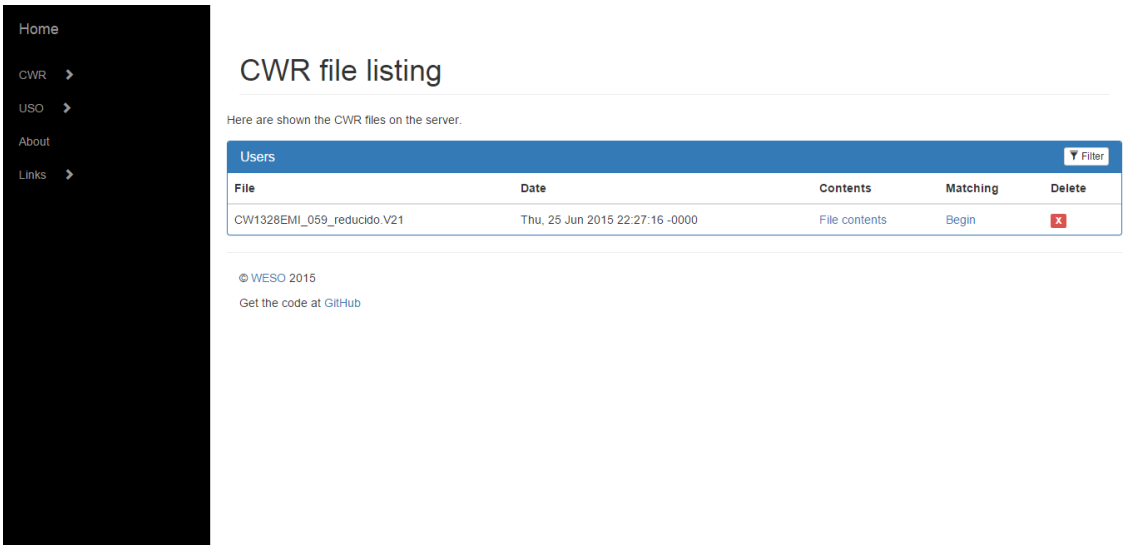


Figura 22.3. Interfaz de listado de ficheros

1.23.4 Contenido fichero CWR

1.23.4.1 Resumen

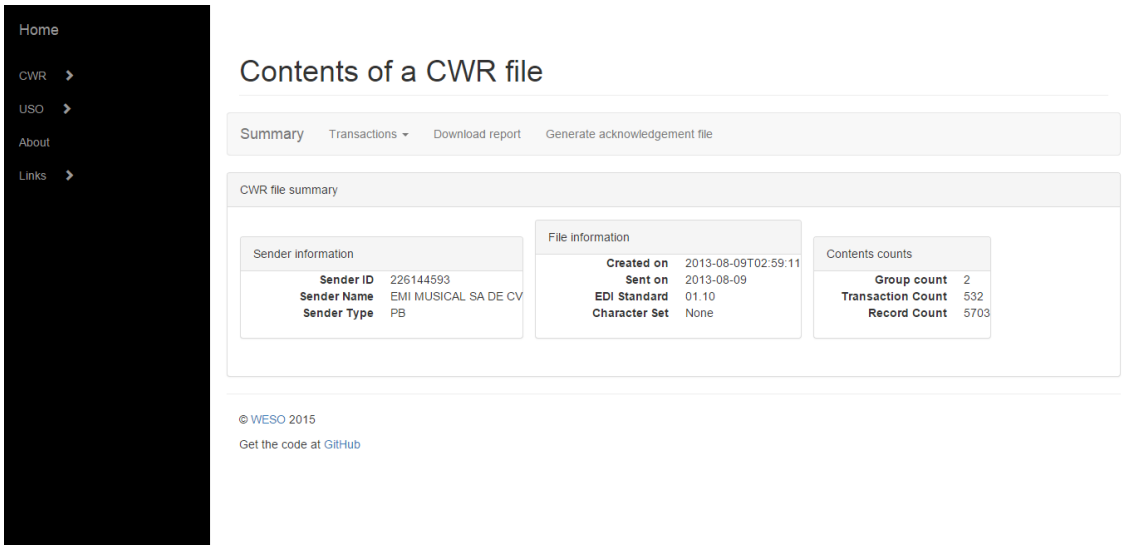


Figura 22.4. Interfaz de resumen de fichero CWR

1.23.4.2 Grupos

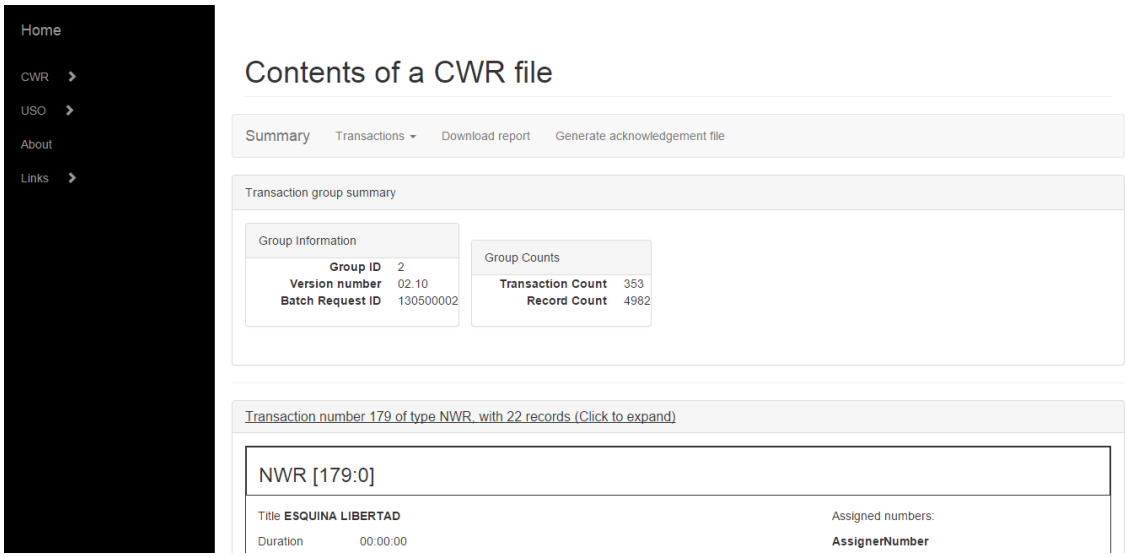


Figura 22.5. Interfaz de grupos de fichero CWR

1.23.5 Conciliación

1.23.5.1 Configuración

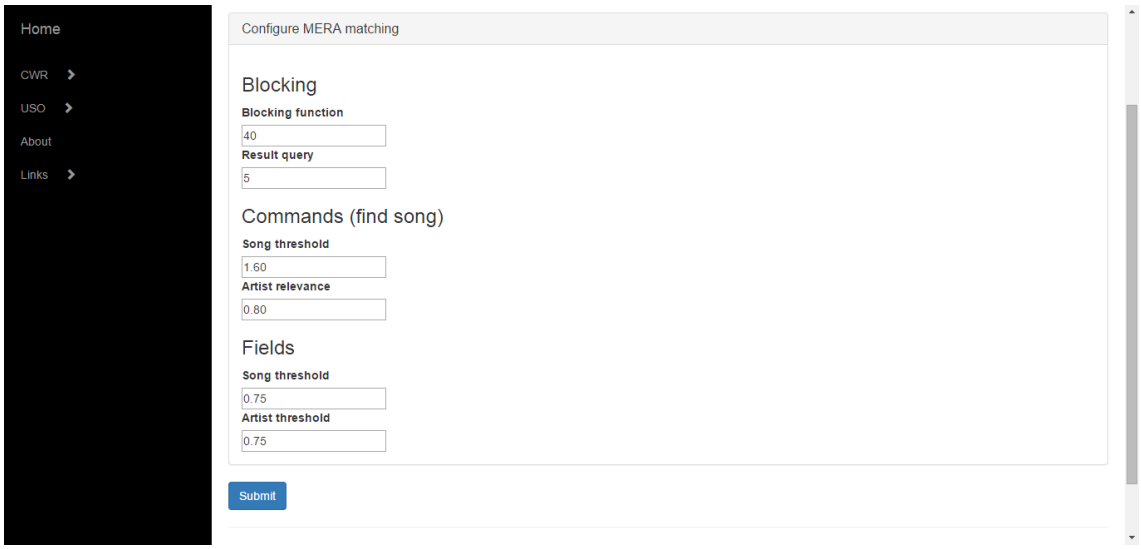


Figura 22.6. Interfaz de configuración de conciliación

1.23.5.2 Resultados

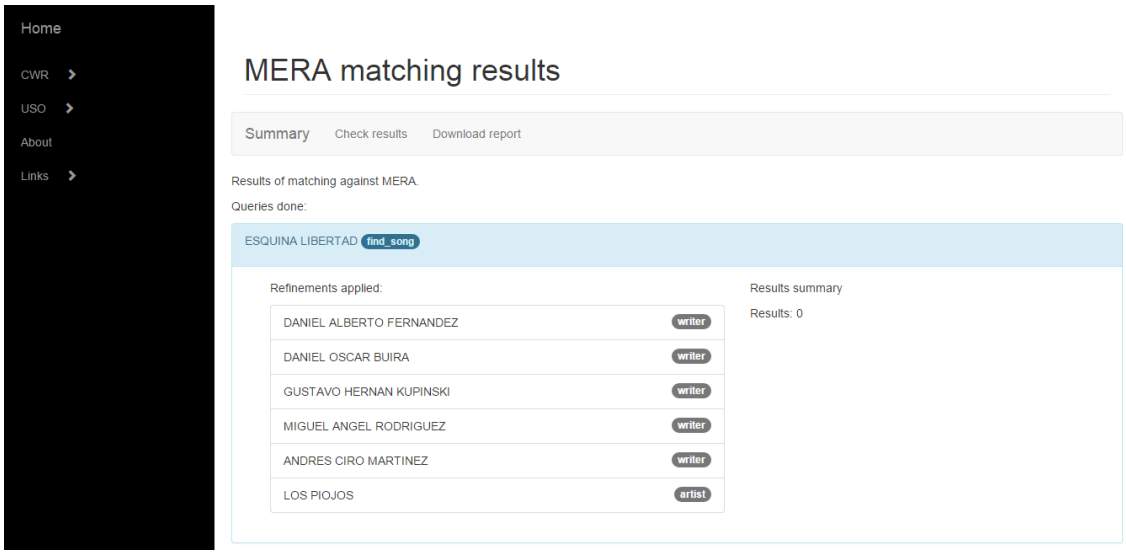


Figura 22.7. Interfaz de resultados de conciliación

1.24 Especificación Técnica del Plan de Pruebas

1.24.1 Pruebas Unitarias

Las pruebas unitarias son ejecutadas por medio de la librería de pruebas de Python unittest, y el framework para pruebas Tox. Cuando resulta necesario eliminar dependencias se utiliza stubs de prueba, aprovechando la facilidad de Python para crear prototipos, o si esto no resulta viable se crean mocks utilizando la librería mock de Python.

Tox ejecuta los tests en máquinas virtuales, de manera que sea posible siempre replicar las mismas condiciones en múltiples máquinas, y se utiliza para generar entornos de prueba para diversas versiones del interprete de Python.

Estos tests son ejecutados automáticamente tras cada modificación del código, gracias al servicio de integración continua (Travis para los repositorios públicos, Codeship para los privados), de manera que estos cambios se validen de manera continua.

En el servicio de integración continuación continua se realizan las pruebas en Linux. Las versiones del interprete Python usado dependerán del módulo. Los módulos web (servicios e interfaz) dependen de Flask, que solo acepta Python 2, con lo que se ejecutan en Python 2.6, Python 2.7 y pypy. Cualquier otro módulo (los de APIs) se ejecutan además con Python 3.2, Python 3.3, Python 3.4 y pypy3. De esta manera se cubren todas las versiones comunes del interprete Python.

Estas pruebas no se realizarán en conjunto para todos los subsistemas, ya que son activadas por cambios en los repositorios, y cada subsistema se guarda en un repositorio. Esto hace que cada subsistema deba tener su propio conjunto de pruebas, almacenado junto al código, que será el que ejecute el servicio de CI.

1.24.2 Pruebas de Integración y del Sistema

Cada subsistema tiene pruebas de integración internas, testeando sus componentes, que se realizan igual que las pruebas unitarias. Esto quiere decir que usan unittest y el servicio de integración continua.

Las pruebas de integración que afecten al sistema en su conjunto no están automatizadas. Periodicamente se despliega todo el sistema en el servidor de pruebas, y se procede a seguir una serie de pasos para su validación, que se pueden resumir en que se introducen datos y se comprueba que son procesados correctamente.

1.24.3 Pruebas de Rendimiento

Las tareas que mayor cantidad de recursos requieren son los sistemas de procesamiento de ficheros y conciliación de datos, sobre todo en casos de grandes volumenes de datos.

Las pruebas se han realizado utilizando ficheros de varios tamaños, comprendidos entre 800KB y los 235MB.

1.24.3.1 Procesamiento de ficheros CWR

Las pruebas de rendimiento con los ficheros CWR se realizarán procesando un fichero de diversos tamaños.

Se poseen varios ficheros de hasta 250Mb que se sabe son correctos para realizar estas operaciones, y se irán procesando uno a uno y midiendo el tiempo de respuesta.

1.24.3.2 Conciliación

El tiempo requerido para el proceso de conciliación dependerá de la cantidad de datos manejados por este. La aplicación no tiene control sobre estos datos, que forman parte de la necesidades propias del algoritmo MERA, pero gira entorno a varios gigas, lo que hace que incluso manejando datos pequeños requiera un tiempo demasiado largo.

Implementación del Sistema

1.25 Estándares y Normas Seguidos

1.25.1 PEP 0008

La PEP (Python Enhancement Proposal) 0008, localizada en https://www.python.org/dev/peps/pep-0008/, ofrece una guía de estilo que seguir a la hora de crear aplicaciones Python.

Esta guía se ha seguido de la manera más estricta posible a la hora de programar la aplicación. Para comprobarlo se ha utilizado la herramienta de Python flake8, que examina los ficheros de Python e indica todas las violaciones del standard encontradas.

Esta prueba se ha añadido a los casos de prueba automatizados, creando un entorno para tox que se encarga de este proceso. Pero no se ejecuta con el resto de las pruebas, ya que es demasiado estricto, y con que una sola linea sea demasiado larga provoca que el servicio de integración continua rechace el código.

Por esto, este test se ejecuta cada una o dos semanas, corrigiendo los fallos más importantes encontrados.

1.26 Lenguajes de Programación

1.26.1 Python

Es un lenguaje de programación interpretado, orientado a la programación funcional y dinámica, lo que facilita la creación de prototipos y el desarrollo rápido.

La Python Software Foundation ofrece implementaciones públicas y gratuitas del lenguaje.

Hay que tener cuidado con el hecho de que la versión más moderna, Python 3, no está soportada por todas las librerías, y tiene varios cambios al lenguaje que pueden hacerlo incompatible por la versión anterior, Python 2, que es mucho más popular.

Se han desarrollado los diferentes subsistemas de la aplicación utilizando Python 2, e intentando dar soporte a Python 3 siempre que fuera posible, pero se ha de considerar Python 2 como la versión soportada por el proyecto.

1.26.2 HTML

Lenguaje de marcado que se ha convertido en el standard por defecto para las páginas web.

Se utiliza para configurar las vistas de la aplicación web, combinándolo con CSS y Jinja

1.26.3 CSS

Lenguaje utilizado para definir estilos visuales.

Se utiliza para configurar el estilo de la aplicación web.

1.26.4 Jinja

Lenguaje de plantillas, que permite indicar patrones de sustitución con los que combinar o modificar ficheros.

Se utiliza para generar los ficheros HTML finales que serán utilizados por la aplicación. Esto se consigue principalmente marcando en los distintos ficheros que secciones serán sustituidas, o qué código importarán de otro fichero, aunque también su usan scripts de Jinja para el contenido dinámico.

1.27 Herramientas y Programas Usados para el Desarrollo

1.27.1 PyCharm

IDE para Python desarrollado por IntelliJ, similar a su IDE para Java IDEA.

Se ha utilizado la versión profesional, con licencia de estudiante, para el desarrollo de la aplicación, ya que da soporte a todos los lenguajes utilizados.

1.27.2 Flask

Framework que permite desarrollar aplicaciones web para Python. Posee extensiones que permiten también desarrollar servicios web. Su principal diferencia con otros frameworks similares (principalmente Django) es el ser un framework ligero y de bajos requisitos.

Se ha utilizado tanto para el interfaz web como los servicios.

1.27.3 Pyparsing

Librería de parsing orientada a objetos, en contraposición a ficheros de gramáticas.

Se utiliza para procesar los ficheros, y para gestionar el DSL de configuración del procesador de ficheros CWR.

1.27.4 pip

Gestor de dependencias para Python, que adquiere estas dependencias de un repositorio local o del repositorio remoto PyPI (Python Package Index).

Todas las dependencias se gestionan por medio de este gestor, aunque las librerías que no son públicas (las de conciliación) es necesario instalarlas de manera local.

1.27.5 Git

Sistema de control de versiones. Permite tener un repositorio de código que además controla el historial de cambios realizado.

Se ha utilizado, en combinación con Github y Bitbucket, para gestionar las versiones del proyecto.

1.27.6 Github

Servicio de gestión de versiones basado en Git. Gratuito para repositorios públicos.

Todos los repositorios públicos se almacenan en este servicio.

1.27.7 Github Issues

Sistema de gestión de incidencias de Github. Utilizado para mantener control de los fallos a corregir, y de las modificaciones pendientes en los subsistemas.

1.27.8 Travis CI

Servicio de integración continua, pensado para ser integrado con repositorios de Github.

Todos los repositorios de Github hacen uso de este servicio para ejecutar los tests tras cada modificación.

1.27.9 Coveralls

Servicio de cobertura de tests.

Con ayuda de Travis se generan informes de cobertura de tests utilizando este servicio.

1.27.10 Landscape

Servicio de calidad de código.

Se integra a Github para generar informes de calidad de código tras cada modificación.

1.27.11 Bitbucket

Servicio de gestión de versiones basado en Git. Ofrece repositorios privados gratuitos.

Se utiliza para almacenar los subsistemas relacionados con el proceso de conciliación.

1.27.12 Codeship

Servicio de integración continua que acepta repositorios de Github y Bitbucket.

Se utiliza para ejecutar los tests de los repositorios privados tras cada modificación.

1.27.13 Basecamp

Servicio de gestión de proyecto, utilizado para comunicarse con el cliente de manera continua.

1.28 Creación del Sistema

Todos los aspectos con los que nos hemos encontrado durante la implementación debemos describirlos aquí.

1.28.1 Problemas Encontrados

1.28.1.1 Incompatibilidades entre Python 2 y Python 3

Originalmente se pretendía realizar la aplicación en Python 3, con la idea de que al ser la versión más moderna del interprete se encontrarían menos problemas durante su desarrollo.

Después de algunas pruebas iniciales se comprobó que esto era inviable, ya que muchas librerías necesarias, principalmente el framework Flask, no eran compatibles con Python 3.

La solución aplicada fue pasar a utilizar Python 2, en particular Python 2.7, como el interprete del proyecto.

1.28.1.2 Despliegue en servidor Apache

Para desplegar la aplicación en el servidor Apache se ha utilizado un servicio WSGI que lo envuelve.

Esto causa un problema a la hora de recibir y almacenar ficheros enviados por el usuario. Para las pruebas en local se usaba una carpeta dentro del proyecto en la que se almacenaban estos ficheros, pero en el servidor surge un choque de permisos, ya que el servicio WSGI posee su propio usuario, mientras que la carpeta del proyecto pertenece al usuario de Apache.

Para solucionar esto se ha optado por procesar los ficheros directamente en memoria, sin almacenarlos previamente.

1.28.1.3 Codificación de caracteres en ficheros

Un problema importante a la hora de manejar ficheros CWR ha estado relacionado con la codificación de caracteres. Mientras que en Python 3 se han arreglado muchos de los problemas para detectar codificaciones, en Python 2 esto no ocurre.

Incluso utilizando ficheros con la codificación UTF-8 surgen problemas, ya que los ficheros recibidos suelen tener caracteres extendidos, lo que requiere leerlos con la codificación latin-1, y con esto surgen problemas con el BOM de inicio que algunos ficheros UTF-8 tienen.

La solución encontrada a sido decodificar los ficheros con la codificación latin-1, de manera que se lean caracteres especiales, y eliminar todos los símbolos iniciales inválidos, de manera que el BOM se ignore.

1.28.1.4 Estandard CWR y procesamiento de estos ficheros

El estandard CWR ha sido desarrollado por CISAC para la industria musical. Permite crear y gestionar ficheros de transacciones, que indican cambios en las licencias poseídas por una compañía. En el contexto de la aplicación, sirve para saber que obras musicales pertenecen a una compañía, ya que estos datos se encuentran contenidos en el fichero.

El problema es que no es un estandard claro, y existen numerosas ambigüedades, y varias versiones alternativas que cambian datos muy concretos de la estructura.

Esto hace que no sea raro que cada empresa o agencia interprete de manera distinta el estandard, ya que no existe ninguna implementación oficial, y la que se acaba utilizando se negocia entre las empresas que vayan a usar este standard. Estas implementaciones no tienen porque ser compatibles entre si.

No existe ninguna solución buena a este problema, ya que siempre existirá una implementación alternativa, y los miembros de la industria musical mantienen una política de opacidad acerca de como gestionan ellos estos ficheros. Por tanto se ha optado por hacer el parser de ficheros CWR configurable por medio de un DSL.

1.28.1.5 Grandes volumenes de datos

El mayor problema que se ha encontrado ha sido el manejo de grandes volumenes de datos. Cuando los ficheros de entrada alcanza un tamaño de unos poco cientos de megas resulta imposible gestionarlo, debido al enorme coste que esto requiere, lo que provoca que el servidor pare el proceso. Esto ocurre tanto con el sistema de procesamiento de ficheros como con el de conciliación.

Para solucionar este problema haría falta rediseñar el sistema de procesamiento de ficheros y el de conciliación.

Desarrollo de las Pruebas

1.29 Pruebas Unitarias

1.29.1 Caso de Uso: Introducir ficheros de datos

Caso de Uso : Introducir ficheros de datos	
Prueba	**Resultado Esperado**
Se intenta subir un fichero de texto con extensión valida	El fichero es aceptado
	Resultado Obtenido
	El fichero es aceptado
Prueba	**Resultado Esperado**
Se intenta subir un fichero de texto cuya extensión indica que puede ser peligroso (ficheros ejecutables, por ejemplo)	El fichero es rechazado
	Resultado Obtenido
	El fichero es rechazado
Prueba	**Resultado Esperado**
Se intenta subir un fichero binario	El fichero es rechazado
	Resultado Obtenido
	Se acepta el fichero, ya que Python es incapaz de diferenciar entre ficheros binarios y de texto

1.29.2 Caso de Uso: Consultar Datos de Ficheros CWR

Caso de Uso : Consultar Datos de Ficheros CWR	
Prueba	**Resultado Esperado**
Se piden datos de un fichero existente	Se muestra el estado del fichero
	Resultado Obtenido
	Se muestra el estado del fichero
Prueba	**Resultado Esperado**
Se piden datos de un fichero inexistente	Se rechaza la petición
	Resultado Obtenido
	Se rechaza la petición
Prueba	**Resultado Esperado**
Se pide acknowledgement de un fichero CWR inexistente	Se rechaza la petición
	Resultado Obtenido
	Se rechaza la petición
Prueba	**Resultado Esperado**
Se pide informe de un fichero CWR inexistente	Se rechaza la petición
	Resultado Obtenido
	Se rechaza la petición
Prueba	**Resultado Esperado**
Se pide eliminar un fichero existente	Se elimina el fichero
	Resultado Obtenido
	Se elimina el fichero
Prueba	**Resultado Esperado**
Se pide eliminar un fichero inexistente	Se rechaza la petición
	Resultado Obtenido
	Se rechaza la petición

1.29.3 Caso de Uso: Generar Conciliación de Ficheros CWR

Caso de Uso : Generar Conciliación de Ficheros CWR	
Prueba	**Resultado Esperado**
Se reciben datos de configuración inválidos	Se rechaza el proceso de conciliación
	Resultado Obtenido
	Se rechaza el proceso de conciliación
Prueba	**Resultado Esperado**
Se reciben datos de configuración inválidos	Se usa la configuración por defecto
	Resultado Obtenido
	Se usa la configuración por defecto

1.29.4 Caso de Uso: Comprobar Resultados de Conciliación

Caso de Uso : Comprobar Resultados de Conciliación	
Prueba	**Resultado Esperado**
Se piden resultados de fichero inexistente	Se rechaza la petición
	Resultado Obtenido
	Se rechaza la petición
Prueba	**Resultado Esperado**
Se piden resultados de fichero existente pero no procesado por el sistema de conciliación	Se rechaza la petición
	Resultado Obtenido
	El fichero es rechazado
Prueba	**Resultado Esperado**
Se piden resultados de fichero existente y procesado por el sistema de conciliación	Se muestran los datos
	Resultado Obtenido
	Se muestran los datos
Prueba	**Resultado Esperado**
Se pide informe de fichero inexistente	Se rechaza la petición
	Resultado Obtenido
	Se rechaza la petición
Prueba	**Resultado Esperado**
Se pide informe de fichero existente pero no procesado por el sistema de conciliación	Se rechaza la petición
	Resultado Obtenido
	Se rechaza la petición

1.30 Pruebas de Integración y del Sistema

1.30.1 Caso de Uso: Introducir ficheros de datos

Caso de Uso : Introducir ficheros de datos	
Prueba	**Resultado Esperado**
Introducir un fichero CWR valido en la sección de CWRs	El fichero es aceptado y procesado
	Resultado Obtenido
	El fichero es aceptado y procesado
Prueba	**Resultado Esperado**
Introducir un fichero CWR con errores	El fichero es aceptado pero al procesarlo se rechaza
	Resultado Obtenido
	El fichero es rechazado
Prueba	**Resultado Esperado**
Introducir un fichero USO valido en la sección de CWRs	El fichero es aceptado y procesado
	Resultado Obtenido
	El fichero es aceptado y procesado
Prueba	**Resultado Esperado**
Introducir un fichero USO con errores	El fichero es aceptado pero al procesarlo se rechaza
	Resultado Obtenido
	El fichero es aceptado pero al procesarlo se rechaza

1.30.2 Caso de Uso: Consultar datos de ficheros CWR

Caso de Uso : Consultar Datos de Ficheros CWR	
Prueba	**Resultado Esperado**
Se pide acknowledgement de un fichero CWR existente	Se genera el fichero
	Resultado Obtenido
	Funcionalidad no implementada en el prototipo

1.30.3 Caso de Uso: Generar Conciliación de ficheros CWR

Caso de Uso : Generar Conciliación de Ficheros CWR	
Prueba	**Resultado Esperado**
Se pide conciliación de un fichero CWR procesado	Se genera el resultado de conciliación
	Resultado Obtenido
	Se genera el resultado de conciliación

1.30.4 Caso de Uso: Comprobar Resultados de Conciliación

Caso de Uso : Comprobar Resultados de Conciliación	
Prueba	**Resultado Esperado**
Se pide informe de fichero existente y procesado	Se genera el informe
	Resultado Obtenido
	Se genera el informe

1.31 Pruebas de Rendimiento

1.31.1 Procesamiento de ficheros CWR

Se han realizado pruebas con diversos ficheros para medir el tiempo que se tarda en procesarlos.

Tamaño	Tiempo
4KB	0.12s
795KB	11s
10,7MB	3,13m
19MB	5,7m
34MB	53m

Como se puede comprobar, existe un punto en que el coste de procesar un fichero explota, impidiendo el uso de los ficheros más grandes.

Manuales del Sistema

1.32 Manual de Instalación

Existen dos opciones para instalar el proyecto, que son utilizando un entorno de desarrollo o uno de despliegue.

Para el primero solamente es necesario poseer el código fuente de los diversos subsistemas y un IDE capaz de ejecutar código Python. Mientras que para el despliegue hará falta un servidor capaz de dar soporte a Python, y se explicará la instalación en un servidor apache.

1.32.1 Gestión de dependencias

Un problema común a ambos entornos de desarrollo es la gestión de dependencias. La mayor parte de las dependencias son accesibles a través de PyPi, por tanto solo es necesario ir a la carpeta raiz de cada proyecto y utilizar el comando:

pip install -rrequirements.txt

Esto instalará las dependencias, que siempre están indicadas en el fichero requirements.txt. En caso de utilizar un IDE como PyCharm para ejecutar el código (en caso de utilizar un entorno de desarrollo), este se encargará automáticamente de instalar las dependencias.

1.32.1.1 Dependencias privadas

El servicio de conciliación requiere de la librería MERA, la cual no es accesible públicamente.

La única forma de instalarla es adquirir una copia del código y ejecutar, desde la raiz del proyecto, el siguiente comando:

python setup.py install

Esto instalará en el repositorio local la librería.

1.32.2 Configuración de puntos de acceso

Otro punto común al entorno de desarrollo y despliegue es la configuración de los puntos de acceso, las direcciones web que se utilizarán para comunicarse con los servicios web.

Por defecto, estas estarán configuradas para funcionar correctamente en el entorno de desarrollo, pero es indispensable modificarlas en el entorno de despliegue.

Para configurarlas existen dos opciones. Se puede modificar el fichero app.py, en la carpeta del proyecto, o, la forma recomendada, se pueden añadir variables de entorno al sistema.

La primera opción requiere modificar el fichero src/app.py, donde 'src' será la carpeta con el código fuente, y modificar dentro del método _load_services las direcciones usadas. Pero solo ha de utilizarse como última instancia.

La segunda requiere añadir una serie de variables de entorno, que han de ser accesibles al proceso que ejecute la aplicación web y los servicios:

Variable	Uso
CWR_ADMIN_WS	Dirección del servicio web administrador
CWR_FILE_WS	Dirección del servicio web de procesamiento de ficheros
CWR_MATCH_WS	Dirección del servicio web de conciliación

1.32.3 Instalación en un entorno de desarrollo

Para instalar el proyecto en local hace falta un IDE para Python (se recomienda Pycharm) y los siguientes proyectos:

- Cliente web
- Servicio web administrador
- Servicio web de procesamiento de ficheros
- Servicio web de conciliación

Los tres poseen en la carpeta raiz un fichero de python que permite ejecutarlos. Estos son run_client.py para el cliente web, y run_ws.py para los servicios web.

Solo es necesario ejecutar estos ficheros y visitar la dirección http://127.0.0.1:33507/

1.32.4 Instalación en un entorno de despliegue

Para instalar el proyecto en un servidor hace falta un servidor capaz de hospedar Pythony los siguientes proyectos:

- Cliente web
- Servicio web administrador
- Servicio web de procesamiento de ficheros
- Servicio web de conciliación

Para que el servidor sea capaz de ejecutarlos se usará WSGI. Todos los proyectos poseen un fichero ejecutable para tal fin, estos son run_client.wsgi para el cliente web, y run_ws.wsgi para los servicios web.

Para la instalación se supondrá que se posee un servidor apache funcionando en un servidor Linux.

1.32.4.1 Instalación de WSGI

Es necesario instalar el servicio de WSGI para Apache para poder ejecutar los servicios web y la aplicación.

Para esto se utilizará el siguiente comando:

sudo aptitude install libapache2-mod-wsgi

Tras lo cual solo es necesario reiniciar Apache:

sudo service apache2 restart

1.32.4.2 Configuración de Apache

Para ejecutar los servicios es necesario modificar el fichero de sitios de Apache, que normalmente se encuentra en '/etc/apache2/sites-available/default'.

A este fichero hay que añadirle, dentro de <VirtualHost>, las siguientes lineas:

```
WSGIDaemonProcess cwr python-path=/var/www/cwr

WSGIProcessGroup cwr

WSGIScriptAlias /cwr /var/www/cwr/run_client.wsgi process-group=cwr

<Directory “/var/www/cwr/”>

  Order deny, allow

  Allow from all

</Directory>
```

Estas se han de adaptar a los distintos servicios web, asignandoles un identificador distinto a cada uno, e indicando correctamente la carpeta en la que se encuentran.

Tras esto, se ha de reiniciar apache.

1.33 Manual de Usuario

1.33.1 Añadir ficheros CWR

El primer paso a la hora de utilizar la aplicación es añadir nuevos datos. Esto se hace a traves de la sección para subir ficheros.

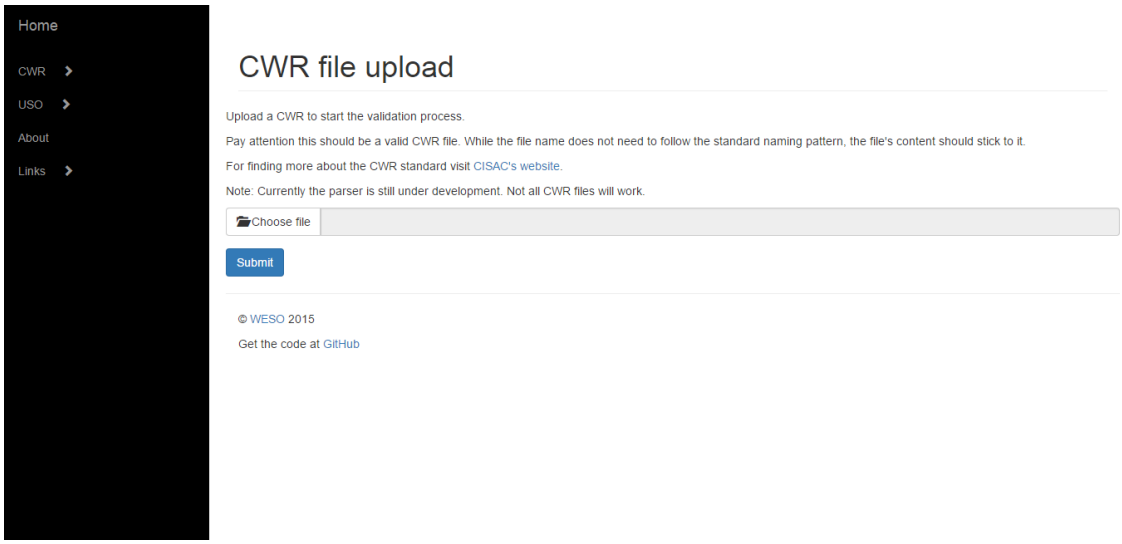


Figura 32.1. Ejemplo uso de interfaz de subida de ficheros

Tras elegir un fichero adecuado, este será gestionado. El proceso puede tardar, dependiendo del tamaño del fichero, y el listado de ficheros indicará su estado.

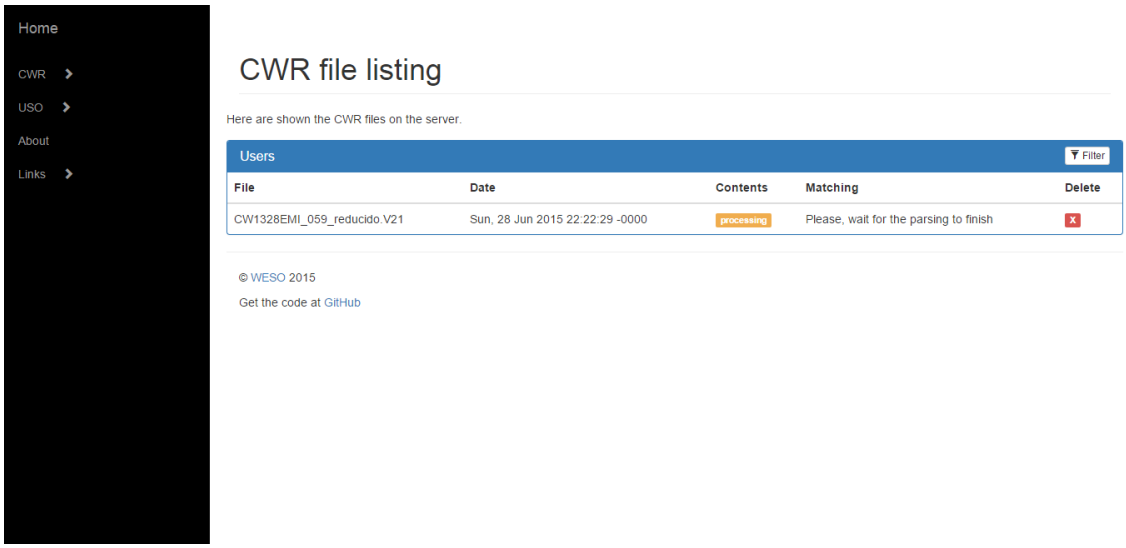


Figura 32.2. Ejemplo uso de interfaz de listado de ficheros

Una vez el fichero haya sido procesado, es posible consultar su contenido, o iniciar el proceso de conciliación.

1.33.2 Consultar datos de fichero CWR

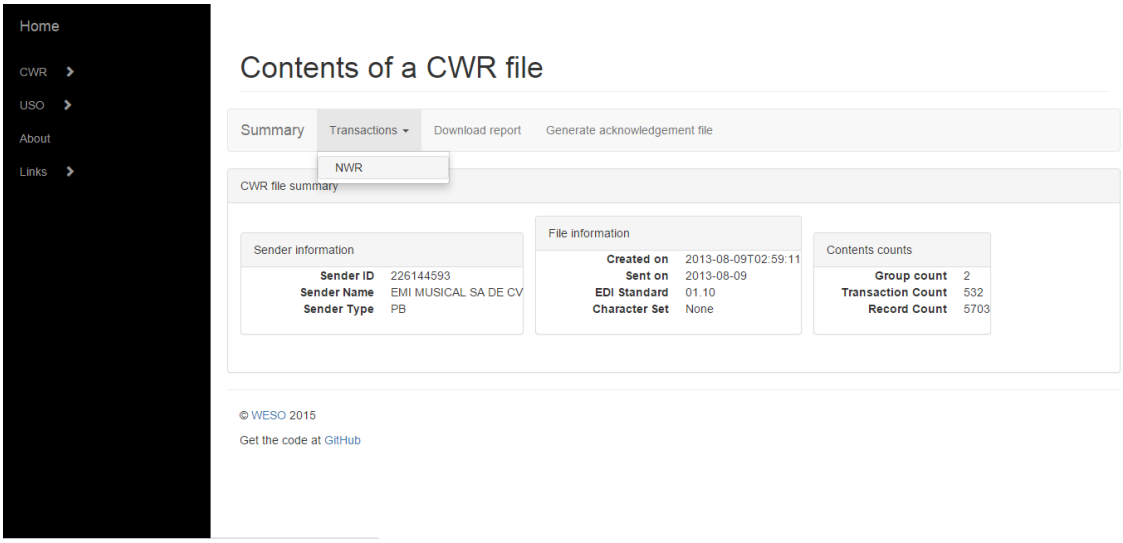


Figura 32.3. Ejemplo uso de interfaz de resumen de consulta de datos de fichero CWR

Tras elegir un fichero que consultar se mostrará el sumario. Es posible consultar los distintos grupos de transacciones por medio de la opción de transacciones.

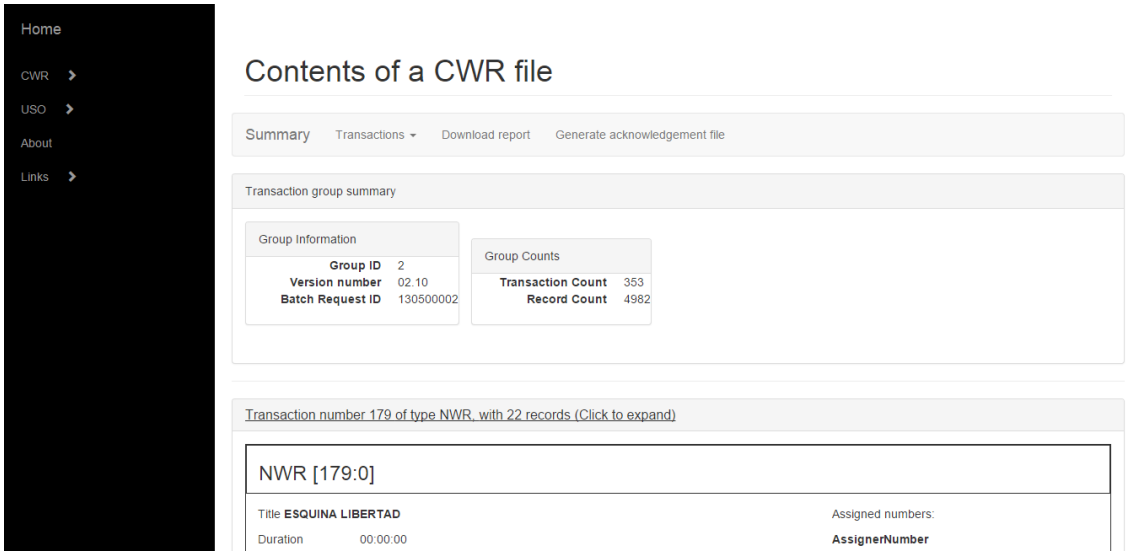


Figura 32.4. Ejemplo uso de interfaz de consulta de datos de fichero CWR

1.33.3 Conciliación

Al pedir iniciar el proceso de conciliación primero será neesario dar los datos de configuración.

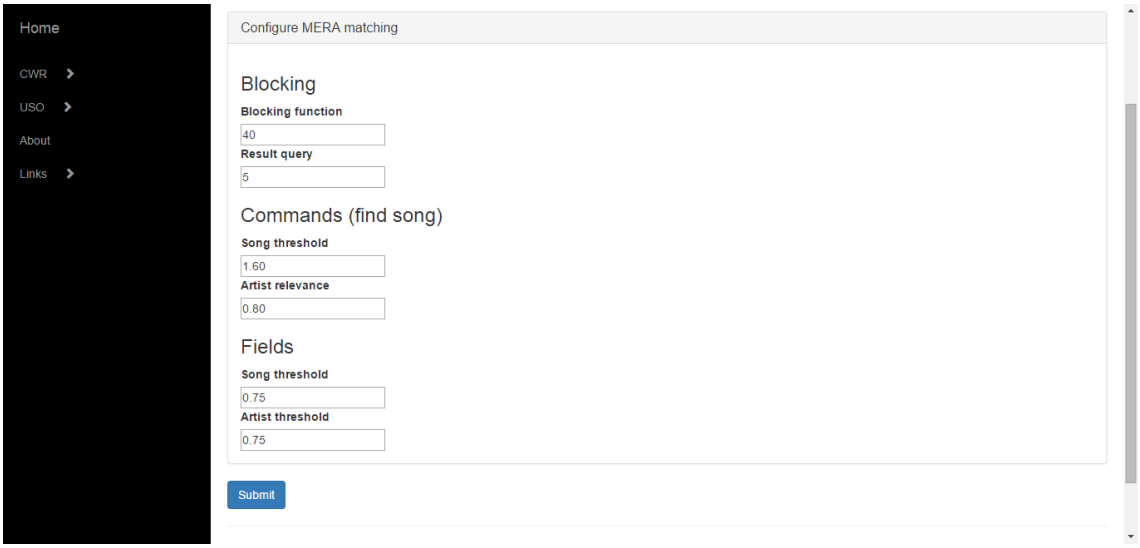


Figura 32.5. Ejemplo uso de interfaz de consulta de configuración de conciliación

Tras esto será necesario esperar a que termine el proceso de conciliación.

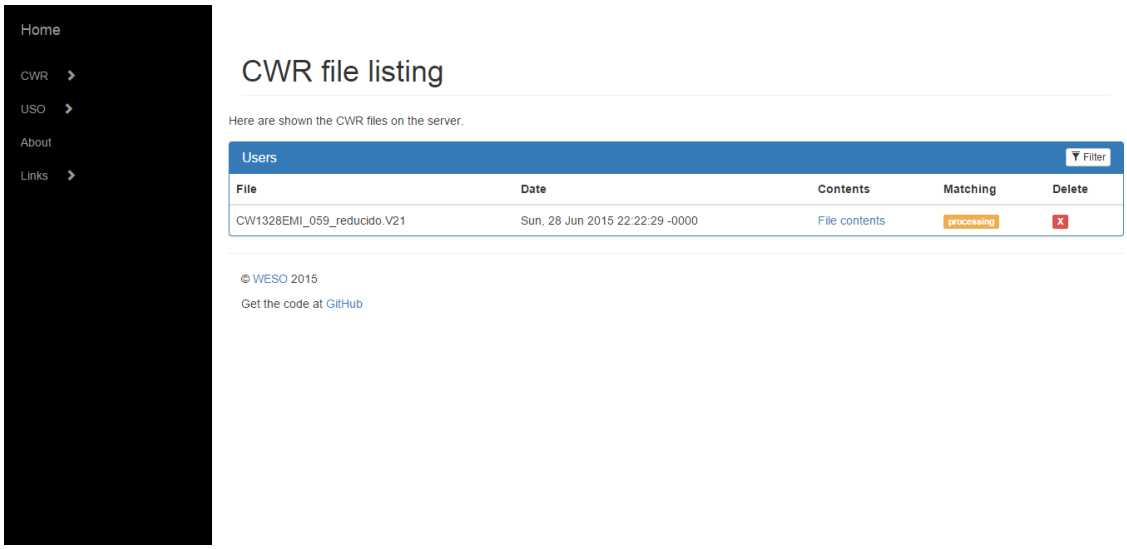


Figura 32.6. Ejemplo uso de interfaz de consulta de configuración de conciliación

Una vez ha terminado el proceso de conciliación es posible consultar sus resultados.

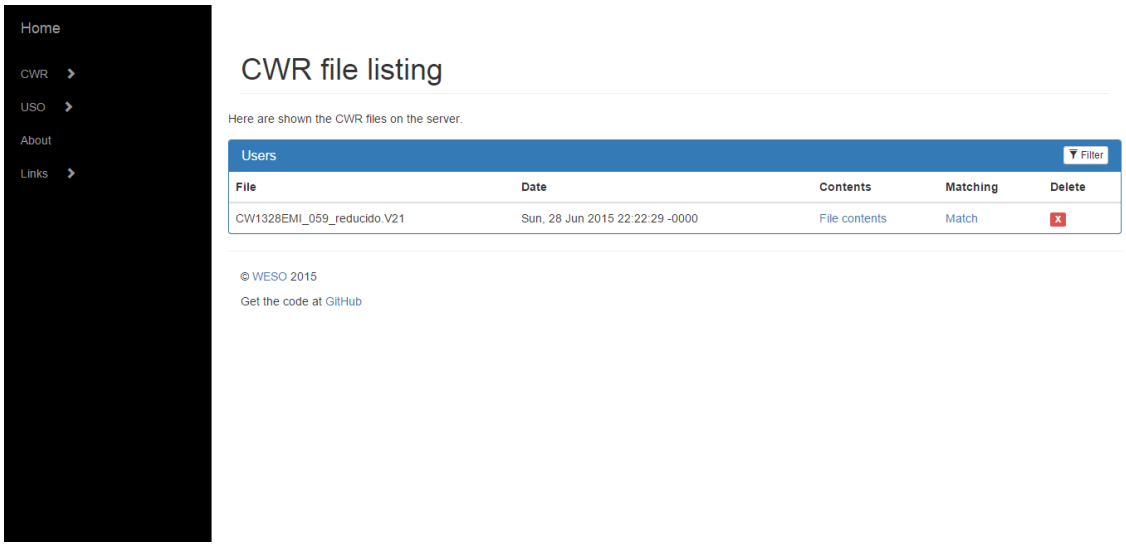


Figura 32.7. Ejemplo uso de interfaz de listado de ficheros

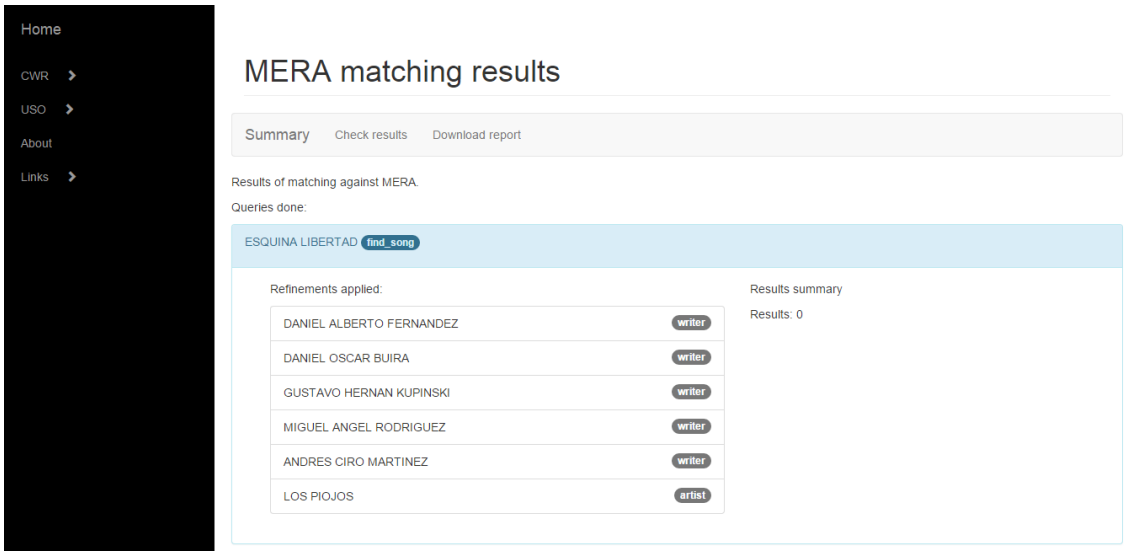


Figura 32.8. Ejemplo uso de interfaz de datos de resultado de conciliación

1.34 Manual del Programador

En este manual debemos describir cualquier aspecto que pueda ayudar a otros programadores a ampliar, modificar o entender aspectos de la construcción de nuestra aplicación. Debemos por tanto hacer una descripción general de los distintos aspectos involucrados en la construcción del sistema que puedan ser más difíciles de entender y también describir los procedimientos necesarios para hacer ciertas ampliaciones que hayamos contemplado en el diseño del sistema (añadir nuevas entidades, nuevos atributos a entidades existentes, nuevos servicios que usen a los ya desarrollados, modificaciones en la interfaz, etc.).

Conclusiones y Ampliaciones

1.35 Conclusiones

Aunque en términos generales la aplicación cumple con lo esperado, existen muchos aspectos en los que se puede mejorar.

El cliente ha pedido un sistema que se encargue de procesar datos de entrada y realizar labores de conciliación con ellos. Y este aspecto se ha cumplido, pero los problemas que sufre al manejar grandes volúmenes de datos es una enorme traba para su aplicación real, ya que los datos pueden llegar a ser varios gigas.

La aplicación requiere una mayor optimización en estos aspectos, y donde más falla es en el aspecto de procesar los ficheros de entrada. Esto se podría solucionar, en parte, utilizando la librería Ply para procesar estos ficheros, ya que es mucho más eficiente para esta labor.

Pero también sería necesario mejorar los servicios web para que puedan encargarse de varios procesos a la vez, teniendo una cola de datos a procesar o a conciliar. De esta manera podría controlarse mejor la carga del sistema.

También se habría facilitado mucho la labor de desarrollar el cliente web si se hubiese hecho uso de un CMS. Está opción había sido rechazada inicialmente, ya que se buscaba un cliente web lo más ligero que fuera posible, pero con el tiempo se ha acabado viendo que habría facilitado el desarrollo del cliente web.

Donde el proyecto ha fallado ha sido en el sistema para validar los resultados de la conciliación y realimentar el sistema de conciliación. Por falta de tiempo, y problemas de integración, no ha sido posible llevar a cabo este punto.

Aun así, el proyecto ha sido muy provechoso en el plano personal, ya que ha servido para aprender el lenguaje Python, no solo en el ámbito de las aplicaciones web, desarrollando el cliente web y los servicios web, si no también en el ámbito de los parsers, al haber sido necesario desarrollar uno para procesar los ficheros de entrada.

1.36 Ampliaciones

Existen varias ampliaciones que no ha sido posible añadir por falta de tiempo, entre ellas:

- Edición manual de los datos utilizados por el sistema de conciliación, de manera que sea posible que el usuario los introduzca
- Optimización del parser de CWR
- Uso de un CMS. Esta opción se rechazó inicialmente ya que se buscaba una infraestructura lo más ligera posible, pero facilitaría mucho el uso de la interfaz utilizar un CMS como por ejemplo Quokka Flask CMS.
- Mejorar el sistema de almacenamiento de datos, para utilizar una base de datos Mongo

Presupuesto

El proyecto tiene una duración de cinco meses, con cuatro horas de trabajo diarias, dando una duración total de seiscientas horas de trabajo.

Meses	Trabajo diario	Horas totales
5	4h/día	600h

Los precios aplicados son los siguientes:

- Analista: 35€ por hora
- Desarrollador: 15€ por hora

El desglose detallado de coste es el siguiente:

Concepto	Cantidad	Precio Unitario	Total
Recursos humanos			
Analisis			
Fase de analisis	96h	50,00 €/h	4.800,00 €
Desarrollo			
Fase de diseño	71h	35,00 €/h	2.485,00 €
Fase de desarrollo	103h	35,00 €/h	3.605,00 €
Fase de despliegue	49h	35,00 €/h	1.715,00 €
Fase de corrección	107h	35,00 €/h	3.745,00 €
Otros recursos			
Costes de infraestructura			
Servidor	5 meses	20,00 €/mes	100,00 €
Conexión internet	5 meses	65,00 €/mes	325,00 €
Alquiler oficina	5 meses	640,00 €/mes	3.200,00 €

Subtotal	21.145,00 €
Total (IVA 21%)	4.440,45 €
Total	25.585,45 €

Referencias Bibliográficas

1.37 Libros y Artículos

Libros y artículos usados de alguna forma durante el desarrollo del proyecto o su documentación.

Referencia	Descripción
Especificaciones funcionales de CWR	Manual que describe el standard CWR y la estructura que han de seguir los ficheros CWR

1.38 Referencias en Internet

Páginas Web consultadas para cualquier aspecto relacionado con el desarrollo del sistema o su documentación.

Referencia	Dirección
Manual de Flask	http://flask.pocoo.org/
Manual de Flask Restful	https://flask-restful.readthedocs.org/
Manual de Pyparsing	https://pyparsing.wikispaces.com/
Manual de referencia rápida para Pyparsing	http://infohost.nmt.edu/tcc/help/pubs/pyparsing/web/index.html

Apéndices

1.39 Glosario y Diccionario de Datos

Por orden alfabético, todos los términos que se consideren importantes en la aplicación con una descripción breve de su significado dentro de la aplicación.

- **Conciliación**: Proceso para vincular dos conjuntos de datos.
- **CWR**: Common Works Registration, standard para ficheros de transacciones usado por la industria discográfica.
- **Matching**: Término en inglés para la conciliación.
- **Parsing**: Termino en inglés para el procesamiento de ficheros.
- **Procesamiento de ficheros**: Procedimiento para adquirir los datos contenidos en un fichero.
- **Servicio Web**: Patrón estructural que permite intercambiar datos entre aplicaciones por medio del uso de protocolos.

1.40 Contenido Entregado en el Adjunto

1.40.1 Contenidos

Se incluye el código de los distintos componentes del proyecto en el adjunto de código.

Hay que tener en cuenta que no es posible incluir todo el código, ya que algunos datos son privados, y no puede permitirse el acceso a quienes no formen parte del proyecto.

Estos datos son :

- Proyecto con el modelo de ficheros USO
- Fichero ttl con datos para el servicio web de conciliación de MERA

Lo que provoca el problema de que es posible ejecutar el servicio de MERA en local.

1.40.1.1.1 Introducción

Se han incluido los siguientes componentes del proyecto, cada uno es un proyecto de código de Python, que puede examinarse en cualquier IDE que soporte este lenguaje:

- Servicio web de administración (carpeta WS-Admin)
- Modelo y parser CWR (CWR-DataApi)
- Cliente web (Frontend)
- Servicio web de conciliación (WS-MERA)
- Servicio web de procesamiento de ficheros CWR (WS-Validator)

1.40.1.1.2 Estructura general de los proyectos de código

Directorio	Contenido
./ Directorio raíz del CD	Contiene el fichero README.rst con el leeme, una serie de ficheros de configuración de Python (requirements.txt, setup.cfg, setup.py), que sirven para preparar el despliegue, el fichero de configuración de pruebas tox.ini, y ficheros make.bat y Makefile.
./<nombre_proyecto>	Contiene el código del proyecto
./tests	Contiene el código de las pruebas automatizadas
./docs	Contiene documentación del proyecto en Sphinx
./.scripts	Scripts utilizados para labores de desplieuge o

	testeo automatizado

1.40.2 Código Ejecutable e Instalación

Todos los proyectos están preparados para su rápida instalación, solo requieren Python y pip. Para esto se puede utilizar el comando “make install” o “python setup.py -install”.

Para asegurarse de que se instalan las dependencias correctamente se puede utilizar el comando “pip install -rrequirements.txt”.

1.40.3 Ficheros de Configuración

1.40.3.1 CWR

Los ficheros CWR, que viene de Common Works Registration, son ficheros de transacciones que utiliza la industria discográfica para intercambiar datos sobre las licencias que poseen.

Estos indican cosas como los nuevos contratos que se han firmado, o actualizaciones de los datos referentes a obras poseídas por una empresa.

Este standard lo mantiene CISAC (http://www.cisac.org/), pero tiene una serie de problemas, debido a seguir una estructura incómoda, y estar lleno de variantes y correcciones para problemas específicos.

1.40.3.1.1 Estructura del fichero

1.40.3.1.1.1 Registros

El fichero se compone de una serie de entidades básicas llamadas registros, cada una de las cuales ocupa una sola linea. La longitud de cada registro es variable, y lo compone una serie de campos que no poseen ningún delimitador, identificándose solo por la columna en que empiezan y acaban.

Esto hace que a primera vista sea dificil saber que contiene un registro, aunque mantienen cierta estructura común, ya que comienzan con un código que indica el tipo de registro y su posición en el fichero.

Un ejemplo de registro con datos de un contrato:

```
AGR00000001000000000000041383606100          OS2000060620170101201701O1N      D20170101      00001SYY
```

1.40.3.1.1.2 Transacciones

La mayor parte de los registros se agrupan en transacciones. Este tipo de estructura realmente es un registro principal, seguido por una serie de registros con información adicional.

Por ejemplo, esto es una transacción que incluye el territorio en que se aplica y las quienes lo han firmado:

```
AGR00000000000000000000023683606100          OS2003111820131118201311l8N      D20131118      00009SYY

TER000000000000000002136

IPA0000000000000001AS00261661375000000000000001183606  ITALIAN                                GILBERTI DUANTE
61 0500061 0000061 00000

IPA0000000000000002AC00250165006000000000000066                                        SOCIETY        MUSIC
61 0500061 1000061 10000
```

La composición de las transacciones sigue una serie de reglas que permiten variabilidad. Siempre empiezan con un registro que define el tipo de la transacción, y lo siguen una serie de combinaciones de registros adicionales, que normalmente son opcionales. Esto hace que en un fichero la longitud de las transacciones pueda fluctuar entre tres y quince lineas, por ejemplo, y algunos tipos de transacciones no tienen longitud máxima.

1.40.3.1.1.3 Grupos

A su vez, las transacciones forman grupos. Existe un grupo de transacciones por cada tipo de transacción (el tipo lo indica el primer registro de cada transacción).

Estos grupos se inician con un registro de cabecera de grupo, y terminan con un registro de cola de grupo.

Un ejemplo de registro de cabecera de grupo:

```
GRHAGR0000102.100130400001
```

Un ejemplo de registro de cola de grupo:

```
GRHAGR0000102.100123456789
```

1.40.3.1.1.4 Transmisión

De nuevo, los grupos son agrupados, esta vez dentro de una transmisión. Solo existe una transmisión en todo el fichero, que contiene todos los grupos.

También está delimitado por un registro de cabecera y por uno de cola.

Un ejemplo de registro de cabecera de transmisión:

```
HDRSO000000020ABCD                    01.102013011111110120130111
```

Un ejemplo de registro de cola de transmisión:

```
TRL0000200000053200005703
```

1.40.3.1.2 Complicaciones del standard

Existen numeras complicaciones a la hora de procesar el fichero, como la estructura con cierta frecuencia arbitraria, y las numerosas reglas de validación, también con frecuencia arbitrarias o poco claras.

Transacciones en las que al vez pueden haber solo dos autores y múltiples autores, o la poca claridad a la hora de decidir que valor se ha de poner en campos vacíos son dos ejemplos.

El segundo, de hecho, lleva a uno de los problemas más graves del standard, y es que cada empresa tiene su propia interpretación del fichero, con lo que realmente existen numerosas variantes contradictorias de este standard.

1.40.3.2 USO

Los ficheros USO son utilizados internamente por BMAT, y contienen los resultados de las búsquedas que realizan a través de diversos medios.

En la práctica, son ficheros DSV, con unas pocas particularidades, como el hecho de que algunas columnas pueden contener listas (donde los valores se separan con el carácter '|'), y que no se puede asegurar que dos ficheros contengan las mismas columnas.

1.40.3.3 Configuración procesador de CWR

Para poder adaptar el procesamiento de ficheros a las variantes de CWR se utiliza un fichero de configuración, que el procesador de ficheros utilizará para adaptar la gramática que utiliza para procesar estos ficheros.

El fichero de configuración utiliza un DSL sencillo, donde por ejemplo un registro se puede definir así:

```
transaction_record:
    id: agreement
    head: AGR
    rules:
      [
        field: submitter_agreement_n
        field: international_standard_code
        field: agreement_type
        field: agreement_start_date
        field: agreement_end_date
        field: retention_end_date
        field: prior_royalty_status
        field: prior_royalty_start_date
        field: post_term_collection_status
        field: post_term_collection_end_date
        field: date_of_signature
        field: number_of_works
        field: sales_manufacture_clause
        field: shares_change
        field: advance_given
        field: society_assigned_agreement_n
      ]
```

Este es un caso sencillo, que simplemente define el identificador que usarán las otras reglas, la cabecera que utilizará el registro y los campos que lo componen. Estos campos se definen un un fichero Python.

Pero también permite definir reglas más complejas:

```
group:
    id: agreement_transaction
```

```
    rules:
      [
        record: agreement
        transaction: territory_information (at_least_1)
     ]

group:
    id: territory_information
    rules:
      [
        record: territory_in_agreement (at_least_1)
        record: ipa_information (at_least_2)
     ]
```

En este caso se ve como se define una transacción de nuevos acuerdos. Esta requiere un registro de acuerdos, que es el que se ha mostrado anteriormente, y despues ha de aparecer un grupo indicando la información del territorio al que afecta, y este grupo ha de aparece al menos una vez.

Este grupo con la información del territorio a su vez está compuesto de registros, que también pueden aparecer multiples veces.

1.40.3.3.1 Composición de una regla

Las reglas, como las que se acaban de ver, se definen de manera general de la misma manera. Todas contienen los siguientes campos:

- id: el identificador utilizado por las demás reglas para referirse a esta
- head: solo es necesario para registros. Indica el identificador de cabecera que usará este registro, y que identifica su función
- rules: las reglas que se aplican a esta regla.

La lista de reglas puede contener los siguientes tipos de reglas:

- Campos (field): son las reglas terminales, y son las únicas que se definen fuera del fichero de configuración. Están guardadas en un fichero Python.
- Registros (record): son reglas compuestas solamente por un número fijo campos.
- Grupos (group): son reglas que pueden poseer registros u otros grupos, y el número de estos es variable.

Realmente la diferencia entre registros y grupos es como los maneja el parser, ya que los registros los mapea a clases del modelo. Ambos pueden utilizar el mismo conjunto de reglas, pero es mejor mantener una diferenciación clara entre ellos.

1.40.3.3.1.1 Modificadores

Adicionalmente existen una serie de modificadores que se pueden aplicar a las distintas reglas. Por ejemplo:

```
record: territory_in_agreement (at_least_1)
```

El at_least_1 indica que ha de aparecer una o más veces.

Los diferentes modificadores que se aceptan son:

- `grouped: el resultado se devolverá agrupado en una lista. Solo tiene sentido si la regla puede aparecer múltiples veces`
- `at_least_x: la regla puede aparecer multiples veces, pero ha de aparecer al menos X veces`
- `optional: la regla puede aparecer una o cero veces. Este modificador ha de usarse para indicar grupos que son opcionales`