Universidad Zaragoza

Trabajo Fin de Grado

Bizi2Data. Apertura para la investigación de un sistema legado.

Bizi2Data. Opening for the investigation of a legacy system.

Autor

Daniel Cabrera Ebana

Director

Francisco Javier López Pellicer

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2017

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Daniel Cabrera ebana,

con nº de DNI 44735666E en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Bizi2Data. Apertura para la investigación de un sistema legado.

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 22 de Noviembre de 2017

Fdo: Daniel Cabrera Ebana

AGRADECIMIENTOS

Mi más sincero agradecimiento y cariño a mi familia y amigos por su apoyo incondicional durante estos años de formación. En especial a mis padres por sus palabras de ánimo en los momentos complicados y por hacer posible que haya tenido la posibilidad de continuar mi formación durante estos años.

También agradezco a todos los compañeros y amigos con los que me he cruzado durante mi etapa académica, que me han ofrecido sus diferentes puntos de vista a lo largo de estos años y que me han aportado aprendizajes más allá del puramente académico.

Sin duda para lograr la consecución de este proyecto tengo que agradecer la labor de *Fco. Javier López Pellicer*, que no solo ha ejercido su labor de director del proyecto, sino que también ha desempeñado una labor de enseñanza que considero muy enriquecedora para mi futuro laboral.

Por último, agradecer a todos los profesores que me han impartido asignaturas durante estos años de carrera.

Resumen ejecutivo

El proyecto ***Bizi2Data*** nace para cubrir la necesidad de disponer de una infraestructura capaz de recolectar, procesar y almacenar datos referentes al servicio *Bizi Zaragoza* del *Ayuntamiento de Zaragoza*, con la finalidad de que estos puedan ser explotados de diversas maneras.

El problema que se aborda en este proyecto es la dificultad que tiene el *GEOT*, grupo de investigación perteneciente a la *Universidad de Zaragoza*, para el acceso y estudio de los datos referentes a este servicio. El *GEOT* facilita para la realización del proyecto el acceso a la web *Smartbike Report Manager* (`http://reportingportal-zar.clearchannel.com`) que es la principal fuente de datos del servicio *Bizi Zaragoza* y que dispone de un gran volumen de información diaria e histórica del servicio. Esta web sólo está pensada para la generación de determinados informes y no ofrece un servicio de descarga de datos como servicio. La descarga manual de esta información exige un proceso complejo y tedioso de realizar, ya que la web no está diseñada para ello.

Por lo tanto el proyecto *Bizi2Data*, para dar una solución a este problema, ha desarrollado una infraestructura capaz de realizar las tareas periódicas de extración de la información de *Smartbike Report Manager* para posteriormente almacenarla de forma que pueda ser expuesta en una nueva aplicación web donde es posible consultar y descargar la información de una manera más sencilla, confortable y mejor estructurada.

Desarrollar *Bizi2Data* ha sido una oportunidad para crear un proyecto funcional, robusto y escalable, que pueda servir de base a futuros desarrollos que integren todo tipo de información relacionada con el servicio *Bizi Zaragoza*. Esto implica que el proyecto se centre en la creación de la plataforma y en un proceso de extracción/almacenamiento/tratamiento/explotación de referencia que pueda ser reutilizado y ampliado en un futuro. Además condiciona decisiones de diseño derivadas de las expectativas sobre el volumen futuro de información que se ha de almacenar. De esta forma, el sistema desarrollado puede servir de base para futuros proyectos, ya sean desarrollados como Trabajo Fin de Grado o como proyectos de colaboración con *GEOT*, que puedan aportar nuevas funcionalidades o expandan las capacidades actuales.

Índice

Glosario

AngularJS (`https://angularjs.org`) *Framework* de *JavaScript* que se utiliza para crear y mantener aplicaciones web dinámicas [1].

Apache Hadoop (`http://hadoop.apache.org`) Framework que permite el almacenamiento y procesamiento de grandes volumenes de datos a partir del paradigma MapReduce.

Apache Hive (`https://hive.apache.org`) El software de almacenamiento de datos Apache Hive facilita la lectura, escritura y administración de conjuntos de grandes volumenes de datos que residen en el almacenamiento distribuido mediante SQL.

Apache POI (`https://poi.apache.org`) Proporciona bibliotecas *Java* para leer y escribir archivos en formatos de *Microsoft Office*, como *Word*, *PowerPoint* y *Excel* [2].

Apache Sqoop (`https://sqoop.apache.org`) Herramienta diseñada para transferir de forma eficiente grandes volumenes de datos entre *Apache Hadoop* y almacenes de datos estructurados, como bases de datos relacionales.

Bootstrap (`https://getbootstrap.com`) *Framework* o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en *HTML* y *CSS*, así como, extensiones de *JavaScript*, opcionales, adicionales [3].

BPMN (`http://www.bpmn.org/`) *Business Process Model and Notation*, es una notación gráfica estandarizada que permite el modelado de procesos en un formato de flujo de trabajo.

ChromeDriver Servidor independiente que implementa el protocolo de conexión de *WebDriver* para *Chromium*. *WebDriver* es una herramienta de código abierto para la prueba automatizada de aplicaciones web en muchos navegadores.

Proporciona capacidades para navegar a páginas web, entradas de usuario, ejecución de *JavaScript* y más [4].

Cron En el sistema operativo *Unix*, *cron* es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab [5].

Docker (`https://www.docker.com`) Permite el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel de sistema operativo en Linux [6].

ETL *Extract, Transform and Load* son los procesos que permiten la extracción de información desde fuentes de datos, el tratamiento de la información extraida para el cálculo de nuevos valores y limpieza de posibles datos problemáticos y por último la carga de los datos transformados en el almacen de datos [7].

GEOT *Grupo de Estudios de Ordenación del Territorio* perteneciente al *Instituto Universitario de investigación en ciencias ambientales de Aragón* (`http://iuca.unizar.es/`) (*Universidad de Zaragoza*). Sus proyectos están ligados a la ordenación y análisis territorial, social y sanitario, las infraestructuras de transportes, los equipamientos públicos, y a los problemas de insustentabilidad y de segregación social, económica y ambiental asociados a los modelos recientes de la expansión de las ciudades.

JHipster (`http://www.jhipster.tech`) Generador de aplicaciones libre y de código abierto usado para desarrollar rápidamente una aplicación web moderna usando *Angular* y *Spring Framework* [8].

MoSCoW Técnica de priorización utilizada en la gestión, análisis de negocios, gestión de proyectos y desarrollo de software para llegar a un entendimiento común con las partes interesadas sobre la importancia que otorgan a la entrega de cada requisito [9]. El término *MoSCoW* es un acrónimo derivado de la primera letra de cada una de las cuatro categorías de priorización (*Must have, Should have, Could have, and Won't have*).

REST La Transferencia de Estado Representacional (en inglés *Representational State Transfer*) o *REST* es un estilo de arquitectura software para sistemas hipermedia distribuidos como la *World Wide Web* [10]. En la actualidad se usa para describir

cualquier interfaz entre sistemas que utilice directamente *HTTP* para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (*XML*, *JSON*, etc) sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo *SOAP*.

Selenium (`http://www.seleniumhq.org/`) Entorno de pruebas de software para aplicaciones web. *Selenium* provee una herramienta de grabar/reproducir para crear pruebas sin usar un lenguaje de scripting para pruebas (*Selenium IDE*) [11]. Incluye también un lenguaje específico de dominio para pruebas (*Selanese*) para escribir pruebas en un amplio número de lenguajes de programación populares incluyendo *Java, C#, Ruby, Groovy, Perl, Php y Python*. Las pruebas pueden ejecutarse entonces usando la mayoría de los navegadores web modernos en diferentes sistemas operativos como *Windows*, *Linux* y *OSX*.

Spring Framework (`https://spring.io`) Proporciona un modelo de programación y configuración integral para aplicaciones empresariales modernas basadas en *Java*, en cualquier tipo de plataforma de implementación [12].

Swagger (`https://swagger.io`) *Software* de código abierto que implementa el estándar *Open API* (`https://www.openapis.org`). Dispone de un conjunto de herramientas que dan soporte para documentación automatizada, generación de código y generación de casos de prueba.

Capítulo 1

Bizi2Data

1.1. Introducción

El proyecto ***Bizi2Data*** consiste en la creación de un sistema de información para facilitar el estudio de movilidad del servicio de bicicletas de la ciudad de *Zaragoza*. El proyecto viene derivado de una colaboración entre el *Ayuntamiento de Zaragoza* y el grupo de investigación *GEOT*, perteneciente a la *Universidad de Zaragoza*, para el estudio de movilidad sostenible dentro del área metropolitana de la ciudad de *Zaragoza*.

Bizi2Data busca dar solución a los problemas transmitidos por el *GEOT* acerca de sus estudios sobre los datos del servicio *Bizi Zaragoza*. Estos problemas vienen derivados del uso de la web *Smartbike Report Manager* (`http://reportingportal-zar.clearchannel.com`), que dispone de un gran volumen de información diaria e histórica del servicio. Esta web sólo está pensada para la generación de determinados informes y no ofrece un servicio de descarga de datos como servicio. La descarga manual de esta información exige un proceso complejo y tedioso de realizar, ya que la web no está diseñada para ello. Por este motivo, el grupo *GEOT* ha buscado apoyo técnico, que se ha materilizado en este Trabajo Fin de Grado (TFG), para que se desarrolle una infraestructura capaz de realizar la extracción periódica de los datos disponibles en *Smartbike Report Manager*, para su posterior almacenamiento explotación de una forma más cómoda y eficaz.

Durante el desarrollo del proyecto se han estudiado diferentes alternativas para alcanzar una solución adecuada que cubra las necesidades del *GEOT*. Sin embargo por la naturaleza del problema que se aborda: sistema legado cuya funcionalidad y tecnologías pueden ser modificadas en un futuro próximo en función de lo que pase con la concesión de *Bizi Zaragoza*, posibilidad de que existan otras fuentes de información que en un futuro deberían mezclarse con los datos que se extraigan ahora, previsión

que los datos que se tienen que extraer pueden tener un volumen muy grande o pueden acabar siendo mezclados con datos de igual o mayor volumen. Por ello, se ha optado por desarrollar el proyecto siguiendo los principios de las metodologías ágiles, con el objetivo de crear un proyecto centrado en la creación de la plataforma y en un proceso de extracción/almacenamiento/tratamiento/explotación de referencia que pueda ser reutilizado y ampliado en un futuro ya sea con TFG de continuación o mediante proyectos de colaboración.

El resultado del proyecto se caracteriza por disponer de una infraestuctura que potencialmente es capaz de perdurar en el tiempo, por estar basada en tecnologías que han sido elegidas pensando en las necesidades futuras del proyecto. Siendo posible la escalabilidad o la realización de modificaciones de forma sencilla. Y como se ha comentado en el párrafo anterior, este resultado puede ser continuado mediante futuros proyectos con el grupo de investigación *GEOT* o mediante la realización de nuevos TFG.

1.2. Objetivo del proyecto

Los objetivos de este proyecto vienen, en gran medida, marcados por las necesidades y dificultades transmitidas por el grupo de investigación *GEOT*. Se encuentran con la necesidad de disponer de la información del servicio de bicicletas de Zaragoza (*Bizi Zaragoza*) para realizar estudios de movilidad, pero su única opción para hacerlo es a partir de una plataforma web legada, *Smartbike Report Manager*. Esta plataforma web les genera un alto grado de dificultad a la hora de obtener la información y realizar análisis con los datos obtenidos.

A partir de estas necesidades, el proyecto ***Bizi2Data*** nace para construir un sistema de información capaz de extraer, tratar y alojar tanto los datos históricos, como los datos que pueden ser obtenidos en el futuro, del servicio *Bizi Zaragoza*, a partir de la web *Smartbike Report Manager*, facilitando las tareas de obtención y análisis sobre los datos.

Como punto de partida, a la hora de comenzar el proyecto se han marcado los siguientes objetivos como principales:

- Creación de una base de datos histórica con datos del servicio *Bizi Zaragoza*, y continuamente actualizada.

- Recuperación y almacenamiento de datos publicados en un servicio web legado.
- Recopilación de información diaria de forma automática.
- Permitir consultas sobre la información disponible, con granularidad de una hora.
- Enriquecimiento de la información obtenida antes de ser expuesta.
- Disponer de una aplicación web donde se expongan todos los datos obtenidos.
- Permitir que los datos expuestos puedan ser descargados, con un formato válido y bien formado, desde la aplicación web.

1.3. Alcance

Antes de definir el alcance que debe tener el proyecto *Bizi2Data*, se ha realizado un análisis preliminar sobre las peticiones realizada por el grupo *GEOT*. A partir de este análisis preliminar, se ha llegado a la conclusión de que sus espectativas son demasiado ambiciosas para abarcarlas todas dentro un TFG, tal como se ha adelantando en la introducción. Por esto, como se ha comentado anteriormente, este proyecto se realiza entendiendo que no se va a poder cubrir todas las necesidades actuales de *GEOT*, pero se ve como una oportunidad para generar una solución escalable que sirva para futuros proyectos.

Tras ese análisis preliminar más el análisis realizado de la plataforma web legada (Sección 2.1) y los datos extraidos de ella (Anexo A), una primera captura de requisitos (Sección 2.2), el estudio de alternativas y viabilidad (Sección 2.3) y un análisis de los riesgos potenciales del proyecto (Sección 2.4), se decide que el alcance de este proyecto se centre en la creación de la plataforma y en un proceso de extracción/almacenamiento/tratamiento/explotación de referencia que pueda ser reutilizado y ampliado en un futuro. Esta decisión se ha tomado siguiendo la filosofía del método *MoSCoW* (Sección 2.5), obviando los requisitos secundarios, que pueden ser aportados a futuras soluciones partiendo de esta.

Debido a la descripción inicial del proyecto, existe la restricción de que los datos principales deben ser recuperados únicamente del sistema web legado que se ha proporcionado, *Smartbike Report Manager*. Por este motivo, el proceso de referencia se implementará para extraer la colección de datos que se consideran los más relevantes que en este momento son se pueden extraer del *Smartbike Report Manager*, en particular el conjunto de datos “Uso de Estaciones”. El diseño del proceso tendrá en cuenta que

deberá poderse reutilizar en futuros proyectos para extraer otros datos que más adelante puedan ser considerados relevantes de la plataforma *Smartbike Report Manager*.

Capítulo 2

Análisis del sistema

Para realizar un análisis que posteriormente sirva para la toma de decisiones en el diseño del proyecto se han realizado distintos estudios. Se ha llevado a cabo una descripción de la situación actual (Sección 2.1) donde se ejecuta un análisis global de la web *Smartbike Report Manager* (`http://reportingportal-zar.clearchannel.com/`) buscando puntos claves que puedan propiciar problemas en el desarrollo de la solución, a continuación se ha realizado una captura de posibles requisitos del sistema (Sección 2.2) siguiendo la metodología *MoSCoW*. Para determinar que requisitos de los planteados son viables asumir en este proyecto, se han evaluado las distintas alternativas para desarrollar la solución y su viabilidad (Sección 2.3) y por último, entendiendo las dificultades que plantea un proyecto de esta envergadura, se han definido los requisitos finales que deben estar presentes en la solución al finalizar este Trabajo Fin de Grado.

2.1. Descripción de la situación actual

Se cuenta con la web *Smartbike Report Manager* en la cual se puede consultar información acerca del servicio de bicicletas de Zaragoza (*Bizi Zaragoza*). Este sitio web presenta una vista incompleta del verdadero sistema de almacenamiento (Figura 2.1) ya que solamente ofrece los datos desde hace dos años hasta el día actual.

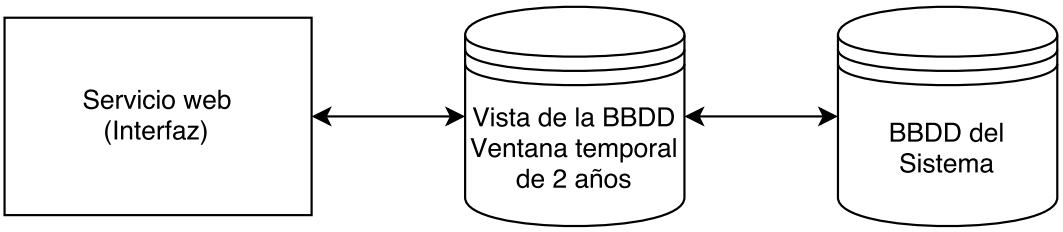

Figura 2.1: Visión simplificada de la posible arquitectura del sistema legado

Aunque se exponga una visión incompleta de la información, a través del análisis de los datos realizado (Anexo A) se ha podido comprobar que cuenta con un gran volumen de información de todo tipo. Mediante este análisis se ha llegado a la conclusión de que la colección de datos más interesante es "Uso de Estaciones".

En cuanto al portal web *Smartbike Report Manager* es necesario usar credenciales para el acceso (Figura 2.2), en este caso se dispone de las credenciales de un tercero. Esto puede ser un riesgo a la hora de realizar muchos accesos continuados a la web para descargar la información, ya que el servidor podría entender que se trata de un uso fraudulento del servicio.

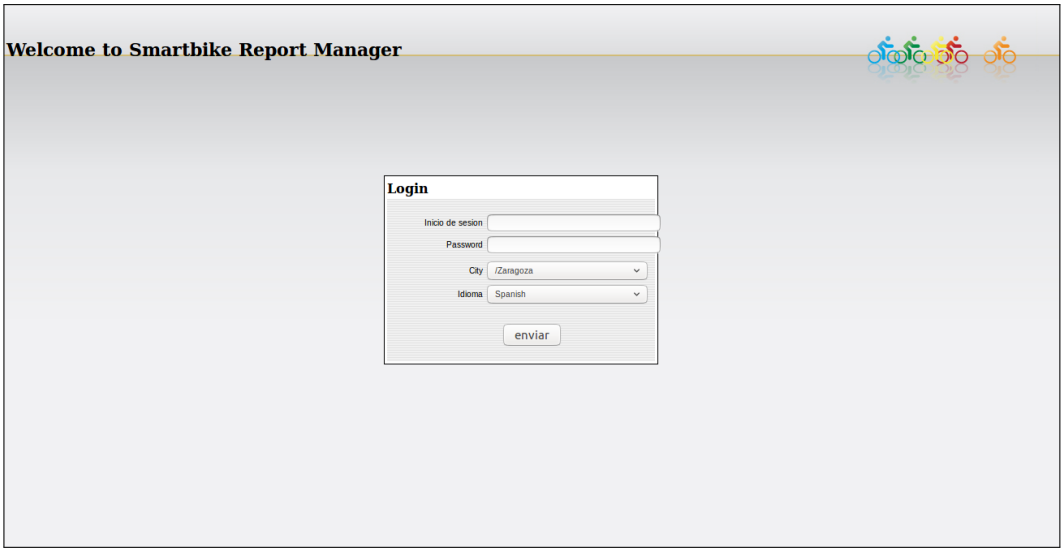

Figura 2.2: Interfaz de inicio de *Smartbike Report Manager*

La web cuenta con una interfaz simple, con algunos errores de diseño, que hacen que la navegabilidad sea más compleja. Cuenta con varias categorías principales, que a la vez se dividen en subcategorías según el tipo de resultados que aporta. Para el acceso a la información, el usuario debe rellenar varios campos de búsqueda, que según en que categoría se encuentre serán unos u otros (Figura 2.3). Estos campos de búsqueda, en algunos formularios, no siguen el estándar de nombrado aunque su finalidad sea la misma. A esto hay que añadir que existen entradas, para los campos de búsqueda, que no funcionan correctamente y hacen que el sistema devuelva errores.

Se trata de una web compleja de manejar, ya que no existe ningún apartado donde se pueda encontrar ayuda sobre el significado de cada valor de entrada, y en la que también se encuentra que hay datos malformados o que, aparentemente, no han sido tratados y/o comprobados antes de ser mostrados o almacenados.

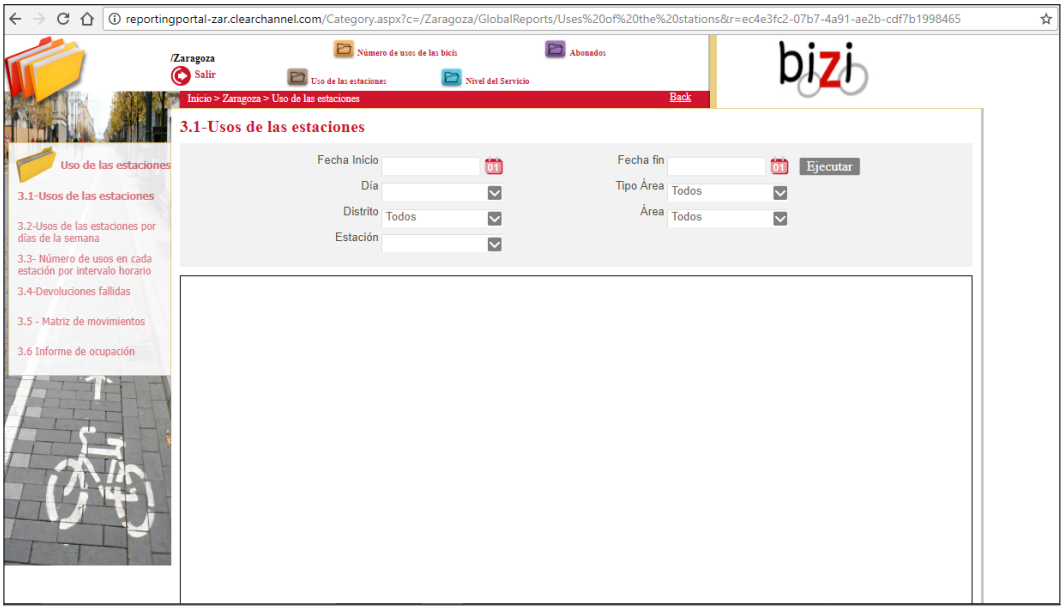

Figura 2.3: Interfaz web de *Smartbike Report Manager*

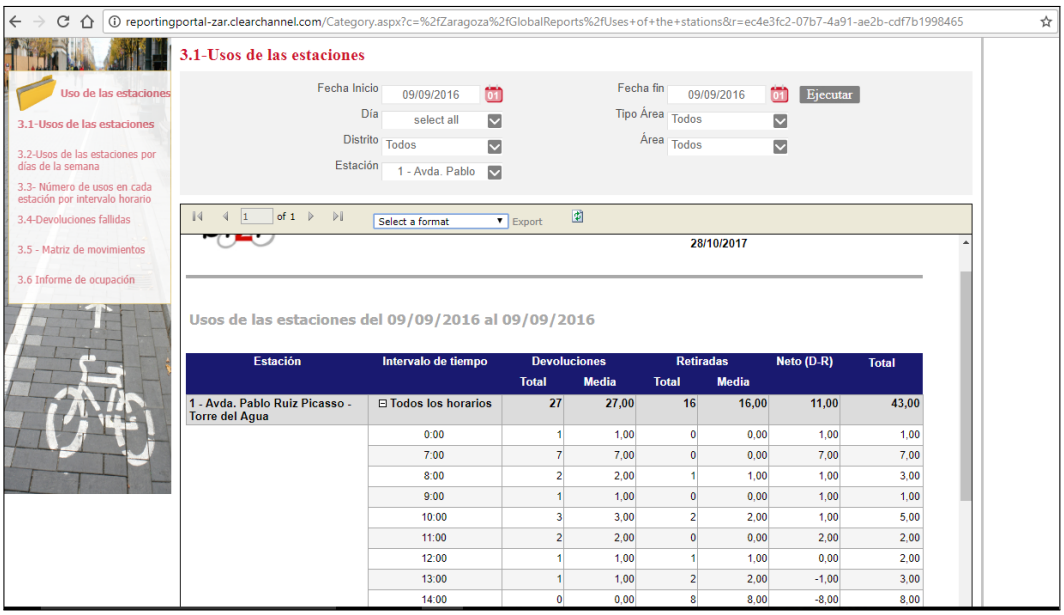

Figura 2.4: Resultado de búsqueda en *Smartbike Report Manager*

Tras obtener los resultados de búsqueda (Figura 2.4), existe la posibilidad de descargar los datos en distintos formatos; *XML*, *CSV*, *TIFF*, *PDF*, *MHTML* y formato de hoja de cálculo *Excel*.

En el caso de la descarga en *CSV* y *XML*, siendo aparentemente las opciones más factible de utilizar en este proyecto por su carácter estructurado, se detecta que al descargarlos no contienen la información generada en la web, por lo que no se

puede contar con estas opción. En el caso de los formatos *TIFF*, *PDF*, *MHTML* no se cree conveniente trabajar con ellos ya que no ofrecen una posibilidad de trabajar a partir de la información que contienen. Por lo que la única opción viable es descargar los ficheros en formato de hoja de cálculo *Excel*, porque dispone de la información completa y parcialmente estructurada (Anexo C).

En conclusión, se dispone de un sistema web legado con credenciales de acceso, en el cual la navegabilidad es poco intuitiva, la recuperación de información es engorrosa y, en el caso de algunos formatos, la información obtenida, en los ficheros descargados, no es la misma que se muestra en la interfaz o, directamente, no contienen ninguna información.

2.2. Captura de posibles requisitos

2.2.1. Metodología

Como primera aproximación para la captura de requisitos del sistema se ha utilizado como referencia la metodología *MoSCoW*. El método *MoSCoW* es una técnica de priorización de requisitos basada en el hecho de que aunque todos los requisitos se consideren importantes es fundamental destacar aquellos que permiten darle un mayor valor al sistema, lo que permite enfocar los trabajos de manera más eficiente [13]. Lo que la diferencia de otras técnicas tradicionales como por ejemplo calificar los requisitos como de prioridad alta, media o baja es que en este caso la escala utilizada tiene un significado intrínseco, de manera que el usuario responsable de asignar la prioridad conoce el efecto real que producirá su elección.

- ***M (Must) o Debe:*** Requisito que tiene que estar implementado en la versión final del producto para que la misma pueda ser considerada un éxito.
- ***S (Should) o Debería:*** Requisito de alta prioridad que en la medida de lo posible debería ser incluido en la solución final, pero que llegado el momento y si fuera necesario, podría ser prescindible si hubiera alguna causa que lo justificara.
- ***C (Could) o Podría:*** Requisito deseable pero no necesario, se implementaría si hubiera posibilidades presupuestarias y temporales.
- ***W (Won't) o No hará:*** Hace referencia a requisitos que están descartados de momento pero que en un futuro podrían ser tenidos de nuevo en cuenta y ser reclasificados en una de las categorías anteriores. Estos requisitos podrían ser abordados en futuros Trabajos Fin de Grado.

Esta clasificación puede ser modificada durante el proceso de desarrollo y definirse, en el caso de desarrollos iterativos incrementales, prioridades a nivel de iteración.

2.2.2. Primera captura

Para definir los requisitos del sistema se ha utilizado el método *MoSCoW* como apoyo. Estos requisitos se generan a partir de las conversaciones previas que se han tenido con el *GEOT*. Esta captura de requisitos se ha realizado entendiendo que las peticiones realizadas por el *GEOT* son demasiado ambiciosas para un único Trabajo Fin de Grado.

El resultado del proyecto **DEBE** ser capaz al final de esta iteración:

- Recuperar y almacenar a largo plazo la colección de datos "Uso de Estaciones" en *Smartbike Report Manager* para cada estación y con granularidad de una hora (si está disponible) en el formato estructurado (si está disponible) original.
- Ofrecer un proceso replicable para recuperar todos los datos históricos disponibles en *Smartbike Report Manager* de dicha colección.
- Ofrecer un proceso replicable para recuperar cada dia todos los datos nuevos disponibles en *Smartbike Report Manager* de dicha colección.
- Gestionar la recuperación de los datos mediante un sistema que permita ver el estado de la recuperación y tomar decisiones para recuperar de forma manual, automática y periódica los datos de dicha colección.
- Procesar todos los datos para ser expuestos mediante un modelo de datos relacional que permita realizar preguntas *SQL*.
- Exponer el modelo mediante un *API REST* que permita hacer preguntas *SQL*.
- Tener un portal que permita navegar por los datos, recuperar subconjuntos de información y visualizar, en formato tabular y espacial, información sobre uso de *Bizi Zaragoza*.

El resultado del proyecto **DEBERÍA** ser capaz al final de esta iteración:

- Enriquecer el modelo con información espacial sobre las estaciones *Bizi* para permitir la realización de preguntas *SQL* espaciales.

– Utilizar un rutómetro y una red vial para estimar intensidades de trafico horarias y exponerlas mediante un *API REST* que permita hacer preguntas *SQL* analíticas y espaciales.

El resultado del proyecto **PODRÍA** ser capaz al final de esta iteración:

– Exponer el modelo mediante un *API REST* que permita hacer preguntas *SQL* espaciales y que devuelva las respuestas en formatos habituales usados por sistemas de información, tanto aplicaciones Web (*GeoJSON*) como de escritorio (*ESRI Shapefile*).

El resultado del proyecto **NO HARA** al final de esta iteración:

– Tener un portal que visualice en formato espacial información de trafico derivada del uso de *Bizi Zaragoza*.

– Recolectar otras colecciones de datos de la web *Smartbike Report Manager*.

2.3. Estudio de alternativas y viabilidad

Como punto de partida del proyecto, se desarrolla una prueba de concepto en la cual se estudian distintas posibilidades para obtener el resultado final. Esta prueba de concepto se divide en varias etapas en donde se realizan las pruebas pertinentes en busca de la opción más factible para abordar el problema. Esta prueba de concepto también sirve para definir finalmente los requisitos que serán implementados en la solución final. A continuación se enumeran los casos más relevantes.

2.3.1. Descarga de ficheros desde Smartbike Report Manager

En este punto de la prueba de concepto se analiza como se debería realizar las descarga de ficheros que se encuentran en el portal web legado. Debido a que para obtener estos ficheros es necesario realizar una navegación secuencial por la interfaz web y rellenar distintos campos de búsqueda, se decide utilizar la herramienta *Selenium* para estudiar como es el comportamiento de las automatizaciones en la web.

Con la herramienta *Selenium* se han realizado diferentes pruebas para encontrar la forma más correcta de realizar la navegación hasta llegar al objetivo. Durante las pruebas se detecta que se trata de una tarea compleja ya que muchos elementos de la web no disponen de identificadores por lo que es necesario recurrir a técnicas manuales

para hacer referencias a ellos.

Se detecta que en caso de querer obtener todos los distintos tipos de información disponibles en la web, sería necesario realizar *scripts* de navegación individuales para cada caso. Debido a esta situación y a que, en muchos casos, la información se encuentra replicada, se decide que la opción más efectiva es escoger la fuente de datos más relevante y diseñar un proceso completo de extracción que ofrezca la posibilidad de replicarlo en proyectos futuros para la recolección de otros datos.

Una vez realizado este estudio, se decide utilizar las librerías que ofrece *Selenium* para realizar un proceso en *Java* para navegar y descargar los ficheros históricos y diarios, de forma autónoma. Como se ha comentado anteriormente, debido a la complejidad de esta tarea, se decide que solo se realizará la recuperación de uno de los formularios, entendiendo que es el más relevante de todos, "Usos de estaciones", y que por ello puede servir de ejemplo para futuros procesos de extracción.

2.3.2. Tratamiento de ficheros descargados

Partiendo de la base de que el portal web legado ofrece una variedad de formatos de ficheros a descargar, en un principio se opta por obtener esos ficheros en formato *CSV*, ya que facilita la tarea de tratamiento y almacenamiento de los datos. Esta idea tuvo que ser descartada ya que al estudiar los ficheros descargados se observaba que no contenian los datos que en la web se mostraban, solo contenian las cabeceras. Una vez descartada esta opción, se decide optar por la descarga de ficheros en formato hoja de cálculo *Excel*, en el que se dispone de los datos correctamente estructurados.

Ya que la intención es introducir la información en la base de datos a partir de ficheros *CSV*, el tratamiento de los datos se realiza a partir de las herramientas disponibles en las librerías de *Apache POI*, con las cuales se obtiene el contenido de las hojas de cálculo y se insertan de manera estructurada en un fichero *CSV*, al cual se le añade información extra que pueda resultar de interés para la solución final.

En este caso se ha optado por la herramienta *Apache POI*, para la lectura y extracción de información de las hojas de cálculo, ya que es capaz de funcionar sin tener instalado *Microsoft Office*, al contrario que otras herramientas similares como *JExcelAPI*.

Las pruebas para la extracción de información de las hojas de cálculo han requerido

de un estudio previo de como está estructurado el fichero descargado (Anexo C.1), ya que utiliza una plantilla por la cual es dificil saber en que posición se úbica realmente la información, a parte en ficheros donde se dispone con más de un conjunto de datos diferentes ha sido necesario encontrar un patrón para diferencia en que momento termina un conjunto y cuando empieza el siguiente. Es evidente que ficheros procedentes de otros formularios requerirán sus propios mecanismo de tratamiento.

2.3.3. Almacenamiento de datos

Debido a la configuración inicial del proyecto, donde se cuenta con un gran volumen de información que necesita ser almacenada y se sabe que seguirá creciendo a lo largo del tiempo, el almacenamiento de los datos debe realizarse sobre un sistema de almacenamiento capaz de albergar una situación como esta. Por ello, se ha decidido probar *Apache Hadoop* y herramientas como *Apache Hive* y *Apache Sqoop*. La utilización de *Apache Hive* aporta a la solución final la posibilidad de realizar consultas *SQL*, además de tener una estructura que puede ser transferida a un *SGBD SQL*, como puede ser *PostgreSQL* o *MySQL*.

Como aproximación al número de ficheros disponibles en la web *Smartbike Report Manager* se ha realizado el siguiente cálculo teniendo en cuenta sólamente los formularios que devuelven información acerca de las estaciones de bicicletas. Esta aproximación sirve para entender porque es necesario contar con un sistema capaz de almacenar grandes volumenes de datos. Se dispone de 7 formularios para distintas búsquedas del uso de estaciones y 129 estaciones de bicicletas, la web *Smartbike Report Manager* alberga información de los dos últimos años que puede ser obtenida con granularidad de un día, por lo que se estima que es necesario realizar, aproximadamente, 660.000 descargas para generar el histórico del uso de estaciones, por lo que el crecimiento a lo largo del tiempo se prevé que sea cada vez más elevado teniendo en cuenta que en total existen 22 tipos de formularios distintos, que según los datos de entrada devuelven un resultado u otro. En cualquier caso, a diferencia de los dos procesos anteriores (extracción y transformación), todos los datos que lleguen a este proceso tendrán el mismo tratamiento cuando se almacenen.

En pruebas posteriores, referentes a la exposición de los datos en el nuevo servicio web, se decide que los datos a mostrar deben pasar a un *SGBD* intermedio, debido a que la herramienta utilizada para generar la web no admite, por el momento, *Apache Hive*. Para ello se estudia la viabilidad de utilizar *PostgreSQL* o *MySQL*, realizando el movimiento de datos con la herramienta *Apache Sqoop*. Se llega a la conclusión de

que ambos aportan las mismas características a la solución final y opta por utilizar *MySQL*. El uso de esta aproximación implica que aunque todos se almacenen de la misma forma, habrá que crear procesos de exportación con la herramienta *Apache Sqoop* comunes para todos los datos que procedan de un mismo formulario.

2.3.4. Despliegue del sistema

Para desplegar el sistema, en un entorno de desarrollo, se opta por estudiar la viabilidad de realizarlo con máquinas virtuales o utilizar *Docker*. El principal inconveniente de utilizar *Dockers* es el poco conocimiento previo de la plataforma, por lo que la curva de aprendizaje fue prolongada, mientras que ya se había trabajado con máquinas virtuales en proyectos anteriores.

Una vez alcanzado un conocimiento básico sobre el funcionamiento de *Dockers*, se comienza a la instalación de un contenedor en el que ya se disponía de *Apache Hadoop* y *Apache Hive*, lo que facilita el despliegue evitando tener que realizar configuraciones desde cero.

Para realizar las pruebas con *Docker* se ha utilizado *docker-compose* para configurar y generar las imagenes del sistema necesarias dentro del contenedor. Se encontraron problemas a la hora de acceder al contenido desde el exterior del *Docker*, ya que es necesario hacer un mapeo de puertos en el fichero de despliegue del sistema (docker-compose.yml), ya que de no hacerlo, es necesario realizarlo manualmente una vez que todo el sistema este montado. También fue necesario configurar una carpeta compartida entre el sistema host y el *Docker* ya que esto facilita cualquier tarea de movimiento de ficheros o inserción de datos desde el exterior.

Al buscar una solución similar con máquinas virtuales, donde se contase con *Apache Hadoop* y *Apache Hive*, se optó por probar las máquinas virtuales expuestas por la empresa *Hortonworks*. El problema principal de esta opción es la cantidad de recursos que consume, siendo imposible desplegarlo en cualquier máquina. Para poder realizar pruebas para este supuesto ha sido necesario solicitar un equipo informático extra, con mayor capacidad de memoria *RAM*.

Tras las pruebas se estudia el impacto que tiene cada una de las opciones (velocidad, escalabilidad, capacidad, tiempo de carga, espacio necesario en disco, consumo de memoria *RAM*), y se decide que la utilización de *Docker* es más factible para efectuar este proyecto. Decisión tomada en gran medida porque la máquina virtual consume

mayor espacio en disco y memoria y los tiempos de procesamiento son mucho más lentos. Esta decisión de utilizar *Docker* implica que a la hora de llevar la solución a un entorno en producción se va a disponer de sistema compacto en un entorno controlable y que puede ser fácilmente escalable dadas las cualidades de los contenedores *Docker*. No solo eso, cada vez es más común el soporte de Docker como herramienta de despliegue a producción.

2.3.5. Explotación de los datos almacenados

Para la explotación de los datos almacenado se desarrolla una aplicación web, y para ello se decide utilizar *JHipster*, ya que ofrece la comodidad de generar una aplicación web completa y que es suficiente para este proyecto, evitando la implicación de horas extra para realizar el desarrollo de una plataforma web desde cero.

Durante las pruebas se detecta un riesgo derivado de las opciones de almacenamiento disponibles con *JHipster*. que puede afectar al desarrollo normal del proyecto (Sección 2.4). El problema reside en que no es posible contectar la aplicación web directamente con los datos almacenados en *Hadoop*, por lo que se realizan pruebas para intentar llegar a una solución de forma manual para realizar la conexión. Tras varios intentos se decide que esta tarea se presenta demasiado compleja, dado que existe escasa información y la que existe no se consigue replicar para conseguir el resultado esperado.

Por lo tanto, se hacen pruebas con las distintas posibilidades que ofrece *JHipster* para la explotación de datos; *MySQL*, *PostgreSQL* y *MongoDB* entre otros, y se decide que para este proyecto se utilizará *MySQL* como base de datos para el consumo de información a partir de la aplicación web. Una ventaja de esta decisión es que será fácil crear un interfaz web de consulta y descarga de cualquier colección de datos que esté publicada en *MySQL*.

2.4. Análisis de riesgos

Al inicio del proyecto se conocía la existencia de riesgos que podían afectar al desarrollo normal de los objetivos planteados y se ha buscado la mejor solución para controlarlos, siempre que fuera posible. Durante el estudio de alternativas y viabilidad (Sección 2.3) aparecieron nuevos riesgos que tuvieron que ser estudiados para buscar una solución al problema que enmarcan.

A continuación se puede ver una lista de los riesgos detectados y las decisiones tomadas para su mitigación:

- **Gran volumen de datos:** La información contenida en la web *Smartbike Report Manager* presenta un gran volumen de ficheros (Sección 2.3.3) de distintos tipos de búsqueda y en muchos casos la información que continen está duplicada.

- **Acceso vía formularios:** Los datos de *Smartbike Report Manager* tienen que extraerse de un portal vía formularios protegidos por usuario y contraseña. Existe un riesgo bajo de cambio del portal durante el proyecto. Este riesgo aumentará con el tiempo una vez finalizado el proyecto.

- **Formato de datos no estructurado:** Los datos de *Smartbike Report Manager* están disponibles en formatos orientados a su consulta via web, usando un lector de *PDF* o una aplicación capaz de trabajar con ficheros *Excel* o *CSV*. Los datos descargados tienen que ser adaptados para su posterior consumo.

- **Inconsistencia de información:** Se ha detectado que hay inconsistencias en la información recuperada al comparar los resultados de la misma pregunta en diferentes formatos. En el momento actual no se puede estar seguro de que toda la información proporcionada es consistente entre sí.

- **Cuenta pensada para analista:** La cuenta disponible para acceder a los datos está pensada para un analista. Existe un riesgo bajo de que *Smartbike Report Manager* cancele este usuario y contraseña por considerar que se está haciendo un uso incorrecto de la cuenta, por utilizarla para realizar descargas sistemáticas de información.

- **Inestabilidad del sistema:** El portal *Smartbike Report Manager* presenta algunos problemas de estabilidad al acceder a la información. Por ejemplo, el 2017-01-18 estuvo parcialmente caido durante varias horas, sin que ninguna información, sobre los motivos del problema, estuviese disponible.

- ***JHipster* y *Apache Hadoop*:** Este riesgo se detecta una vez comenzado el proyecto. La idea inicial del proyecto es conectar el servicio web directamente con los datos almacenados en el ecosistema *Hadoop*. A la hora de trabajar con *JHipster* se detecta que no es posible ya que no tiene soporte para ello.

Como medidas para mitigar estos riesgos se han tomado las siguientes decisiones:

- **Gran volumen de datos:** Se decide que el volumen de datos a recopilar es excesivamente grande, por lo que para el desarrollo de este Trabajo Fin de Grado se decide acotar las descargas a la colección de datos "Uso de las estaciones", que siendo lanzado durante dos semanas es capaz de recopilar aproximadamente 94.000 ficheros históricos más 258 ficheros que se descargarían durante el periodo que la aplicación está funcionando.

- **Acceso vía formularios:** Para simular el comportamiento de un usuario humano se ha utilizado la herramienta *Selenium* que realiza la navegación por las distintas secciones de la web, siendo capaz de rellenar los formularios y de realizar la descarga de los resultados obtenidos. Esta solución no mitiga el riesgo en su totalidad, ya que en caso de existir cambios en el portal habrá que modificar el proceso.

- **Formato de datos no estructurado:** Se decide que los datos serán descargados en formato *Excel*, debido a que son los más completos y estructurados. Una vez que se dispone de los ficheros descargados son procesados con *Apache POI*, que permite la extracción de información que posteriormente será almacenada en un fichero *CSV*, lo que facilita la inserción de la información en la base de datos. Esta solución no mitiga el riesgo en su totalidad, ya que en caso de cambiar el formato de las hojas de cálculo habrá que modificar el proceso de extracción de información.

- **Inconsistencia de información:** Tras analizar la información recuperada se decide descartar, para la solución final, aquella que no presente relevancia o que pueda ser susceptible de inconsistencia.

- **Cuenta pensada para analista:** Tratandose de un riesgo difícilmente controlable desde este proyecto, una de las soluciones que se ha planteado es que las descargas se hagan de manera periódica y de forma lineal, excluyendo la posibilidad de tener varios procesos atacando a la vez el servicio. Con esto se busca que el comportamiento del proceso sea lo más similar a la actuación de un humano que realiza varias descargas a lo largo del día.

- **Inestabilidad del sistema:** Riesgo heredado del sistema anfitrión que no puede ser controlado desde está solución. Hay que destacar, que salvo la que se detecto antes de comenzar el proyecto, que en el periodo de desarrollo del proyecto no se ha detectado ninguna anomalía en el funcionamiento del portal *Smartbike Report Manager*.

– ***JHipster* y *Apache Hadoop*:** En un principio se intenta realizar la configuración de forma manual, convirtiéndose una tarea extremadamente compleja dado el poco conocimiento de la tecnología y la poca documentación existente. Se llega a la conclusión de que la mejor opción es incluir una base de datos intermedia, en este caso *MySQL*, que si es soportada por *JHipster*. Por lo tanto, finalmente se cuenta con una base de datos para escritura (*Hadoop*) y una base de datos para lectura (*MySQL*).

2.5. Conclusiones del análisis y requisitos a implementar

Una vez realizadas todas las tareas de análisis previas se definen los requisitos que finalmente se abordarán en este proyecto, descartando aquellos que se consideran inviables dado el tiempo y coste del proyecto. La elección de estos requisitos, tomados en la primera captura (Sección 2.2), son el producto de detectar cuales son las necesidad reales que son necesarias abordar a día de hoy con este proyecto, dejando un sistema listo para poder ser expandido en futuras iteraciones.

Requisitos funcionales:

– Recuperar y almacenar a largo plazo la colección de datos "Uso de Estaciones" en *Smartbike Report Manager* (`http://reportingportal-zar.clearchannel.com/`) para cada estación y con granularidad de una hora (si está disponible) en el formato estructurado (si está disponible) original.

– Ofrecer un proceso replicable para recuperar los datos históricos más relevantes disponibles en *Smartbike Report Manager*.

– Ofrecer un proceso replicable para recuperar cada día los datos nuevos disponibles en *Smartbike Report Manager*.

– Gestionar la recuperación de los datos mediante un sistema que permita ver el estado de la recuperación y tomar decisiones para recuperar de forma manual, automática y periódica los datos.

– Procesar todos los datos para ser expuestos mediante un modelo de datos relacional que permita realizar preguntas *SQL*.

– Disponer de un portal que permita navegar por los datos, recuperar subconjuntos de información y visualizar en formato tabular información sobre uso de *Bizi Zaragoza*.

Requisitos no funcionales:

- Exponer el modelo mediante un *API REST*.
- La información almacenada debe estar presente en su formato original de descarga, en este caso hoja de cálculo *Excel*.
- La información que ha sido tratada debe almacenarse en ficheros estructurados (*CSV*).
- El sistema dispone de un fichero de configuración para establecer parametros de conexión con las bases de datos.
- La solución sirve como plantilla para futuras implementaciones.
- El sistema es escalable y robusto.
- El sistema dispone de un fichero de configuración para establecer parametros de acceso a la web *Smartbike Report Manager*.
- La implementación del sistema se realiza utilizando la *Java* versión 8.
- Los sistemas de persistencia son diseñados utilizando *Apache Hadoop* versión 2.8 y *MySQL* versión 5.7.18.
- La aplicación web es generada con *JHipster* versión 4.6.
- EL sistema es capaz de ser desplegado y ejecutado sobre un sistema Linux.

Capítulo 3

Diseño del sistema

A la hora de realizar el diseño del sistema acorde con los requisitos (Sección 2.5) y objetivos (Sección 1.2) del proyecto. Se ha optado por un sistema modular capaz de funcionar de manera independiente, automatizada y periódicamente.

3.1. Arquitectura del sistema

En esta sección se aporta una visión global del sistema, mostrando su arquitectura (Figura 3.1) y un diagrama donde se representa el comportamiento de los diferentes elementos del sistema (Figura 3.2). Como se aprecia la arquitectura del sistema está compuesta por una capa de persistencia en la cual se cuenta con dos tipos distintos de almacenamito, una capa denominada servidor que contiene una visión resumida de los procesos que se han generado para la solución y por último una capa cliente, que contiene la interfaz web donde se puede gestionar la información disponible contenida en la capa de persistencia.

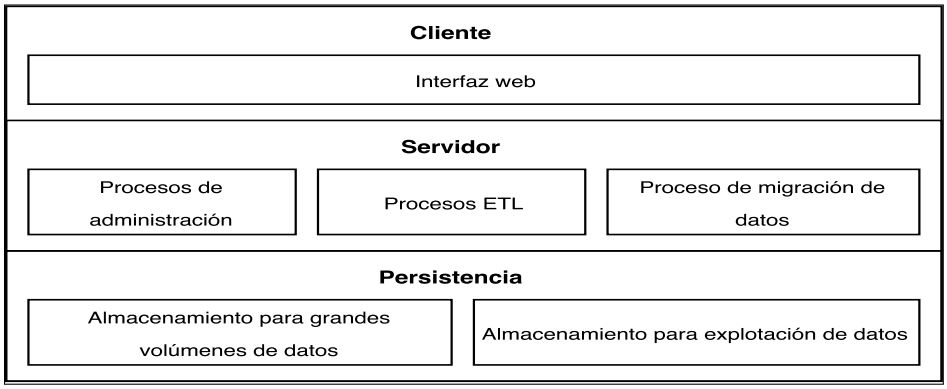

Figura 3.1: Arquitectura del sistema desarrollado.

Por otra parte, la visión global del sistema (Figura 3.2) refleja el flujo lineal que sigue la información desde que es descargada de la web *Smartbike Report Manager*,

hasta que es consumida por el usuario final. En este diagrama no se han incluido los procesos de administración ya que para describir el flujo de la información no son necesarios. Todos los elementos del sistema serán desglozados en las siguientes secciones. Para el modelado de procesos se ha decidido seguir el estándar de modelado *BPMN*, ya que ofrece una visión clara y limpia del ciclo de vida de cada proceso del sistema. Para más información acerca del modelado *BPMN* se puede consultar el Anexo B.

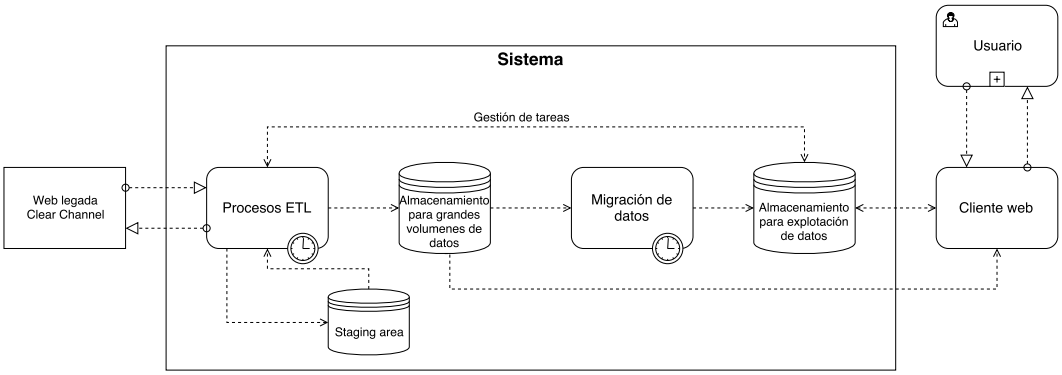

Figura 3.2: Visión global del diseño del sistema.

3.2. Persistencia

Para dar persistencia a la información obtenida por los proceso *ETL* se definen dos sistemas de almacenamiento atendiendo a las necesidades de la solución. Como se puede ver en la visión global de flujo de información (Figura 3.3) se cuenta con un sistema de almacenamiento para grandes volúmenes de datos, donde se almacenan los ficheros extraidos desde la web *Smartbike Report Manager* (Ficheros *XLS*) y los ficheros una vez tratados (Ficheros *CSV*). Por otro lado, se tiene un sistema de almacenamiento para la explotación de datos que es la encargada de disponer de toda la información que luego podrá ser consultada desde la aplicación web.

Como el sistema dispone de un gestor de tareas, se han incluido las entidades que contienen la información para realizar los procesos que hacen que los datos vayan pasando de un estado a otro. Todo esto puede ser consultado o gestionado por los usuarios, según el rol que tengan asignado en la aplicación. Este sistema de persistencia está pensado para facilitar su escalabilidad, siendo posible añadir nuevas fuentes de datos de manera simple, replicando los procesos ya existentes.

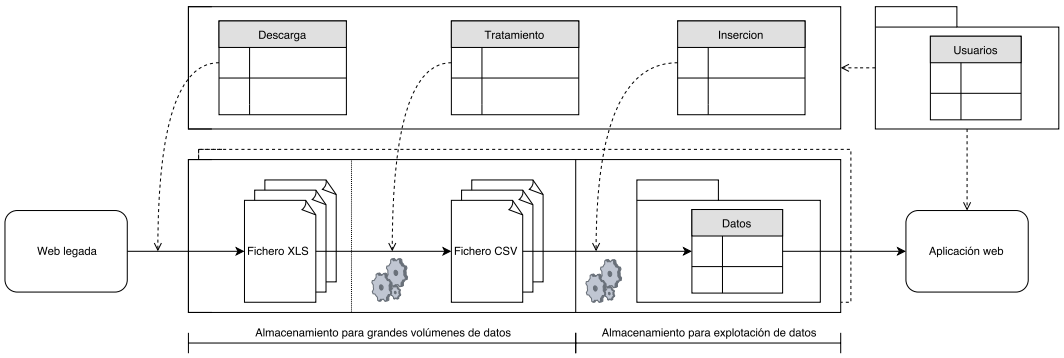

Figura 3.3: Visión global de flujo de información.

3.2.1. Almacenamiento de grandes volúmenes de datos

Uno de los principios a cumplir es dotar al sistema de persistencia para grandes volúmenes de datos, para ello se ha establecido que los datos sean almacenados en un sistema capaz de albergar esta cantidad de información, como podría ser el ecosistema *Apache Hadoop*, ya que *HDFS* cubre esa necesidad con solvencia. Además aporta la ventaja de disponer de los datos en *Apache Hive*, teniendo la posibilidad de realizar sobre los datos consultas *SQL*, y la posibilidad de exportar los datos a bases de datos con esquemas relacionales con la herramienta *Apache Sqoop*.

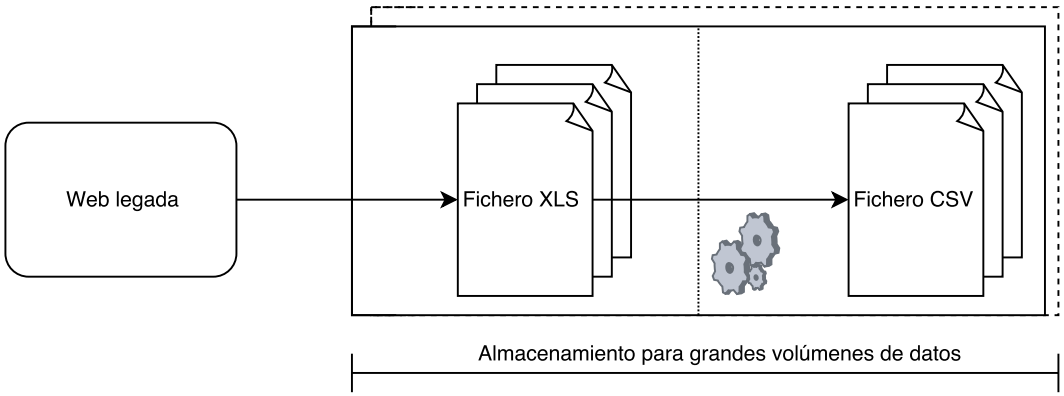

Figura 3.4: Persistencia de ficheros en Hadoop.

Como se puede observar en la Figura 3.4, en este almacen se conservan tanto los ficheros obtenidos por el proceso de extracción (Ficheros *XLS*), como los ficheros que han sido tratados (Ficheros *CSV*). Debido a que, una vez tratados los ficheros descargados, se dispone de los datos en un formato estructurado (*CSV*) capaz de ser insertado directamente, se cree conveniente que el diseño de la base de datos corresponda con la estructura de estos ficheros.

3.2.2. Almacenamiento para explotación de los datos

A parte de la persistencia de grandes volumenes de datos, es necesario generar una base de datos relacional, como por ejemplo *MySQL*, con la que se pueda gestionar la información que será expuesta en la aplicación web (Figura 3.5). Esta base de datos debe albergar la información migrada desde el sistema de persistencia de grandes volumenes de datos y un sistema de registros para el correcto funcionamiento de los procesos *ETL*.

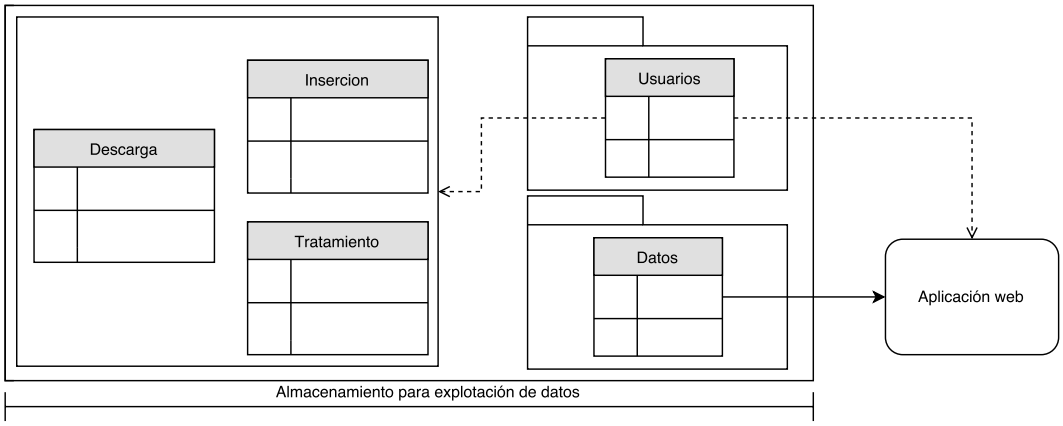

Figura 3.5: Persistencia de ficheros en MySQL.

En este sistema de almacenamiento se han generado diferentes entidades que tienen como finalidad realizar la gestión de usuarios que pueden acceder a la información por medio de la aplicación web (Figura 3.6), la gestión de tareas del sistema (Figura 3.7) y la gestión de la información que ha sido obtenida desde *Smartbike Report Manager*.

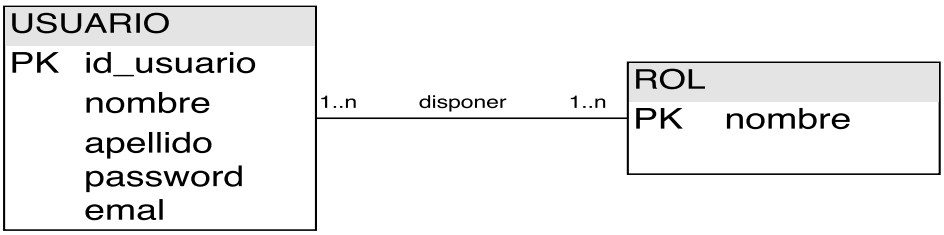

Figura 3.6: Vista lógica de la gestión de usuarios.

Entidades presentes en las Figuras 3.6 y 3.7 se describen de la siguiente forma:

- **Usuario:** Se trata del usuario que se registra en la aplicación web.
- **Rol:** Distinciones, que se asignan a los usuarios, para definir los permisos que tendrán en el sistema.
- **Tareas:** Define las acciones que serán consumidas por los procesos *ETL*.

Figura 3.7: Vista lógica de la gestión de tareas.

- **Descarga:** Especialización de Tareas que alberga la información referente a la descarga de ficheros del sistema web legado.
- **Tratamiento:** Especialización de Tareas que alberga la información referente al tratamiento de los ficheros *Excel*.
- **Inserción:** Especialización de Tareas que alberga la información referente a la inserción de los datos, contenidos en ficheros *CSV*, en *Hadoop*.

Para el caso de la información referente al servicio *Bizi Zaragoza* se realiza el diseño pensando en que cada nueva fuente de información será representada con una entidad nueva, por lo que para la inserción de estos nuevos datos solamente sería necesario hacer una réplica de las entidades existentes, con los atributos pertinentes, y los procesos que se encargan de realizar el que la información fluya, una vez descargada desde *Smartbike Report Manager* hasta su explotación en la aplicación web.

3.3. Procesos de administración de trabajos

Estos procesos son los encargados de gestionar el buen funcionamiento de los procesos de extracción, tratamiento y carga de datos. Aunque son procesos automáticos requieren supervision del usuario. En consecuencia cuando sean implementados, además de las funcionalidades que se describen aquí, será necesario implementar una consola web de control para su gestión.

3.3.1. Gestor de trabajos

La función de este proceso es lanzar periódicamente los procesos *ETL*, para que realicen sus tareas pendientes. Este proceso tendrá un comportamiento similar al administrador de tareas *Cron*, por lo que se puede configurar en que momento del día debe ejecutarse un proceso o cada cuanto tiempo debe ser lanzado.

3.3.2. Generador de trabajos

Proceso lanzado automáticamente cada 24 horas encargado de generar una entrada en la base de datos, indicando los datos que deben ser descargados en la próxima iteración del proceso de descargas (Figura 3.8).

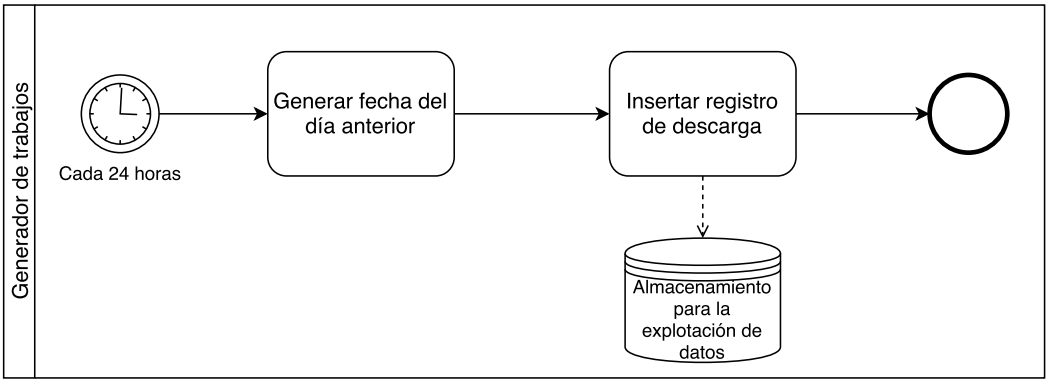

Figura 3.8: Proceso generador de trabajos.

3.3.3. Supervisor de trabajos

Durante la fase de implementación se estima que, para dar mayor robustez al sistema, se implemente un proceso capaz de revisar el registro de tareas, desplegado en la base de datos de gestión de información, en busca de posibles registros que se hayan quedado marcados como que están siendo procesados por un proceso que no haya terminado correctamente su ejecución. En caso de encontrar uno de estos casos, el registro será modificado para que vuelva a estar disponible para su captura por otro proceso.

3.4. Incorporación replicable de una fuente de datos

Una de las claves principales de este proyecto es la automatización de las tareas que permiten recolectar, tranformar y exponer la información, esto se conoce como *ETL*. Estos procesos han sido desarrollados para que a lo largo del día sean lanzados de forma automática y realicen las tareas pendientes que existan. A continuación, se muestran el flujo de trabajo de cada proceso modelado siguiendo el estándar de modelado *BPMN*.

3.4.1. Estrategia general

A la hora de diseñar los procesos, se ha tenido muy en cuenta la estrategia que se quiere abordar para conseguir que los procesos del sistema sean facilmente replicables.

Gracias a esto, la solución cuenta con la virtud de poder acoplar facilmente nuevos procesos, encargados de trabajar con nuevas colecciones de datos. Esto es debido a que el flujo de ejecución que siguen las tarea que a continuación se describen, son controlados por los procesos de administración (Sección 3.3), por lo que el nuevo proceso simplemente necesitaría ser añadido a la gestión de estos.

Poniendo como ejemplo la tarea de descarga de información, para realizar un nuevo proceso que fuese capaz de descargar nuevas colecciones de datos a partir de la web *Smartbike Report Manager*, solamente sería necesario replicar el proceso "Realizar descarga" (Figura 3.9) con su nueva configuración. Este nuevo proceso sería facilmente acoplable al flujo mostrado, ya que su gestión estaría acargo de los procesos de administración de trabajos (Sección 3.3) y no interfiere para nada con el las tareas realizadas por otros procesos de descarga.

3.4.2. Proceso ETL. Descarga de información

Proceso que a lo largo del día se ejecuta cada 30 minutos para comprobar que existe la posibilidad de descargar ficheros del sistema web legado, en caso de existir el proceso es capaz de conectarse a la web *Smartbike Report Manager* y realizar la descarga del fichero, que será almacenado en una carpeta del sistema de ficheros, que sirve de "staging area". Una vez realizada su tarea con éxito inserta nueva información en la base de datos referente al tratamiento del fichero descargado (Figura 3.9).

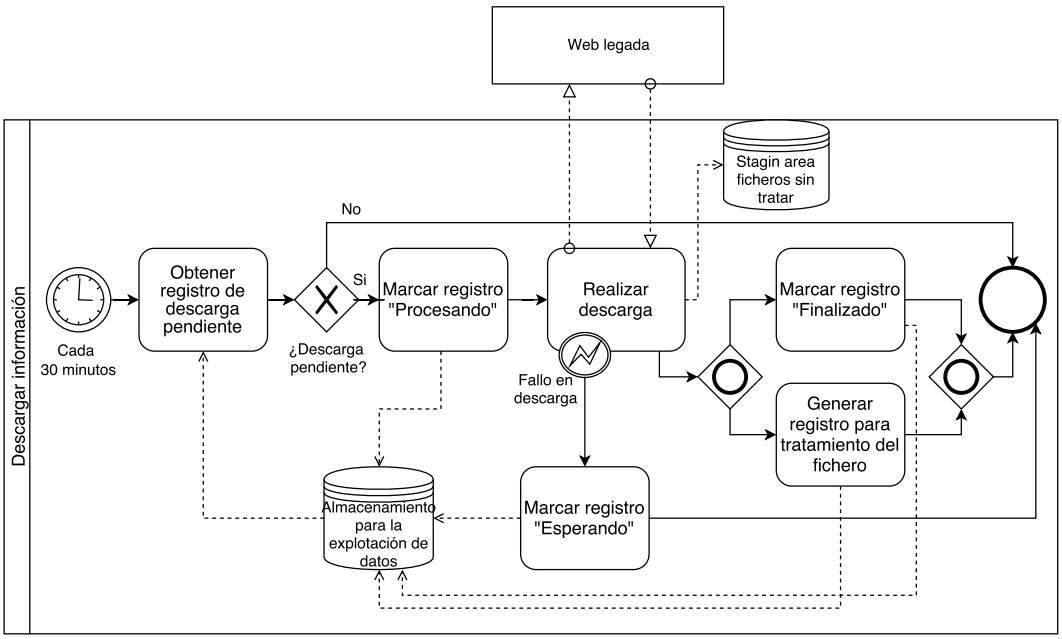

Figura 3.9: Proceso de descarga de ficheros.

3.4.3. Proceso ETL. Tratamiento de datos

Proceso que durante el día se ejecuta cada 30 minutos y se encarga de recuperar aquellos ficheros que se han descargado y deben ser tratados para su conversión en ficheros *CSV*. Una vez realizada su tarea de tratamiento con exito se inserta nueva información en la base de datos referente a que esos fichero ya pueden ser almacenados en la base de datos. El nuevo fichero generado debe ser almacenado en una carpeta del sistema de ficheros, que sirve de "staging area", a la espera del proceso que los inserte en la base de datos. (Figura 3.10).

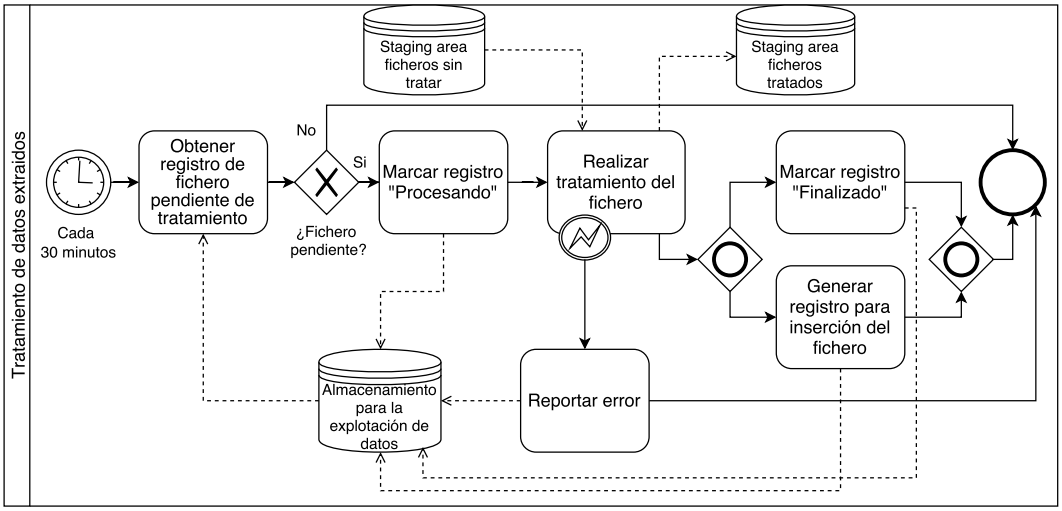

Figura 3.10: Proceso de tratamiento de ficheros descargados.

3.4.4. Proceso ETL. Almacenamiento masivo

Proceso que durante el día se ejecuta cada 30 minutos y es el encargado de recuperar los ficheros que han sido tratados y están listos para ser insertados en la base de datos. En caso de disponer de ficheros para insertar, se realiza la inserción de los datos en el sistema de persistencia de grandes volumenes de datos y marca como finalizado el ciclo de vida de las tareas referentes a este fichero (Figura 3.11).

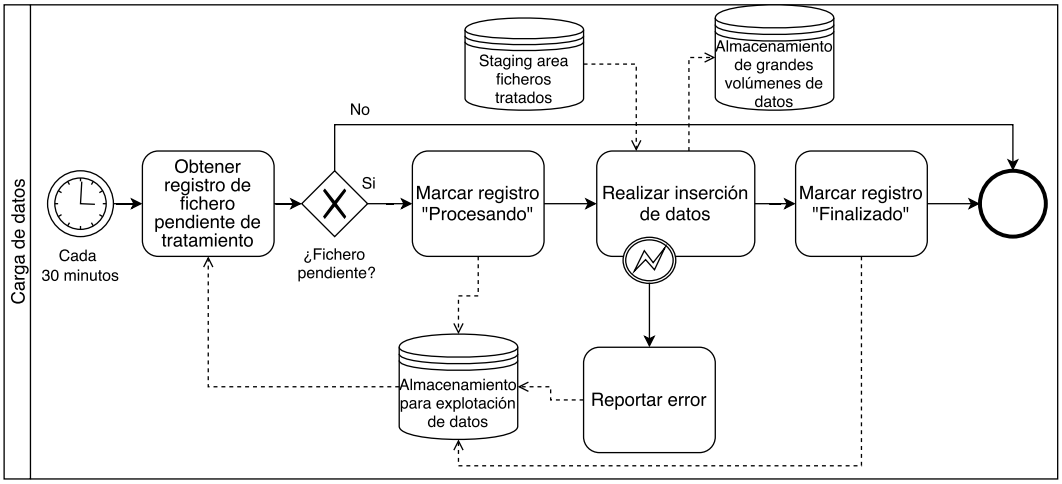

Figura 3.11: Inserción de los datos tratados en Hadoop.

3.4.5. Proceso ETL. Preparación para explotación

Proceso que se ejecuta cada hora recolectando toda la información disponible en el sistema de persistencia de grandes volumenes de datos y realiza un volcado en el sistema de gestión de información (Figura 3.12), para que esta pueda ser consultada desde la aplicación web. Este proceso se ha incluido para mitigar el riesgo detectado (Sección 2.4) de la dificultad de exponer los datos almacenados en *Apache Hadoop* en *JHipster*.

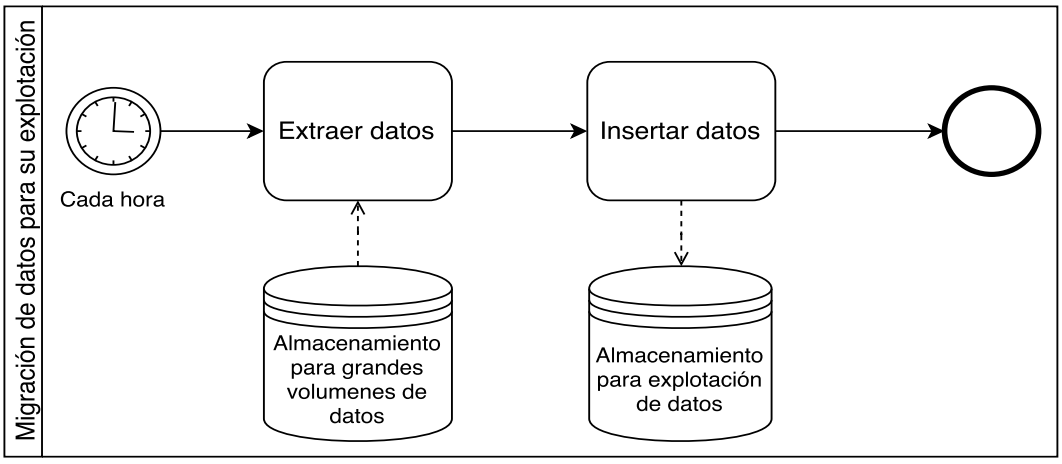

Figura 3.12: Migración de los datos desde Hadoop hacia MySQL.

3.4.6. Explotación de los datos

Para la explotación de los datos se generará una aplicación web que permita al usuario realizar consultas de los datos referentes al servicio *Bizi Zaragoza* que se hayan recolectado hasta el momento. Se diferenciará entre un usuario normal, que solo pueda acceder a datos de caracter informativo, y otro, administrador del sistema, capaz de

visualizar, además, la información de los registros de la gestión de tareas pudiendo modificar el estado de estas. Desde esta aplicación web el usuario contará con una interfaz donde podrá consultar y gestionar los procesos de administración de trabajos.

Capítulo 4

Implementación del sistema

Una vez finalizada la fase de diseño (Capítulo 3) se procede a realizar la implementación de los componentes que van a ser utilizados en la solución final del proyecto. Durante la fase de implementación se han realizado cambios considerando que aportan mayor eficiencia y eficacia. Como se ha nombrado con anterioridad, la implementación del sistema se realiza enfocando la solución al caso más relevante, sirviendo esta solución como base para futuros proyectos.

La implementación del sistema se confecciona en varias iteraciones, una por cada proceso descrito en la Figura 4.1. Durante cada iteración se realizan la pruebas pertinentes para asegurar que cada proceso funciona correctamente, asegurando así la calidad de la solución final.

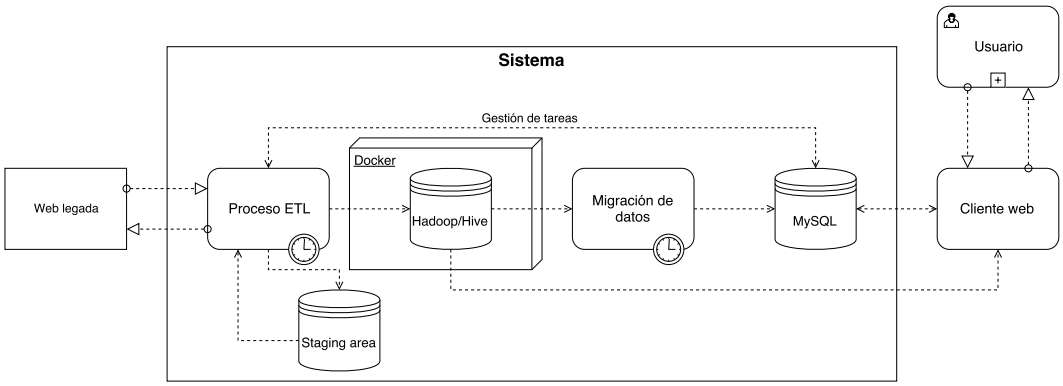

Figura 4.1: Visión global del sistema.

En la Figura 4.1 se dispone de una visión global, y a grandes rasgos, de la estrutura final del proyecto. Como punto de partida se dispone de un proceso *ETL* que, lanzado periódicamente, se encarga de recuperar la información de un sistema web externo y almacenarlo en *Apache Hadoop*, que se encuentra contenido dentro de un desplegado en un *Docker*, este proceso se gestiona a partir de gestor de tareas implementado sobre *MySQL*. A continuación un proceso de migración de datos, también lanzado

periódicamente, que recopila la información en *Hadoop* y la introduce en *MySQL*. La información contenida en *MySQL* es expuesta en un servicio web en el que los usuarios podrán consultarla. También existe la posibilidad de que, desde el cliente web, el usuario pueda obtener los ficheros almacenados en *Hadoop*.

4.1. Persistencia

La implementación de la persistencia del sistema se ha llevado a cabo de acuerdo con el diseño implementado en la fase anterior del proyecto (Sección 3.2). Este diseño a la hora de ser implementado ha sufrido modificaciones, entendiendo que estas eran necesarias para mejorar la solución final.

4.1.1. Almacenamiento para grandes volúmenes de datos. Apache Hadoop

A la hora de trabajar con el ecosistema *Apache Hadoop* se ha optado por trabajar sobre la infraestructura de almacenamiento de datos *Apache Hive*, ya que ofrece la posibilidad de trabajar con los datos como si se tratase de un esquema relacional.

Para trabajar con *Apache Hadoop* se ha realizado el despliegue de un contenedor en *Docker*, en el que se dispone de todos los servicios necesarios para llevar a cabo las tareas de este proyecto, en cuanto a persistencia de datos. Para el desarrollo de este Trabajo Fin de Grado se ha estimado oportuno que solamente se va trabajar con un único nodo de *Hadoop*, siendo posible expandirse el número de estos en futuros proyectos.

Ya que la base de datos se encuentra dentro del contenedor *Docker* ha sido necesario exponer los puertos necesarios hacia exterior para poder comunicar el proceso de carga de datos con la base de datos. También ha sido necesario instalar y configurar la herramienta *Apache Sqoop* para poder realizar de forma automática la migración de los datos contenidos hacia la base de datos relacional (*MySQL*).

Por las decisiones tomadas, sobre los datos que se extraeran en la web *Smartbike Report Manager*, en el análisis del sistema (Capítulo 2), se ha optado por generar una base de datos que contiene una única tabla que recoge toda la información de la que se dispone en los ficheros tratados. Con esto se pretende conseguir que, en caso de aumentar los tipos de datos descargados, sea fácilmente escalable ya que solo sería necesario crear una nueva tabla para la nueva información.

La decisión de almacenar los datos de esta manera es fruto de que *Hadoop* solo es utilizada para el almacenamiento y la aplicación que el cliente recibe no tendrá la posibilidad de realizar consultas directamente contra ella, sino que trabajará en base a los datos que se han migrado hacia *MySQL*. Por lo tanto se sigue contando con un almacenaje capaz de persistir en el tiempo y a su vez crecer sin depender de lo que el usuario final pueda consultar, para eso está *MySQL*. Aunque en caso de requerirlo, el usuario podrá descargar los ficheros originales, que contienen la información, que se encuentran almacenados en el sistema *HDFS* de *Hadoop*.

4.1.2. Almacenamiento para la explotación de los datos. MySQL

Como se explica en la fase de diseño (Sección 3.2), la base de datos de gestión de información, en este caso *MySQL*, es la base de datos intermedia que alberga los datos de la aplicación web y un sistema de registros para las tareas *ETL* que se realizan de manera automática.

En una primera aproximación se generaron dos bases de datos independientes; una para la aplicación web que contenia toda la información que iba a ser visible por un usuario final y los datos de gestión (usuarios, roles, *logs* del sistema, etc.), y otra para gestionar las tareas de los procesos *ETL*. En una siguiente iteración, al hacer una aproximación de trabajo con la aplicación web, se decide que no tiene sentido tener dos bases de datos separadas, ya que para el administrador de la aplicación es interesante poder gestionar las tareas desde la misma interfaz web. Por esto, se decide modificar lo construido anteriormente y generar una única base de datos que contenga toda la información que se maneja en el proyecto.

Durante el desarrollo de esta fase se plantearon dificultades a la hora de insertar nuevas entidades a la base de datos, ya que era necesario tener un mapeo estricto con los procesos generados por *JHipster*, por lo que las entidades propias del proyecto han sido generadas a partir del generador *JHipster Entity* [14], que a partir de un fichero, donde se especifica la estructura de la entidad, es capaz de generar la entidad en la base de datos y los procesos del sistema necesarios para realizar las consultas sobre ellos, al igual que genera también la vista que se añade a la aplicación web final.

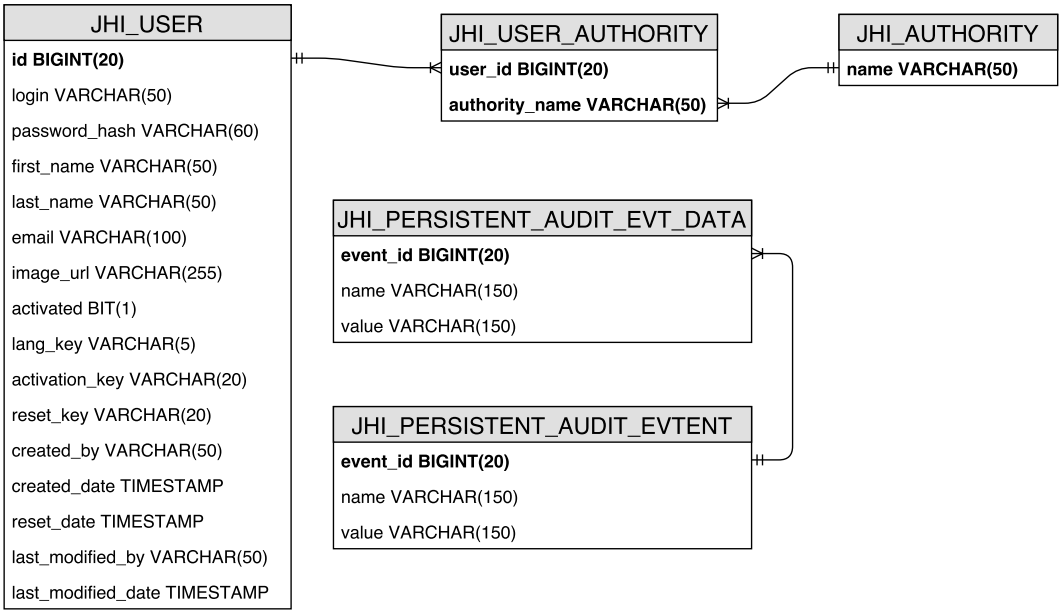

Figura 4.2: Modelo relacional de la base de datos en MySQL. Entidades desplegadas por *JHipster*.

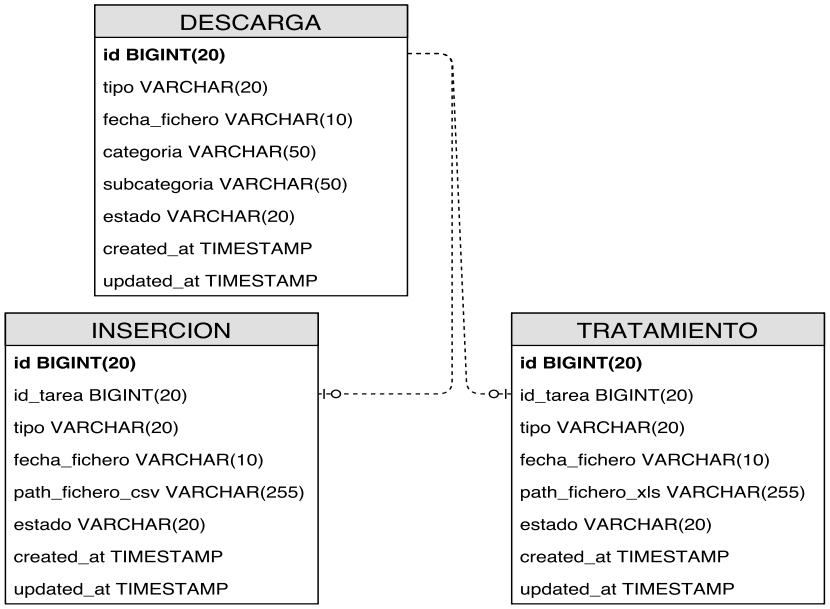

Figura 4.3: Modelo relacional de la base de datos en MySQL. Entidades para gestión de tareas.

En las Figuras 4.2 y 4.3 se puede observar el modelo *Relacional* que representa como ha sido implementada la base de datos finalmente. Se pueden observar varios cambios si se compara con el modelo presentado en la fase de diseño (Capítulo 3). Estos cambios son promovidos por la propia inercia de los procesos que trabajan con la información, ya que con esta nueva estructura se consigue mayor eficacia y rendimiento.

Las entidades con el prefijo *"JHI"* (Figura 4.2), son tablas autogeneradas por

el generador de aplicaciones web *JHipster*. Estas tres entidades corresponden a las entidades *"Usuario", "Rol"* y a la relacion *"disponer"* que se presentan la visión lógica presentada en en la fase de diseño (Figura 3.6). Por otra parte se dispone de dos tablas que son utilizadas para la gestión de auditorias, que recogen todos los eventos que se realizan en el sistema.

Las tareas, que en la visión lógica en la fase de diseño (Figura 3.7) se representaban como una entidad que disponia de tres especializaciones, se ha sustituido por cada una de las tareas (*"DESCARGA", "TRATAMIENTO", "INSERCION"*) por separado, las dos últimas reciben como *Foreign Key* la clave primaria de la entidad *"DESCARGA"*. Esta relación realmente se ha generado a partir de implementación de código ya que se ha preferido que la información que comparten se propague por medio de los procesos del sistema. Esta decisión también viene apoyada por el hecho de que no se van a realizar borrados de tareas, por lo que no hace falta incluir dependencias para un borrado en cascada. También facilita la tarea de acoplar nuevos datos, ya que no es necesario realizar nuevas relaciones a nivel de base de datos,

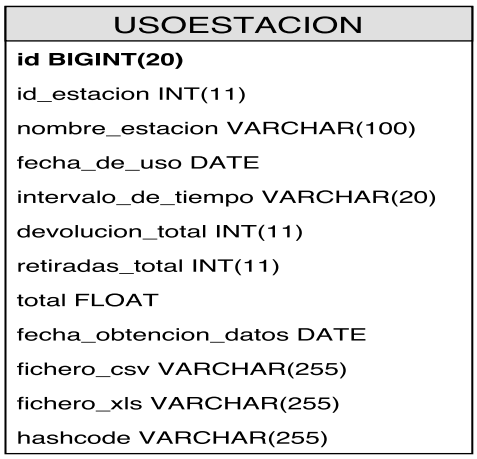

Figura 4.4: Modelo relacional de la base de datos en MySQL. Entidad USOESTACION.

Para el caso particular de "Uso de Estaciones", se ha generado la entidad *"USOESTACION"* que contiene toda la información que ha sido migrada desde *Hadoop* (Figura 4.4). La entidad *"USOESTACION"* puede ser utilizada como referencia a la hora de realizar nuevas entidades que contengan información del servicio *Bizi Zaragoza*.

Por último, hay que mencionar que todas las interacciones de las entidad "JHI" con el resto de entidades se realizan a partir de codificación en la fase de explotación, ya

que vienen configuradas por defecto al desplegar *JHipster*.

4.2. Procesos de administración y su consola web

Una vez finalizado y verificado el diseño de los procesos (Capítulo 3) se lleva a cabo su implementación en varias etapas, obteniendo una solución capaz de ser gestionada de forma automática. En el caso de los procesos de administración hay que destacar que estos procesos pueden ser gestionados por los usuarios a partir de la aplicación web. Por esto, al desplegar la aplicación web con *JHipster* se ha generado una interfaz en la cual se puede consultar y gestionar el control de las tareas llevadas a cabo por los procesos de administración (Sección 4.4).

4.2.1. Gestor de trabajos

Ya que los procesos *ETL* se lanzan de manera automática y cada cierto tiempo, se ha implementado este proceso *Gestor* que se encarga de lanzar los procesos en segundo plano mientras la aplicación está corriendo normalmente.

Para la automatización del lanzamiento de tareas se ha utilizado la libreria *Scheduled* disponible para *Spring Framework*. Está librería aporta la posibilidad de incluir anotaciones en los procesos para indicar en que momento del día se desea ejecutar el contenido del proceso [15]. Su funcionamiento es similar al administrador de tareas *Cron* en los sistemas *Unix*.

4.2.2. Generador de trabajos

Aunque este proceso no entra dentro de la categoría de los procesos *ETL*, también es gestionado por el proceso *Gestor*, ya que debe ser lanzado cada 24 horas para insertar en la base de datos que existe un nuevo fichero que debe ser descargado. Esto es posible ya que a partir del análisis del sistema (Sección 2.1) se sabe que los datos más actuales son los del día anterior al actual.

Como se puede ver en el diagrama de secuencias (Figura 4.5), el proceso *Gestor* indica el comienzo del proceso, y este introduce en la base de datos en *MySQL* la nueva tupla con el contenido necesario para que el proceso de descargas pueda realizar su trabajo. Aunque no se muestre en el flujo del diagrama, ya que se trata de un caso muy excepcional, el proceso es capaz de revisar si los datos que ha generado ya existen en la base de datos, ya que en caso de existir no los volverá a insertar.

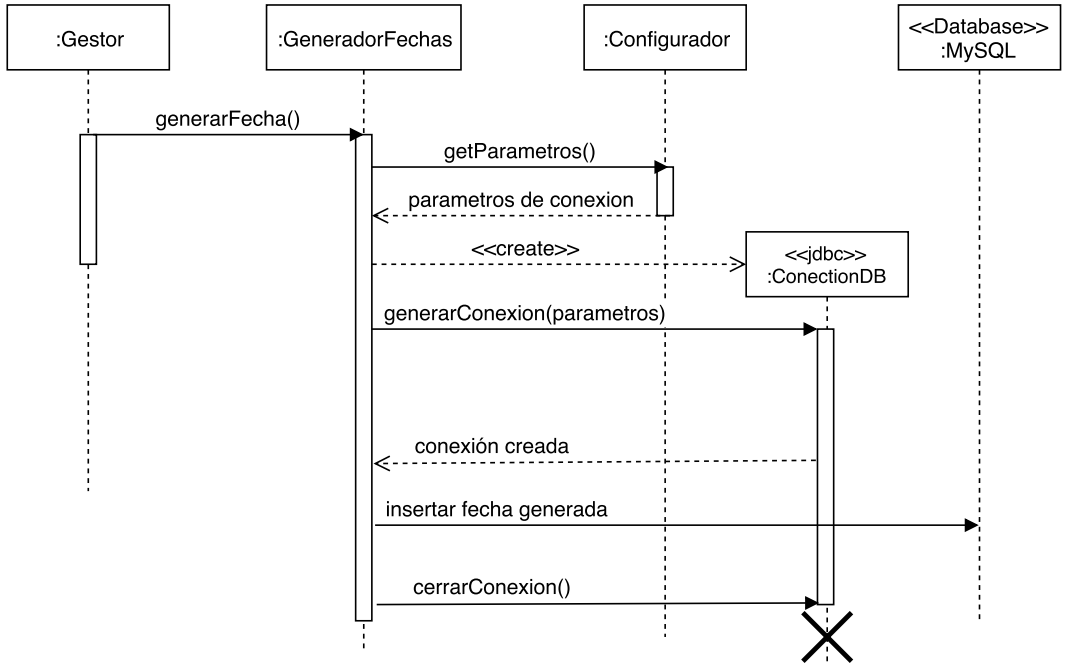

Figura 4.5: Diagrama de secuencia del proceso Generador de trabajos.

Como caso especial de este proceso, existe un proceso similar que genera todas las fechas desde hace dos hasta el día actual, consiguiendo así programar la descarga del historial disponible en el servicio web legado. Este caso especial solo es necesario lanzarlo al desplegar el sistema.

4.2.3. Supervisor de trabajos

Este proceso, aunque en la fase de diseño no se observó la posibilidad de contar con él, durante de la implementación de los procesos *ETL* se decide que es necesario implementar un proceso de este estilo para controlar posibles fallos en el sistema, dando mayor robustez a la solución.

El proceso de limpieza tiene como función única comprobar la gestión de tareas en busca de posibles registros que hayan quedado bloqueados en el estado *"PROCESING"*, bien porque el proceso que lo estuviese procesando fallase y no llegase a modificar su estado o por una caida del servicio en algún momento.

Por lo tanto este proceso se lanza de forma automática, durante una franja temporal donde el resto de procesos no esten en ejecución, y realiza un barrido por las tareas en busca de estas anomalías. En caso de encontrar alguna, modifica el estado del recurso a *"WAITING"*, así la próxima vez que se ejecuten los procesos sean capaces de gestionar de nuevo ese recurso.

Para evitar el problema de contar con varios procesos capturando un mismo recurso, cada proceso solo recupera registros que tengan su estado como *"WAITING"*, al obtener un registro de la base de datos actualizará su estado a *"PROCESING"* y una vez finalizado, si terminan correctamente modifican el estado a *"FINISHED"*, en caso contrario el estado volverá a ser *"WAITING"*.

4.3. Proceso de referencia: Uso de Estaciones

Una vez finalizado y verificado el diseño de los procesos (Capítulo 3) se lleva a cabo su implementación en varias etapas, obteniendo una solución capaz de ser gestionada de forma automática. La implementación de los procesos se ha realizado en *Spring Framework*, por lo que se ha utilizado el lenguaje de programación *Java*.

A continuación se muestra detalladamente el comportamiento de los procesos presentes en la visión global del sistema (Figura 4.1). Estos procesos reciben los parámetros necesarios para realizar configuraciones y conexiones a partir de un gestor de configuraciones que obtiene los datos a partir del fichero *myConfig.json* (Anexo C.2).

4.3.1. La fuente Uso de Estaciones

La colección de datos "Uso de estaciones" ha sido escogida como colección de ejemplo para desarrollar esta solución debido a que es la que aporta mayor cantidad de información relevante dispone. A partir del análisis de datos realizados (Anexo A.3) se corrobora que no contiene datos replicados o malformados. Como se puede observar en el Anexo C, el fichero, en formato hoja de *Excel*, obtenido desde *Smartbike Report Manager* presenta una estructura homogénea que hace que sea perfecta para realizar un proceso "plantilla" que posteriormente pueda ser replicado para siguientes colecciones de datos. Además el formulario de descarga de la web contiene la mayoría de los campos de búsqueda que se utilizan en el resto de formularios, por lo que además de "plantilla", este proceso sirve como guía para entender como se debe realizar la navegación por la web *Smartbike Reporting Manager*.

4.3.2. Proceso ETL. Descarga de información

El proceso de extracción de los datos del sistema web legado solo se ejecuta en caso de existir registros en la base de datos que le indiquen que debe realizar una nueva descarga.

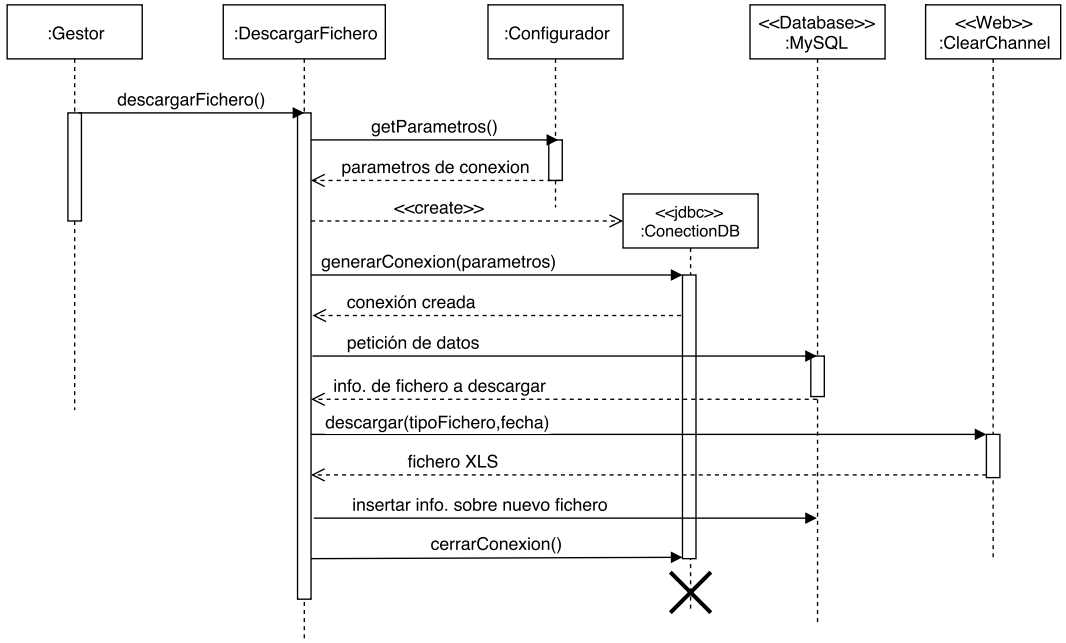

Figura 4.6: Diagrama de secuencia del proceso Descarga de información.

En el diagrama de secuencia (Figura 4.6) se observa el comportamiento normal del proceso, no se ha incluido el flujo del proceso si falla ya que lo único que realizaría es marcar el estado del recurso como *"WAITING"* y finalizar su ejecución.

Para poder efectuar la descarga de la información es necesario conectarse a la web *Smartbike Report Manager* (`http://reportingportal-zar.clearchannel.com/`), que necesita credenciales de acceso y navegar a través de diversas pestañas hasta llegar al formulario para solicitar los datos. Por esta razon ha sido necesario utilizar las herramientas *Selenium* y *ChromeDriver*. Con ayuda de la herramienta *Selenium* se define el patrón a seguir para realizar la navegación a través de la interfaz web. Una vez definido se procede a acoplar el resultado junto a las opciones que da *ChromeDriver*, obteniendo así todo el proceso automatizado de navegación y descarga de la información. Debido a que para poder realizar está tarea es necesario que el navegador esté visible y tenga un tamaño donde quepan todos los elementos de la web, se ha optado por configurar *ChromeDriver* para que la posición de la ventana este desplazada hacia una zona no visible del escritorio. También ha sido necesario modificar atributos de los campos de búsqueda a partir de inyección de código en *JavaScript*.

Para el caso en particular de la extracción de los datos referentes a "Uso de Estaciones" se muestra un pequeño ejemplo de las instrucciones necesarias para navegar y completar el formulario de búsqueda. En este ejemplo de navegación por

```
// Modificar atributo del elemento para que permita escritura
((JavascriptExecutor) driver)
.executeScript("document.
    getElementById('ucReportParameters1_EndDate').
    removeAttribute('readonly',0);");

WebElement toDateBox = driver.findElement(By.
    id("ucReportParameters1_EndDate"));
toDateBox.clear();
toDateBox.sendKeys(fecha); // Insertar dato en
                           el campo de busqueda

// Hacer click sobre la casilla de eleccion de dia
driver.findElement(By.xpath("//*
    @id=\"ucReportParameters1_
        ReportParametersColumn2\"]/
            div[2]/span/span/span")).click();
```

Listado 1: Ejemplo de navegación por los elementos de la web.

los elementos de la web (Listado 1) se puede observar que se ha necesitado realizar modificaciones de atributos en campos de búsqueda, ya que no permitían la escritura y hacian que la tarea se complicará demasiado. Se ve como para la selección de algunos elementos de la web se cuenta con un identificador, por lo que es simple la tarea de acceder a ellos, pero para otros casos es necesario recurrir al *"xpath"* (posición relativa del elemento dentro de la interfaz a partir del resto de elementos).

Viendo la dificultad real que entraña realizar esta tarea, se corrobora la decisión tomada al realizar el análisis de riesgos (Sección 2.4) de que se ve que la mejor opción es realizar un proceso de descarga para el tipo de fichero que contiene los datos más relevantes del servicio *Bizi Zaragoza*, "Uso de Estaciones", y que este proceso sirva como base para futuros proyectos que necesiten replicar este proceso con nuevos ficheros.

4.3.3. Proceso ETL. Tratamiento de datos

El proceso de tratamiento de ficheros se encarga de obtener los ficheros que han sido descargados del servicio web legado, que se encuentran almacenados como hojas de cálculo de *Excel*, para procesar la información que contienen y generar un fichero *CSV* con los datos recopilados.

Como se puede observar en el diagrama de secuencia del proceso (Figura 4.7) donde

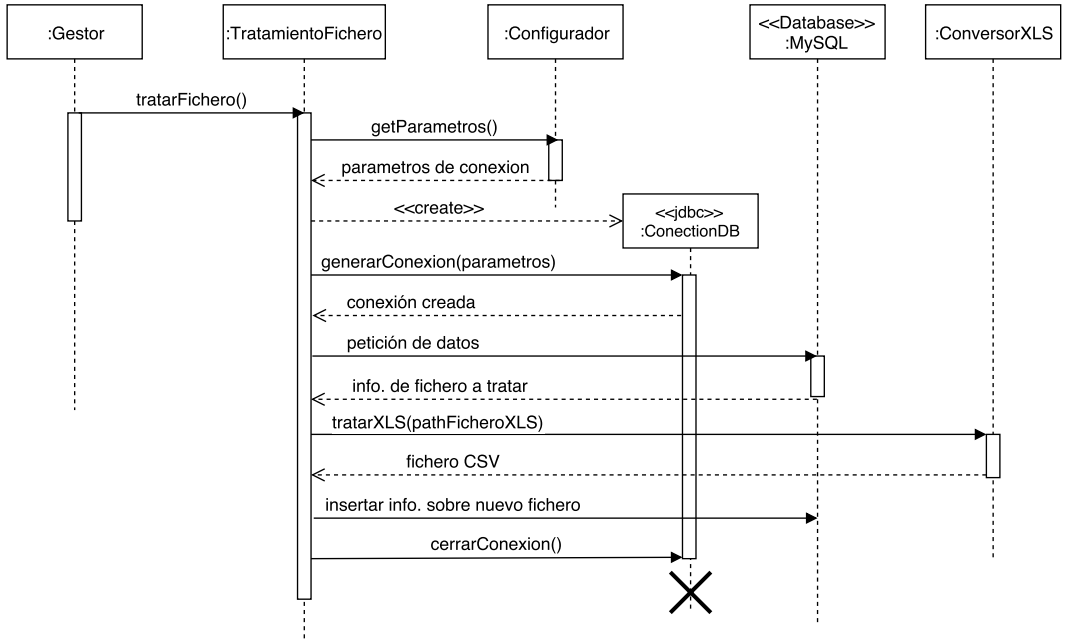

Figura 4.7: Diagrama de secuencia del proceso Tratamiento de ficheros.

se ha reflejado el flujo de las tareas principales, se puede observar que el patrón seguido es similar al del proceso de descarga de ficheros (Figura 4.6). En este caso también se puede observar que solo se ha integrado el comportamiento normal del proceso, sin incluir los casos de fallo, que en este caso podrían ser de los siguientes tipos:

- El fichero buscado no existe en el directorio especificado: En este caso se procede a marcar el estado del recurso como *"ERROR"* y se indica que es necesario descargar el fichero y finalizar su ejecución.

- Problema en el tratamiento del fichero: En este caso se marca el estado del recurso como *"WAITING"* y finalizar su ejecución.

Para proceder con la extracción de la información de las hojas de cálculo ha sido necesario utilizar la librería expuesta por *Apache POI*, que permite realizar captura de datos a partir de un *Excel*, sin necesidad de disponer de *Microsoft Office*. Dado el formato de estos ficheros (Anexo C.1) se ha generado un sistema de captura de contenido a partir del formato, por lo que se ha realizado exclusivamente para este tipo de información.

Para el caso particular de la colección de datos "Uso de Estaciones", se puede observar en el ejemplo de extracción de información de las hojas de cálculo (Listado 2), que es necesario hacer referencia a celdas exactas para recuperar según que información, ya que por la estructura del fichero (Anexo C) hay datos estáticos y otros que pueden

```
// Obtiene la fecha que especifica de cuando
// es la informacion contenida en el fichero.
private static String extraerFechaDeUso(HSSFSheet sheet) {
    CellReference cellReference = new CellReference("C9");
    HSSFRow hssfrow = sheet.getRow(cellReference.getRow());
    HSSFCell hssfcell = hssfrow.getCell(cellReference.getCol());
    String fechaDeUso = hssfcell.toString();
    String[] split = fechaDeUso.split(" ");
    fechaDeUso = split[split.length - 1];
    return fechaDeUso;
}
```

Listado 2: Ejemplo de extracción de información.

ocupar diferentes celdas según el volumen de información que contengan. Esto conlleva a que para otras colecciones de datos deban ser analizadas para generar este proceso acorde con la forma en que se encuentren localizados los datos, en las hojas de cálculo.

Una vez obtenida la información de la hoja de cálculo se genera un fichero *CSV* (Anexo C.1) donde se inserta la cabecera de los datos que va a contener y se añade linea por linea los datos separados por coma. A parte de los datos extraidos del fichero original, se inserta un dato adicional que se trata de una ristra de caracteres aleatorios (*"hashcode"*) generada a partir de la información de la linea. Este "hashcode" será utilizado a la hora de migrar los datos desde *Hadoop* hacia *MySQL*, para controlar que no se migren datos duplicados.

4.3.4. Proceso ETL. Almacenamiento masivo

Para la carga de datos en Apache Hadoop, se ha optado por realizar inserciones en Apache Hive aprovechando la posibilidad de realizar inserciones con el formato *SQL*. Para ello se obtienen los registros con la información de los nuevos ficheros listos para se cargados en la base de datos y se utiliza la opción de la que dispone *Hive* para hacer cargas a partir de ficheros *CSV*. Este proceso, a diferencia de los anteriores, es el único que necesita ser conectado a las dos bases de datos (*MySQL* y *Hadoop*).

En el diagrama de secuencia del proceso (Figura 4.8) se puede observar la generación de las dos conexiones necesarias, una para recolectar los registros de tareas pendientes (*ConnectionMySQL*) y otra para realizar la inserción de los datos (*ConnectionHive*).

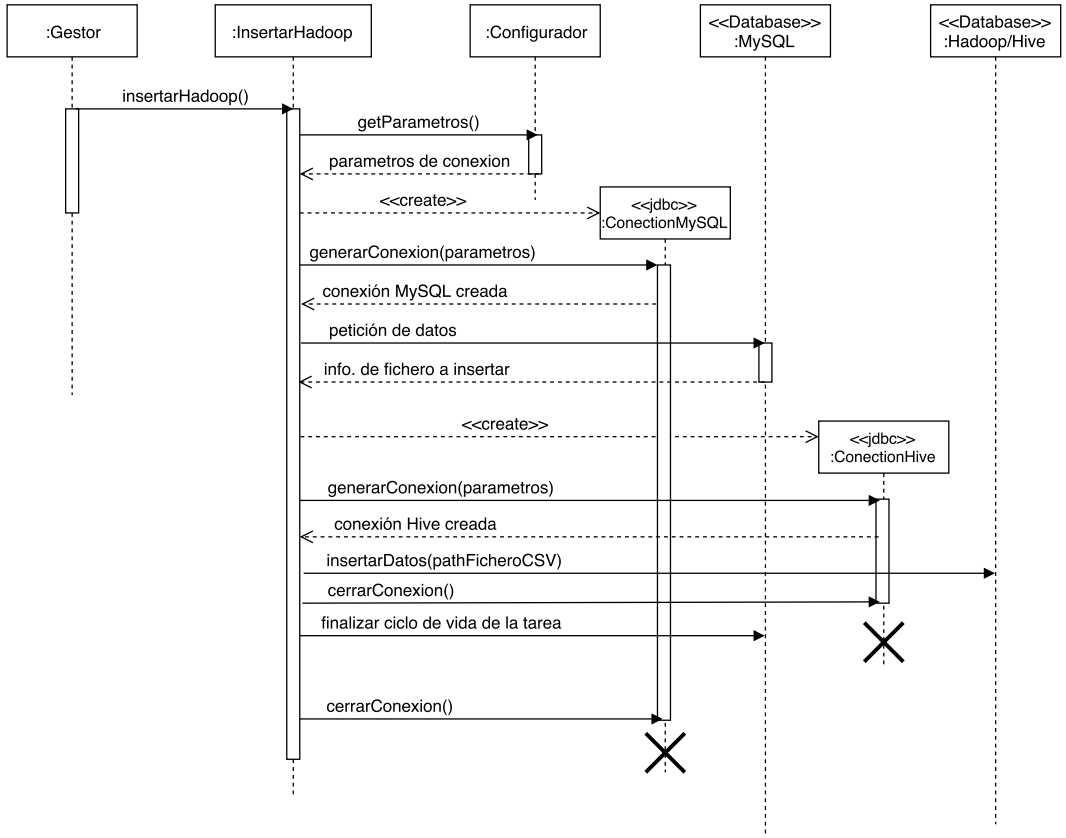

Figura 4.8: Diagrama de secuencia del proceso Carga de datos.

En este caso también se puede observar que solo se ha integrado el comportamiento normal del proceso, sin incluir los casos de fallo, que en este caso podrían ser de los siguientes tipos:

- El fichero buscado no existe en el directorio especificado: En este caso se procede a marcar el estado del recurso como *"ERROR"* y se indica que es necesario realizar el tratamiento del fichero *XLS* correspondiente y finalizar su ejecución.

- Problema en la carga de datos del fichero: En este caso se marca el estado del recurso como *"WAITING"* y finalizar su ejecución.

Para replicar este proceso simplemente es necesario indicar la fuente de datos que se quiere introducir en la base datos.

4.3.5. Proceso ETL. Preparación para explotación

Para la migración de datos desde *Hadoop* hacia *MySQL* se ha utilizado la herramienta *Apache Sqoop*. Esta herramienta es lanzada como un *script* desde consola ya que a partir de una orden el dispara todos los procesos necesarios para la migración. Como se puede observar en el ejemplo del comando *Sqoop* (*Listado* 3) para realizar la

exportación de los datos hacia *MySQL* solamente es necesario indicar cual es la base de datos destino, la tabla en la que se quiere insertar, las credenciales para la conexión y opcionalmente el nombre de las columnas que se desean exportar [16].

```
sqoop export --connect jdbc:mysql://<<host>>:<<port>>/<<dbName>>
   --table <<tableMySQLName>>
   --export-dir <<hdfsDirectoryPath>>
   --username <<username>>
   --password <<password>>
   --direct
   [--columns <<column1, column2, ...>>]
```

Listado 3: Ejemplo comando *sqoop*.

Para poder realizar estre proceso ha sido necesario instalar y configurar la herramienta *Apache Sqoop* dentro del contenedor *Docker* donde se encuentra *Hadoop*. En un principio al realizar las primeras migraciones se detecta que existen datos que se insertan duplicados, ya que *Sqoop* no controla si los datos que mueve existen en la tabla de destino, esto ha sido controlado añadiendo en el proceso de tratamiento de ficheros un campo más a los datos. Se trata de una ristra de caracteres aleatorios (*"hashcode"*), generada a partir de la información de la linea almacenada, y en la base de datos *MySQL* se ha indicado que este campo debe tener el atributo *"UNIQUE"*, evitando así que se vuelvan a insertar tuplas existentes.

Durante el desarrollo de esta tarea, se identifica el problema de que el servicio *MySQL*, por defecto, no acepta conexión que llegan desde el exterior (en este caso desde el contenedor *Docker*) por lo que ha sido necesario modificar la configuración del servicio de *MySQL* para que las acepte. Esto se ha realizado por medio de la modificación del fichero *mysqld.cnf*, que contiene la configuración del servicio, y se ha añadido la opción de aceptar conexiones externas. A parte, desde la consola de *MySQL*, se le han dado todos los privilegios al usuario de *MySQL* que se está utilizando. [17].

```
GRANT ALL PRIVILEGES ON *.* TO <<user>>@'%' IDENTIFIED BY '<<password>>';
```

4.4. Interfaz de explotación de datos

Para la explotación/exposición de la información se ha generado una aplicación web a partir del generador de aplicaciones *JHipster*, que permite a a través de *Yeoman*

generar una aplicación *Spring Boot* (componente de *Spring Framework* capaz de generar por completo una aplicación a partir de una configuración inicial) con un *front-end* en *AngularJS*. La capa de *back-end* está implementada con *Java* y *Spring Boot*, mientras que el *front-end* es *responsive* y está implementado con *AngularJS* y *Bootstrap*.

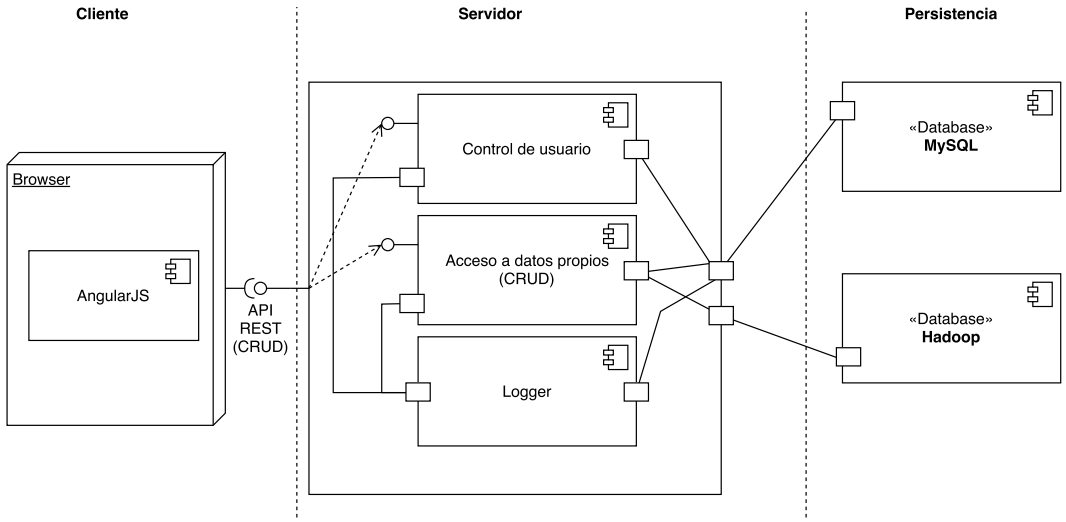

Figura 4.9: Diagrama de componentes simplificado del sistema.

La utilización de *JHipster* ha permitido extraer la complejidad de desarrollar una aplicación web desde cero, permitiendo centrar el foco en lo realmente importante del proyecto, además añade una visión más compacta, estructurada y profesional a la solución. Una de las ventajas que aporta utilizar *JHipster* es la generación de nuevas entidades. Como ya se ha mencionado anteriormente, *JHipster* dispone de la posibilidad de generar nuevas entidades a partir de *JHipster Entity*, esta herramienta permite que a partir de un fichero (Listado 4) se generen automáticamente su respectiva tabla en la base de datos *MySQL*, y todos los procesos necesarios para que, desde la aplicación web, esos datos puedan ser consumidos. Esto implica que si en futuros proyectos se decide incluir nuevas entidades para recoger nuevos datos, descargados desde *Smartbike Report Manager* o desde otra fuente, solo es necesario reproducir el contenido del fichero *jhipster-jdl.jh* (Listado 4), para generar una nueva tabla en la base de datos y todas las operaciones necesarias para su explotación.

Como se puede observar en el diagrama de componentes simplificado (Figura 4.9) se dispone de un API REST (Anexo D) por el cual el cliente es capaz de realizar peticiones de información. En este diagrama solo se presentan los componentes más relevantes para la aplicación. En el caso de *"Control de usuarios"* y *"Logger"* son

```
entity Usoestacion {
    nombreCompleto String,
    idEstacion Integer,
    nombreEstacion String,
    fechaDeUso LocalDate,
    intervaloDeTiempo String,
    devolucionTotal Integer,
    devolucionMedia Float,
    retiradasTotal Integer,
    retiradasMedia Float,
    neto Float,
    total Float,
    fechaObtencionDatos LocalDate,
    ficheroCSV String,
    ficheroXLS String,
    hashcode String
}

// Set pagination options
paginate Usoestacion with pagination

dto * with mapstruct
```

Listado 4: Fichero jhipster-jdl.jh.

dos componentes que se han generado a partir de la utilización de *JHipster* pero se consideran relevantes ya que a partir de ellos se puede llevar un mejor control de la aplicación final, mientras que el componente de "Accesos a datos propios" engloba todo lo referente a la información que ha sido almacenada a partir del proceso *ETL*, incluyendo la información de la gestión de las tareas. Dentro de "Accesos a datos propios" también se contempla el caso de que el usuario haga una petición de los ficheros originales almacenados en *Hadoop*.

A continuación, se muestra un ejemplo de las pantallas de la interfaz más relevantes. Se consideran como las más relevantes ya que son las que contienen la información que le interesa gestionar al usuario.

Como se puede observar, se dispone de una interfaz principal (Figura 4.10) que cuenta con una breve explicación de lo que se puede llevar a cabo desde la aplicación. En la parte superior cuenta con los menús desplegables desde donde se puede acceder a la información. La información referente al proyecto se encuentran en la pestaña

"Entidades".

Figura 4.10: Interfaz de inicio de la aplicación web.

Si se despliega la pestaña *"Entidades"* se podra navegar hasta las siguientes opciones; "Gestión de Descargas", "Gestión de Tratamientos", "Gestión de Inserciones" y, ademas, se ha añadido "Uso estaciones".

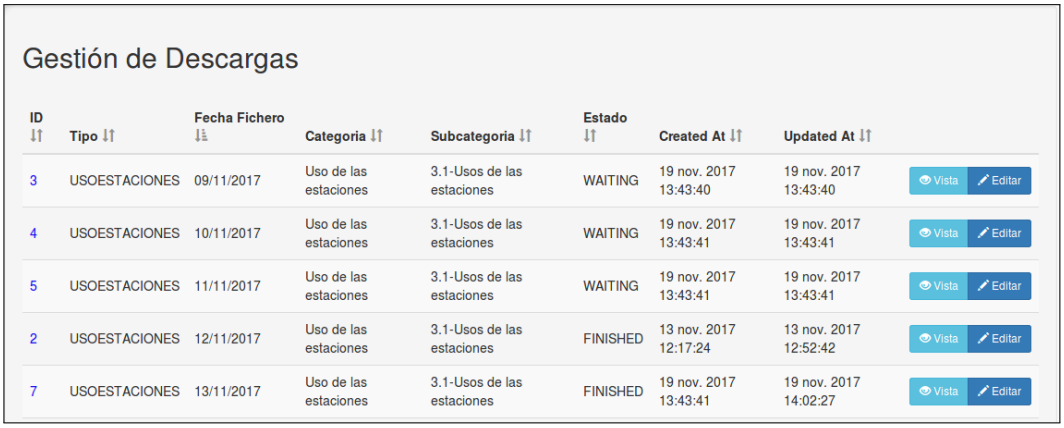

Figura 4.11: Interfaz con el contenido de la gestión de descargas.

En *"Gestión de Descargas"* (Figura 4.11) se muestra el estado de las tareas de descarga. La opción de gestionar esta información es exclusiva para aquellos usuarios que dispongan del rol "Administrador". En este caso se puede observar como existen tareas que están a la espera de ser ejecutadas y otras que han finalizado. Es posible editar estos registros de forma manual, permitiendo modificar el estado de una tarea.

Figura 4.12: Interfaz con el contenido de la gestión de tratamientos.

Figura 4.13: Interfaz con el contenido de la gestión de inserciones.

Para la *"Gestión de Tratamientos"* (Figura 4.12) y la *"Gestión de Inserciones"* (Figura 4.13) se dispone de la misma información y las mismas posibilidades que las expuestas para la *"Gestión de Descargas"*.

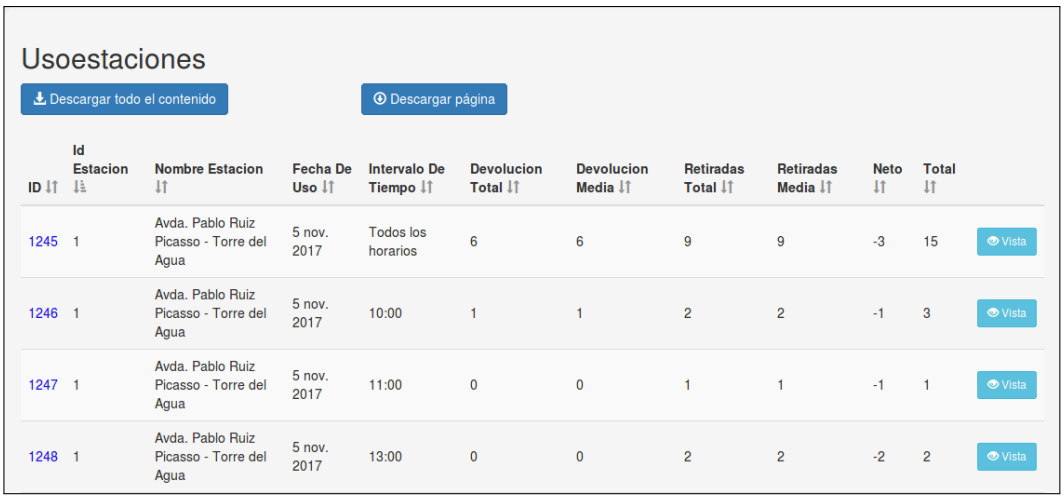

Figura 4.14: Interfaz con el contenido del uso de estaciones de *Bizi Zaragoza*.

Para el caso de ejemplo *"Uso estaciones"* (Figura 4.14) se muestra la información disponible en la base de datos en *MySQL* que ha sido migrada desde *Hadoop*. Esta información se muestra paginada (20 registros por página) y es posible descargar la

totalidad de la información contenida en la base de datos o solo aquella que se este exponiendo en ese momento. En caso de añadir nuevas colecciones de datos, como se ha comentado anteriormente, con el uso de *JHipster Entity* el sistema añadirá esta nueva colección como una nueva entidad del sistema y desplegará todas las operaciones e interfaces necesarias para su explotación desde la aplicación web.

Como se puede observar se cuenta con una interfaz simple y funcional, ya que no entra en los objetivos de este proyecto realizar una interfaz que contenga mayor complicación, ya que lo importante es la posibilidad de consultar y extraer datos de forma clara y rápida. El desarrollo de una interfaz más completa podría contemplarse como un objetivo en futuros proyectos de Trabajo Fin de Grado.

Capítulo 5

Organización y gestión del proyecto

5.1. Planificación del proyecto

Como se puede apreciar en el ciclo de vida del proyecto (Figura 5.1) durante el desarrollo se ha pasado por diferentes fases hasta llegar al momento final de entrega del producto. A grandes rasgos, se cuenta con cinco etapas, comenzando por la selección del proyecto hasta terminar con una versión del proyecto lista para su puesta en producción y la documentación acerca del proyecto. A continuación se procede a explicar cada una de las etapas reflejadas.

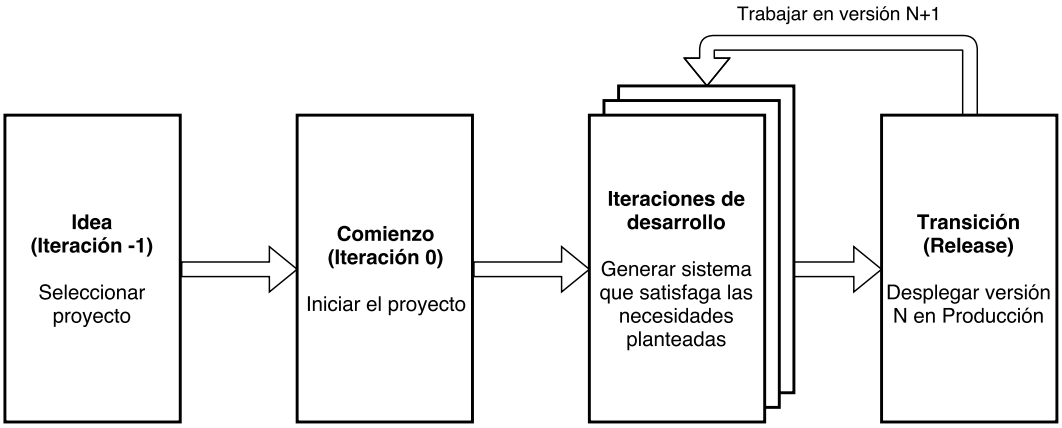

Figura 5.1: Ciclo de vida de desarrollo de un sistema ágil

5.1.1. Idea

Durante esta fase, también conocida como *"Iteración -1"*, se barajan las posibles opciones disponibles para realizar un proyecto que sea factible de ser utilizado como Trabajo Fin de Grado. Una vez fijado el proyecto a desarrollar, *Bizi2Data*, se realiza una prueba de concepto que sirve como una primera aproximación muy simplificada del problema que se aborda. Esta primera aproximación sirve para descubrir si el proyecto

es factible o las exigencias sobrepasan las capacidades de quien lo va a desarrollar. Las conclusiones extraidas en esta fase se reflejan en el análisis del sistema (Capítulo 2).

5.1.2. Comienzo

En esta fase, también conocida como *"Iteración 0"*, se realiza el modelado de la arquitectura del sistema y se eligen las tecnologías a usar a partir de la información que se haya obtenido del análisis del sistema (Capítulo 2). Una vez realizado todo el diseño del sistema se realiza una primera estimación de costes del proyecto (Sección 5.4). El modelado del sistema que se obtiene en esta fase es incluido en el diseño del sistema (Capítulo 3).

5.1.3. Desarrollo

La fase de desarrollo consta de varias iteraciones ya que para la implementación de cada componente del sistema se destina un plazo para configuraciones, codificación y pruebas. Es la que mayor carga de trabajo engloba, ya que se prevé y asume que las tareas a realizar conlleverán modificaciones que alteren las especificaciones recogidas en la fase de diseño.

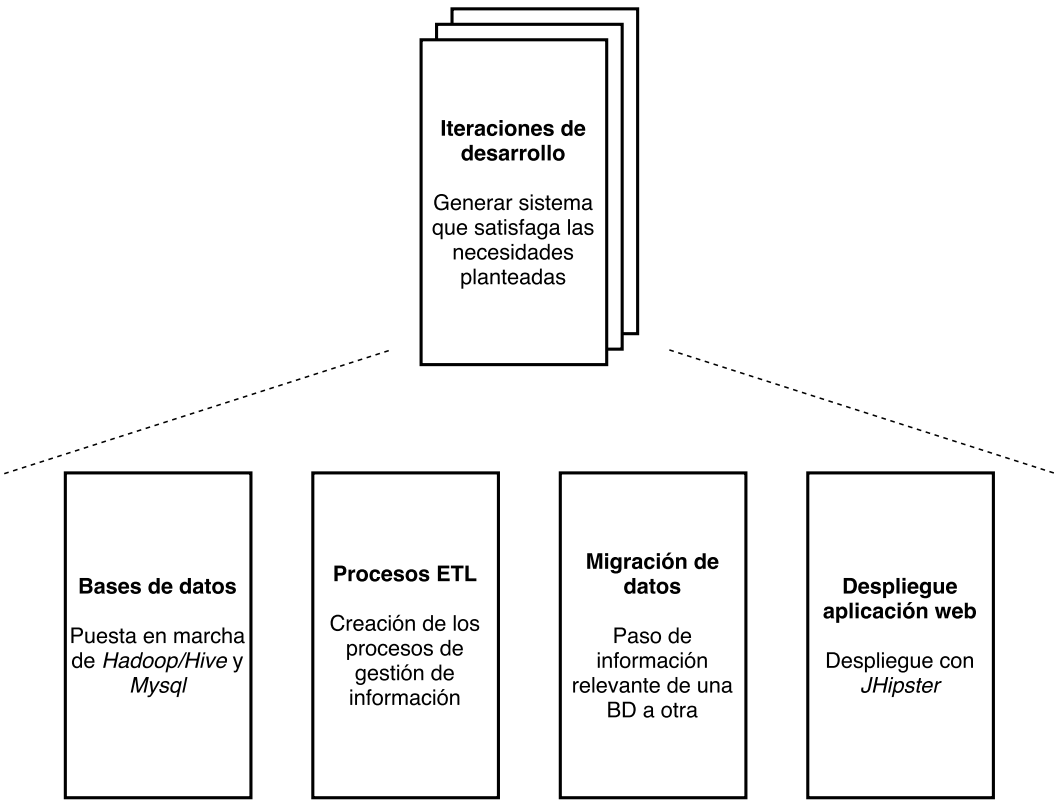

Figura 5.2: Iteraciones más relevantes de la fase de desarrollo.

Como se puede observar en la Figura 5.2, donde se muestran las iteraciones más relevantes dentro del proyecto, no se dispone de un orden lineal, ya que durante cualquier iteración de esta fase se puede volver asumir que es necesario saltar (para implementar o modificar) al desarrollo previsto para otra iteración. Todo lo relacionado con la fase de desarollo se recoge dentro de la implementación del sistema (Capítulo 4).

5.1.4. Transición

Una vez obtenida una solución que se considere entregable, que satisfaga con los requisitos definidos, se genera una versión de *"Release"* (Sección 5.2.4) indicando la versión que está lista para ser puesta en producción.

A partir de esta versión se realizarán futuras modificaciones, también pueden servir como punto de partida para futuros Trabajos Fin de Grado, que pueden aumentar o modificar el trabajo realizado. En esta fase se recolecta toda la documentación, que se ha ido realizando en paralelo, sobre el proyecto *Bizi2Data* y se redacta la memoria que sirve para realizar el deposito del Trabajo Fin de Grado.

5.2. Gestión de configuraciones

5.2.1. Sistema

La fase en la etapa de desarrollo se ha realizado sobre un ordenador portatil con el sistema operativo *Ubuntu 16.04 x86_64*, con un procesador *Intel CORE i3* y 12 *GB* de memoria *RAM*. En este sistema se ha desplegado un contenedor *Docker*, creado por la empresa *Big Data Europe* (`https://www.big-data-europe.eu`), utilizando *docker-compose* versión 1.12.0, que contiene *Apache Hadoop* versión 2.8 [18]. Para el desarrollo de procesos se ha utilizado el lenguaje de programación *Java* en la versión 8, disponiendo así de la última versión disponible (en el momento de iniciar el proyecto). En cuanto al almacenamiento de los datos que son expuestos en la aplicación web se ha optado por utilizar *MySQL* versión 5.7.18.

La versión de *JHipster* empleada para el despliegue de la aplicación web ha sido la 4.6.1 y se ha generado con la siguiente configuración:

– Aplicación monolitica: modelo de aplicación local, de un solo nodo, fácil de desarrollar y usar.

– JWT authentication (stateless, with a token): estándar abierto (RFC-7519) basado en JSON para crear un token que sirva para enviar datos entre aplicaciones o servicios y garantizar que sean válidos y seguros.

– MySQL como base de datos en desarrollo.

– Compilado y gestión de dependencias con Gradle.

– AngularJS (versión 1).

5.2.2. Composición del sistema en ejecución

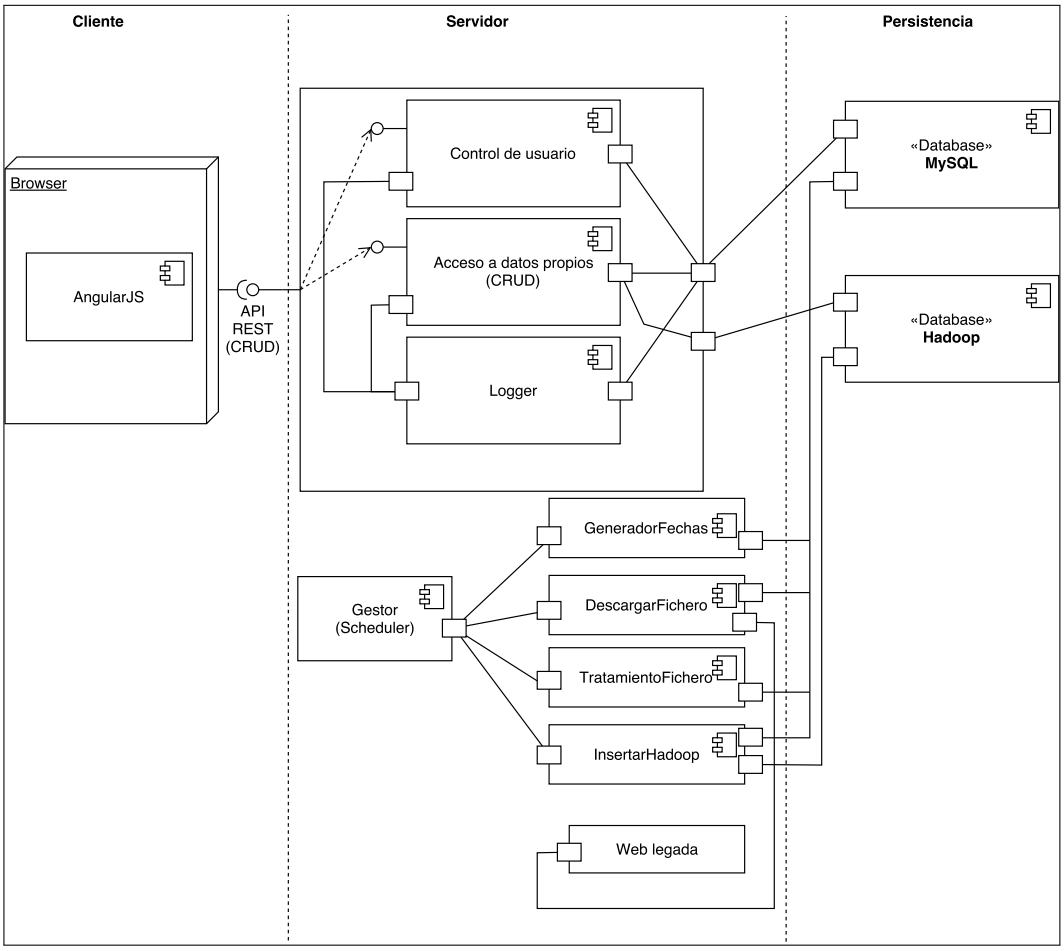

Figura 5.3: Diagrama de componentes del sistema en ejecución.

En el diagrama de componentes del sistema (Figura 5.3) se muestran los componentes más relevantes que interactuan mientras la aplicación se encuentra en ejecución. Se puede observar como a partir de un *API REST* (Anexo D) el cliente puede realizar peticiones sobre información que se encuentra almacenada en la capa de persistencia. En la capa del servidor se puede observar como se dispone de un gestor encargado de contactar con los procesos *ETL* (Sección 4.2.3) que deben ser lanzados

periódicamente y las relaciones que existen entre ellos y las distintas bases de datos.

El componente *"Control de usuarios"* aporta la posibilidad de manejar distintas cuentas de usuarios, que disponen de distintos permisos según el rol que desempeñan en la aplicación. Este componente se genera automáticamente a partir de la creación de la aplicación *JHipster*, pero se asume como muy relevante para la solución por aportar un mayor control de acceso a los datos.

El "Acceso a datos propios" gestiona las peticiones de información referente a los datos sobre el servicio *Bizi Zaragoza*, los registros de las tareas, almacenados ambos casos en *MySQL*, y gestiona la petición de los ficheros originales almacenados en *Hadoop*.

Es interesante mencionar el componente *"Logger"* ya que se trata de un componente que da a la solución la posibilidad de tener auditadas todas las acciones que se realizan en el sistema, lo cual favorece que el sistema disponga de una buena trazabilidad en caso de ocurrir algún error.

Se puede concluir que se dispone de un sistema robusto en el cual todas las tareas realizadas están controladas y auditadas correctamente, separando las posibilidades de acceso según un control de usuarios que permite diseñar políticas de acceso a los recursos.

5.2.3. Control de versiones

El proyecto se ha gestionado a partir de un repositorio creado en *GitHub* (`https://github.com/DaniCab/UNIZAR-TFG-BIZI`). En este repositorio se ha generado la estructura del proyecto sobre la que se ha trabajado en cada una de las fases de desarrollo. Dado que se trabaja a partir de la rama *master* se ha implantado que la gradularidad de los *"commits"* debe ser por cada funcionalidad implementada y realizando un *"commit"* al finalizar la jornada de trabajo diario. Esta decisión es tomada por motivos de comodidad a la hora de recuperar alguna versión anterior en caso de fallo en el proyecto.

5.2.4. Versión en producción

Una vez obtenida una solución que da una funcionalidad óptima, que cumple con los requisitos definidos, se despliega en *GitHub* la opción de generar un *Release*. A

partir de ese momento, es posible descargar la última versión estable para producción desde el mismo repositorio. A su vez, esto permite seguir trabajando sobre el proyecto en producción e ir generando nuevas versiones a medida que se implementen nuevas funcionalidades.

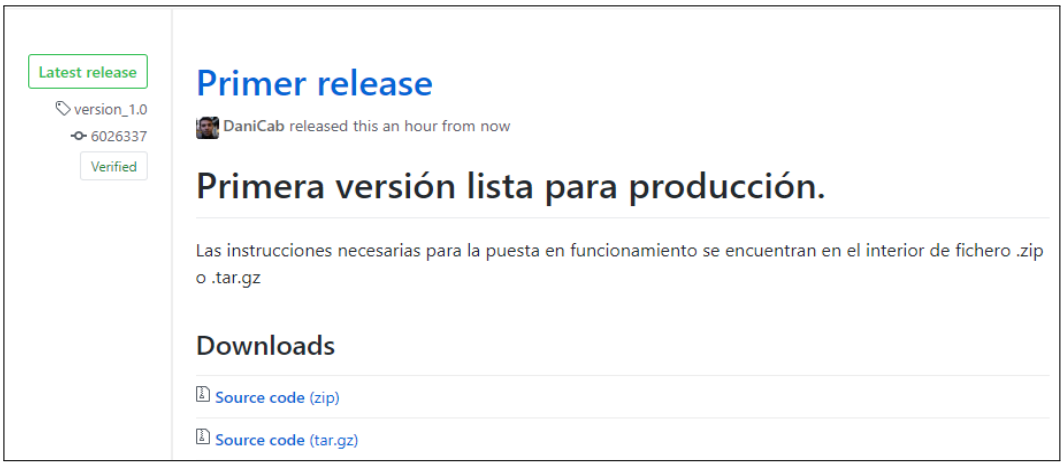

Figura 5.4: Versión Release en *GitHub*.

Como se puede observar en la Figura 5.4 se cuenta con la posibilidad de descargar en un fichero comprimido (*.zip* o *.tar.gz*) todo lo necesario para ejecutar la aplicación. Al igual que en desarrollo, la versión en producción necesita de un fichero de configuración (Anexo C.2) que debe ser añadido y completado antes de su puesta en funcionamiento.

5.2.5. Gestión de tareas del proyecto

Como apoyo para la gestión de tareas se ha utilizado el *Kanban* disponible en *GitHub* (Figura 5.5), disponiendo de un tablero en el cual se incluyen las tareas a realizar con una granularidad a nivel de proceso. Para que una tarea se considere como finalizada (*"Done"*) debe haber sido validado su correcto funcionamiento. En caso de que una tarea finalizada deba volver a ser revisada o modificada se colocará en el apartado *"Review"*.

Dado que el *kanban* del proyecto puede ser visualizado por terceras personas, en las tareas se han realizado comentarios acerca de posibles fallos ocurridos durante el desarrollo y las medidas adoptadas para solucionarlas (Figura 5.6). Como apoyo adicional para el control de versiones, se ha creado un documento de texto (Figura 5.7) en el que se recogen todos los aspectos importantes realizados a lo largo de la jornada de trabajo, sirviendo también como una pieza clave a la hora de recolectar las horas invertidas en cada tarea del proyecto.

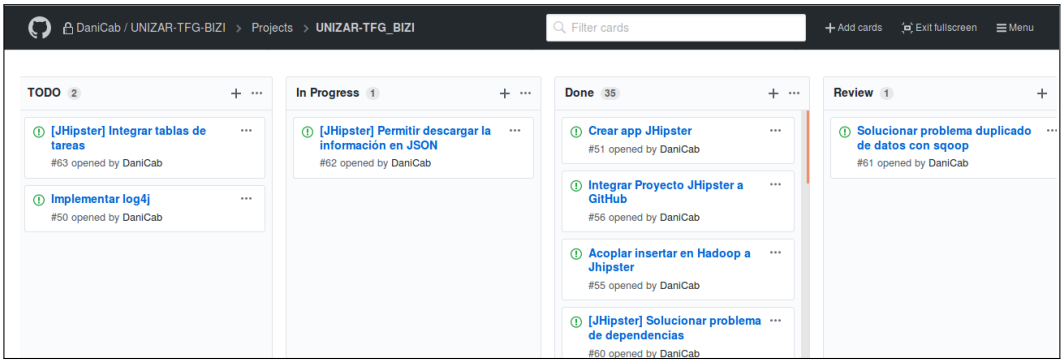

Figura 5.5: Kanban de tareas en *GitHub*.

Figura 5.6: Anotaciones de ayuda en las tareas de *GitHub*.

```
01/11
13:00 - INICIO
    Comienzo de acoplamiento de insercion en base de datos Hadoop
    Sqoop necesita que se le diga las columnas a rellenar, en caso contrario no inserta bien, debido a que
    sqoop export --connect jdbc:mysql://192.168.1.36:3306/jhipsterBizi --table usoestacion --export-dir /u
    Buscar solución para que al realizar el sqoop no duplique lo que ya está contenido:
        - Poner todas las columnas en mysql como PK????
15:00 - COMIDA
16:30 - VUELTA
    Acoplamiento de insertarHadoop en jhipster
    Probar a lanzar todas las tareas con @Scheduled(cron = "...")
    Problema con dependencias a la hora de añadir a gradle las entradas necesarias para conectar con hive
18:30 - FIN
```

Figura 5.7: Ejemplo de anotaciones diarias para la gestión del proyecto.

5.2.6. Gestión de documentación

Para la gestión de la documentación del proyecto se ha utilizado la plataforma *Overleaf* (`https://www.overleaf.com/`), editor de documentos en *LaTex* que dispone de gestión automática de versiones del documento realizado. Para la gestión de imagenes, diagramas de la memoria, y para el control de esfuerzos, se ha creado una carpeta en *Google Drive*.

5.3. Esfuerzos realizados

En la Tabla 5.1 se refleja, de manera global, el computo de horas que realmente se han utilizado para cada una de las fases del proyecto. Esto ha sido posible de gestionar gracias a que durante toda la realización del proyecto se ha llevado una contabilidad de las horas utilizadas en cada tarea a partir de anotaciones internas. Las horas consumidas para reuniones con el director del proyecto y las horas dedicadas a la redacción de la memoria del proyecto se han incluido dentro del computo de horas de las fases del proyecto.

Concepto	**Horas invertidas**
Análisis del sistema (Capítulo 2)	97 horas
Diseño del sistema (Capítulo 3)	87 horas
Implementación de infraestructura (Capítulo 4)	102 horas
Implementación del "*workflow* plantilla" (Capítulo 4)	75 horas
TOTAL	**361 horas**

Tabla 5.1: Horas invertidas en el proyecto.

Viendo el despligue de horas utilizadas para la consecución del proyecto (Tabla 5.1), se puede observar que la decisión tomada en la fase de análisis (Capítulo 2) de focalizar la solución al desarrollo de un *workflow* que pueda servir como plantilla en futuros proyectos, se ve corroborada por las horas que han sido necesarias para dar una solución que satisfaga las necesidades planteadas. Esto beneficia que en futuros proyectos la tarea de generar nuevos recursos para la solución verán su coste temporal reducido, gracias a disponer de esta plantilla como base. Además, se podrán enfocar los objetivos más orientados a la obtención y exposición de la información, al contar con una infraestructura diseñada pensando en que la escalabilidad no suponga un problema.

5.4. Valoración económica del proyecto

5.4.1. Coste del proyecto

Para el cálculo del coste del proyecto se ha asumido que el coste medio por horas de trabajo es de 26,43 €/h. En la Tabla 5.2 se puede observar un resumen de los *items* que se han realizado a lo largo del proyecto al igual que otros costes derivados.

Tarea/Componente	**Horas**	**Coste (€)**
Análisis del sistema	97 horas	2.563,71 €
Diseño del sistema	87 horas	2.299,41 €
Implementación de infraestructura	102 horas	2.695,86 €
Implementación del *"workflow" plantilla*	75 horas	1.982,25 €
TOTAL Tareas/Componentes	**361 horas**	**9.514,80 €**
Gestión (G)	$361\,h \cdot 0,15$	1.431,19 €
Gestión de configuraciones (GC)	$361\,h \cdot 0,05$	477,06 €
Aseguramiento de la calidad (AC)	$361\,h \cdot 0,07$	667,90 €
TOTAL GESTIÓN Y CALIDAD	**98 horas**	**2.576,15 €**
Transporte (T)	$60\,viajes \cdot 2,70$ €	162,00 €
TOTAL MACROS	$G + GC + AC + T$	**2.738,15 €**
Amortización estaciones de trabajo	$(361\,h + 98\,h) \cdot \frac{400}{361\,h}$	551,90 €
TOTAL		**12.804,85 €**

Tabla 5.2: Estimación de costes del proyecto Bizi2Data.

5.4.2. Precio del proyecto

A partir de la estimación de costes expuesta en la Tabla 5.2, se ha decidido asumir un beneficio para el proyecto del 35 %. Por lo que el precio del proyecto es el siguiente (Tabla 5.3).

SOLUCIÓN	**CANTIDAD**
BASE (precio)	**17.286,55 €**
IVA Admón.	3.630,18 €
TOTAL (Precio)	**20.916,73 €**

Tabla 5.3: Precio del proyecto Bizi2Data.

Capítulo 6

Conclusión

Durante el desarrollo de "*Bizi2Data. Apertura para la investigación de un sistema legado.*" se han generado ideas, visiones y aprendizajes que pueden considerarse como conclusiones validas para este proyecto.

Ha sido necesario realizar un análisis en frío para acotar las dimensiones del proyecto, ya que la primera visión obtenida por las conferencias con el grupo *GEOT* son demasiado ambiciosas como para ser abarcadas por un único Trabajo Fin de Grado. Por esto, se ha realizado un sistema que pueda servir de base para futuros desarrollos que integren todo tipo de información relacionada con *Bizi Zaragoza*. Esto implica que el proyecto se haya centrado principalmente en la creación de la plataforma y en un proceso de extracción/almacenamiento/tratamiento/explotación de referencia que pueda ser reutilizado y ampliado en un futuro. De esta forma, el sistema que se ha desarrollado puede servir de base para futuros proyectos, ya sean desarrollados como TFG o como proyectos de colaboración con GEOT, para aportar nuevas funcionalidades o expandan las capacidades actuales.

Una vez finalizado este proyecto se dispone de un sistema robusto, funcional y facilmente replicable que es capaz de realizar, de forma automática, los trabajos que para el grupo *GEOT* suponían un problema por su complejidad (obtención y tratamiento de los datos de la web *Smartbike Report Manager* para su posterior análisis). El sistema también da soporte para que la información recolectada sea expuesta en una nueva aplicación web donde se puede consultar y extraer de una forma simple en comparación con la opción anterior (*Smartbike Report Manager*).

El trabajo realizado con las diferentes tecnologías usadas (*JHipster*, *Spring Framework*, *MySQL*, *Apache Hadoop*, entre otras) y la interrelación entre ellas ha concluido en un sistema robusto capaz de cumplir con los objetivos y requisitos

marcados en el inicio del proyecto, dando una solución que se caracteriza por contar finalmente con un resultado sencillo de utilizar para el usuario, y que puede ser utilizado como base para siguientes proyectos, ya que ha sido diseñado pensando en que puedar ser facilmente escalable.

En referencia al servicio de bicicletas de *Zaragoza*, durante los análisis realizados para la consecución del proyecto, se estima que sería necesario que el *Ayuntamiento de Zaragoza* de mayor cobertura a los datos referentes a este ámbito, ya que se trata de una entidad que apuesta por los datos datos abiertos y, a parte, se trata de un servicio muy utilizado por la ciudadanía en general, lo que hace que se generen continuamente datos para el estudio y mejora del servicio.

Desde el punto de vista personal, este proyecto ha aportado una visión real de lo que es asumir el desarrollo de un proyecto profesional que, a diferencia de los proyectos realizados para asignaturas del grado, ha sufrido modificaciones y cambios de rumbo a lo largo de su ejecución. Además, aporta una experiencia positiva el realizar un trabajo que se sabe que será la base para futuros proyectos y que tiene como destino mejorar el ámbito de trabajo de un grupo de investigación, en este caso el *GEOT*.

Capítulo 7

Bibliografía

[1] AngularJS. What is angularjs? `https://docs.angularjs.org/guide/introduction`, 2017. [Online; accedido 19-Noviembre-2017].

[2] Wikipedia. Apache poi — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Apache_POI`, 2017. [Online; accedido 28-Octubre-2017].

[3] Wikipedia. Bootstrap (framework) — Wikipedia, the free encyclopedia. `https://es.wikipedia.org/wiki/Bootstrap_(framework)`, 2017. [Online; accedido 19-Noviembre-2017].

[4] Google. Chromedriver - webdriver for chrome. `https://sites.google.com/a/chromium.org/chromedriver/`, 2016. [Online; accedido 15-Noviembre-2017].

[5] Wikipedia. cron (unix) — Wikipedia, the free encyclopedia. `https://es.wikipedia.org/wiki/Cron_(Unix)`, 2017. [Online; accedido 16-Noviembre-2017].

[6] Wikipedia. Docker (software) — Wikipedia, the free encyclopedia. `https://es.wikipedia.org/wiki/Docker_(software)`, 2017. [Online; accedido 28-Octubre-2017].

[7] Panos Vassiliadis. A survey of extract-transform-load technology. *International Journal of Data Warehousing and Mining*, 2009.

[8] Wikipedia. Jhipster — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/JHipster`, 2017. [Online; accedido 28-Octubre-2017].

[9] Dai Clegg and Richard Barker. *Case Method Fast-Track: A RAD Approach*. Addison-Wesley, 2004. ISBN: 978-0-201-62432-8.

[10] Wikipedia. Transferencia de estado representacional — Wikipedia, the free encyclopedia. `https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional`, 2017. [Online; accedido 28-Octubre-2017].

[11] Selenium Project. Selenium documentation. http://www.seleniumhq.org/docs/01_introducing_selenium.jsp, 2017. [Online; accedido 28-Octubre-2017].

[12] Spring project. Spring framework. https://projects.spring.io/spring-framework/, 2017. [Online; accedido 15-Noviembre-2017].

[13] Jummp's Blog. Método moscow. https://jummp.wordpress.com/2013/04/27/metodo-moscow/, 2013. [Online; accedido 14-Noviembre-2017].

[14] JHipster. Creating an entity. http://www.jhipster.tech/creating-an-entity/, 2017. [Online; accedido 15-Octubre-2017].

[15] Spring. Integration. https://docs.spring.io/spring/docs/5.0.1.RELEASE/spring-framework-reference/integration.html#spring-integration, 2017. [Online; accedido 15-Noviembre-2017].

[16] The Apache Software Foundation. Sqoop user guide (v1.4.6). https://sqoop.apache.org/docs/1.4.6/SqoopUserGuide.html, 2017. [Online; accedido 10-Noviembre-2017].

[17] Mario Pérez. Cómo permitir el acceso remoto a una base de datos mysql. https://geekytheory.com/como-permitir-el-acceso-remoto-a-una-base-de-datos-mysql, 2015. [Online; accedido 10-Septiembre-2017].

[18] Ivan Ermilov. docker-hive. https://github.com/big-data-europe/docker-hive/tree/2.1.0-postgresql-metastore, 2016. [Online; accedido 20-Noviembre-2017].

[19] Camunda Services GmbH. Bpmn 2.0 implementation reference. https://docs.camunda.org/manual/7.7/reference/bpmn20/, 2016. [Online; accedido 15-Noviembre-2017].

Lista de Figuras

Lista de Tablas

Anexos

Anexos A

Análisis del Sistema

En este anexo se presenta, en formato de tablas, el análisis de datos extraido a partir de la interfaz del servicio web legado. Este análisis ha servido para realizar un estudio sobre los datos que pueden ser aportados a la solución final en esta fase del proyecto, los datos que no son relevantes y/o interesantes para este estudio, pero que si podrían ser incluidos en estudios futuro, y los datos que por su poca relevancia no se tendrán en cuenta.

Tras el análisis de los datos obtenidos a partir de la interfaz del servicio web, se han sacado las siguientes conclusiones:

- El sistema cuenta con secciones que contienen datos irrelevantes o inconscistentes, para la solución buscada, por lo que no serán incluidas en la solución final.
- El sistema cuenta con mucha información redundante por lo que se decide optar por la descarga de las búsquedas más completas, obviando las que puedan ser derivadas de éstas.
- Tras observar los datos obtenidos se puede deducir que la interfaz está devolviendo la vista de una tabla de hechos. Esto facilita el tratamiento de los datos ya que se entiende que la información suministrada es la relevante para el análisis.
- Las búsquedas de "Usos de las estaciones" y "Matriz de movimiento", son el eje principal de la solución final, ya que cuentan con la información necesaria para alcanzar los objetivos.
- Búsquedas como "Perfil - Postal Code" son desechadas de esta solución ya que no existe un control sobre la veracidad de los datos por parte del sistema legado.

A partir de este análisis de la interfaz y lo datos que es capaz de aportar, se ha concluido que la posible estructura del sistema es el siguiente:

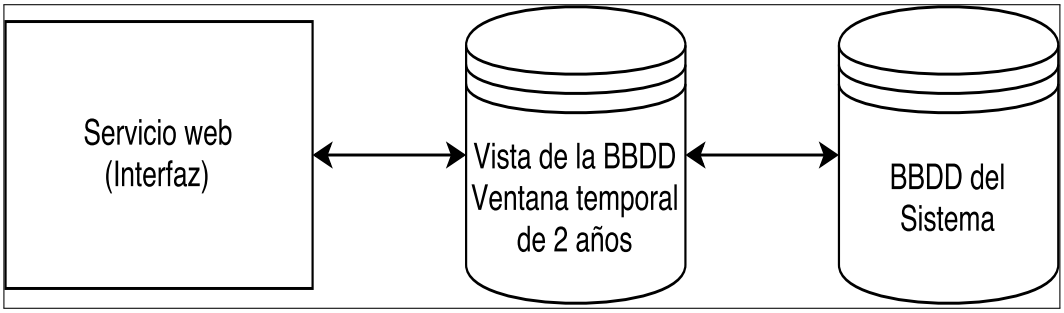

Figura A.1: Posible arquitectura del sistema legado

Como se puede ver en la Figura A.1, la interfaz web solo tiene posibilidad de acceder a los datos de una vista de la base de datos, y está acotada dentro de una ventana temporal de dos años, por lo que la decisión más óptima es obtener la información de más antigua a más actual, obteniendo así un mayor almacenamiento histórico.

A partir de los datos estudiados también se puede intuir un posible modelo de la base de datos del sistema, por supuesto hay que recalcar que se trata de una aproximación a partir del análisis, el cual sólo deja ver una pequeña parte del sistema general. En la Figura A.2 se puede observar esta aproximación a partir de un diagrama Entidad-Relacion.

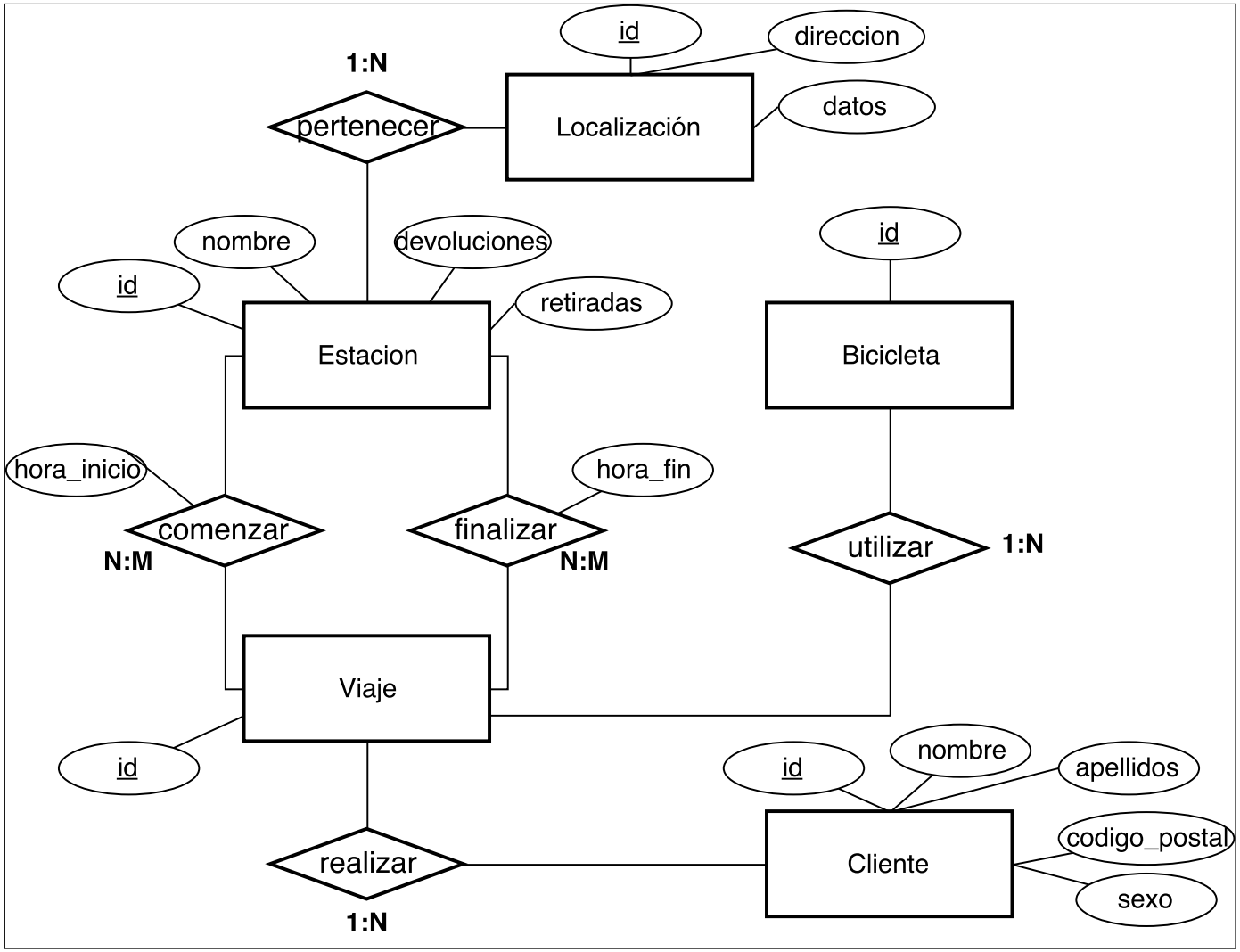

Figura A.2: Posible esquema E-R del sistema legado

Para obtener la información de cada sección es necesario rellenar previamente un formulario, a continuación se exponen los diferentes campos presentes en los formularios:

Nombre	**Rango de búsqueda**	**Descripción**
Fecha Inicio	Desde 2 años antes de la fecha actual hasta día actual - 1.	Fecha desde la que se comienza la búsqueda.
Fecha Fin	Desde 2 años antes de la fecha actual hasta día actual - 1.	Fecha hasta la que se realiza la búsqueda.
Día	"select all", "Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo".	Selección de día de la semana a mostrar.
Usuarios	"Usuarios abonados" o "Usuario activos".	Tipo de usuario registrado en el sistema.
Edad Inicial	ENTERO	Edad a partir de la que se comienza la búsqueda.
Edad Final	ENTERO	Edad hasta la que se realiza la búsqueda.
Rango de edad	ENTERO POSITIVO	Define la agrupación de edades para el resultado.
Tipo Área	"Todos", "Cluster", "SubCluster", "Crown", "Zone".	Tipos de áreas definidas en el sistema.
Área	"Todos", ["CL1", "CL2", ···, "CL23"], ["01.01", "01.02", ···, "23.02"], "ZaragozaCrown"	Subconjuntos según el tipo de área seleccionado.
Distrito	"Todos" o "Todos los distritos".	Distritos por lo que el sistema divide la localización de las estaciones.
Estación	"select all" o selección de una o varias estaciones a partir de su "id - nombre de estación".	Estación o estaciones sobre las que se realizan las consultas.
Uso	"Total", "Devoluciones", "Recogidas".	Tipo de operación realizada en la estación que se desea consultar.
Bicis contratadas	NUMÉRICO	Número de bicicletas contratadas para el servicio.

Tabla A.1: Campos de entrada de los formularios

A.1. Número de usos de las bicis

Número de usos de las bicis

Formulario de búsqueda: "Fecha Inicio" y "Fecha Fin".

Resultados:

– Detalles de uso de bicicletas en el periodo comprendiddo del dd/MM/aaaa al dd/MM/aaaa.

Nombre	Tipo de dato	Formato	Descripción
Media de bicis en servicio	REAL	#.###,##	Media de bicicletas disponibles en el rango de fechas introducido.
Media de bicis distintas usadas	REAL	#.###,##	Media de bicicletas distintas que han sido usadas en el rango de fechas introducido.
Total usos en el periodo	ENTERO	#.###	Número de usos de bicicletas en el rango de fechas introducido.
Media de usos por bici por día	REAL	#.###,##	Media de usos de las bicicletas por día dentro del rango introducido.
Ratio de bicis usadas	REAL	#.###,## %	Porcentaje de bicicletas usadas en el rango de fechas introducido.
Media de usuarios totales	ENTERO	#.###	Media de usuarios que han utilizado alguna bicicleta en el rango de búsqueda.
Total de usuarios distintos por día	ENTERO	#.###	Número de usuarios que han usado el servicio en el rango de búsqueda.
Media de usuarios distintos por día	ENTERO	#.###	Media de usuarios que han usado el servicio en el rango de búsqueda.
Media de usos por usuario por día	REAL	#,##	Media de usos de bicicletas por los usuarios durante un día.

Tabla A.2: Resultado búsqueda Número de usos de las bicis. Detalles.

– Usos por día/Media de usos por bici.

Nombre	**Tipo de dato**	**Formato**	**Descripción**
Día	FECHA	dd/MM/aaaa	Muestra las fechas presentes dentro del rango de búsqueda.
Total de usos	ENTERO	#.###	Total de usos de bicicletas en el día señalado.
Total bicis en servicio	ENTERO	#.###	Total de bicicletas que se encuentran en servicio ese día.
Media de usos por bici	REAL	#.###,##	Media de usos por bicicletas de ese día concreto.

Tabla A.3: Resultado búsqueda Número de usos de las bicis. Usos por día.

Las búsquedas en rangos superiores a dos días son demasiado costosas temporalmente, incluso llegan a quedarse colgadas.

Uso de las bicis por día de la semana

Formulario de búsqueda: "Fecha Inicio" y "Fecha Fin".

Resultados:

– Usos de bicis por días de la semana del dd/MM/aaaa al dd/MM/aaaa.

Nombre	Tipo de dato	Formato
Lunes	COMPUESTO	#.### (#%)
Martes	COMPUESTO	#.### (#%)
Miércoles	COMPUESTO	#.### (#%)
Jueves	COMPUESTO	#.### (#%)
Viernes	COMPUESTO	#.### (#%)
Sábado	COMPUESTO	#.### (#%)
Domingo	COMPUESTO	#.### (#%)

Tabla A.4: Resultado búsqueda Uso de las bicis por día de la semana.

Uso de las bicis por intervalos horarios

Formulario de búsqueda: "Fecha Inicio", "Fecha Fin" y "Día".

Resultados:

– Usos de bicis por intervalos horarios del dd/MM/aaaa al dd/MM/aaaa.

Nombre	Tipo de dato	Formato	Descripción
Franja horaria	CADENA	hh:00 o Totales	Intervalo horario granulado por horas.
Usos	ENTERO	#.###	Usos de las bicicletas durante esa franja horaria.
Media de usos por día	REAL	#.###,##	Media de uso en relación al día completo.
Porcentaje	REAL	#.###,##%	Porcentaje de utilización de bicicletas con respecto al día completo.

Tabla A.5: Resultado búsqueda Uso de las bicis por intervalos horarios.

Tiempo medio de uso de las bicis por días de la semana

Formulario de búsqueda: "Fecha Inicio" y "Fecha Fin".

Resultados:

– Tiempo medio de usos de las bicis por días de la semana del dd/MM/aaaa al dd/MM/aaaa.

Nombre	Tipo de dato	Formato	Descripción
Tiempo medio de uso por bici	DURACIÓN	#m #s	Tiempo medio total del uso de las bicicletas en el rango introducido.
Lunes	DURACIÓN	#m #s	Media de uso de las bicicletas los lunes que se encuentran dentro del rango de búqueda.
Martes	DURACIÓN	#m #s	Media de uso de las bicicletas los martes que se encuentran dentro del rango de búqueda.
Miércoles	DURACIÓN	#m #s	Media de uso de las bicicletas los miércoles que se encuentran dentro del rango de búqueda.
Jueves	DURACIÓN	#m #s	Media de uso de las bicicletas los jueves que se encuentran dentro del rango de búqueda.
Viernes	DURACIÓN	#m #s	Media de uso de las bicicletas los viernes que se encuentran dentro del rango de búqueda.
Sábado	DURACIÓN	#m #s	Media de uso de las bicicletas los sábado que se encuentran dentro del rango de búqueda.
Domingo	DURACIÓN	#m #s	Media de uso de las bicicletas los domingo que se encuentran dentro del rango de búqueda.

Tabla A.6: Resultado búsqueda Tiempo medio de uso por días de la semana.

Anotación: m (minutos), s (segundos)

Estadísticas de usuarios distintos por día

Formulario de búsqueda: "Fecha Inicio" y "Fecha Fin".

Resultados:

– Frecuencia de uso del dd/MM/aaaa al dd/MM/aaaa.

Nombre	Tipo de dato	Formato	Descripción
Nº de usos	CADENA	# Uso	Número de veces que un usuario distinto a utilizado el servicio dentro del rango selecionado.
Rango de edad	CADENA	"Todas las edades', " " o [rango definido]	Media de uso de las bicis los lunes que se encuentran dentro del rango.
Lunes	DURACIÓN	#m #s	Media de uso de las bicis los lunes que se encuentran dentro del rango.
Martes	DURACIÓN	#m #s	Media de uso de las bicis los martes que se encuentran dentro del rango.
Miércoles	DURACIÓN	#m #s	Media de uso de las bicis los miércoles que se encuentran dentro del rango.
Jueves	DURACIÓN	#m #s	Media de uso de las bicis los jueves que se encuentran dentro del rango.
Viernes	DURACIÓN	#m #s	Media de uso de las bicis los viernes que se encuentran dentro del rango.
Sábado	DURACIÓN	#m #s	Media de uso de las bicis los sábado que se encuentran dentro del rango.
Domingo	DURACIÓN	#m #s	Media de uso de las bicis los domingo que se encuentran dentro del rango.

Tabla A.7: Resultado búsqueda Estadísticas de usuarios distintos por día.

Los rangos definidos son: [14-20],[21-30],[31-45],[46-60],[61-100],[100+].

Aparecen usos por usuarios en el rango [100+], lo cual podría se fallo del sistema ya que por la naturaleza del problema es dificil ver a una persona de mas de 100 años cogiendo una bicicleta. Tambien existen algunos resultados sin ningun rango de edad lo cual habría que saber el porqué.

Anotación: m (minutos), s (segundos).

A.2. Abonados

Abonados

Formulario de búsqueda: "Fecha Inicio" y "Fecha Fin".

Resultados:

– Total hasta el "Fecha Fin".

Nombre	Tipo de dato	Formato	Descripción
Abonados anuales	ENTERO	#.###	Nº de clientes con un abono anual hasta la fecha.
Abonados actuales activos	ENTERO	#.###	Nº de clientes con un abono activo hasta la fecha.
Ratio de renovación	REAL	#,## %	Porcentaje de renovaciones realizadas hasta la fecha.
Abonados anuales renovados	ENTERO	#.###	Nº de abonados que han renovado su abono anual hasta la fecha.
Abonados anuales cancelados	ENTERO	#.###	Nº de abonados que han cancelado su abono anual hasta la fecha.
Abonados con tarjeta operativa	ENTERO	#.###	Nº de abonados que disponen de una tarjeta operativa hasta la fecha.
Ratio de tarjetas operativas	REAL	#,## %	Porcentaje de tarjetas operativas hasta la fecha.
Abonados temporales nuevos	ENTERO	#.###	Nº de abonos temporales expedidos hasta la fecha.
Abonados temporales activos	ENTERO	#.###	Nº de abonos temporales activos hasta la fecha.

Tabla A.8: Resultado búsqueda Abonados. Total de abonados.

– En el periodo de dd/MM/aaaa al dd/MM/aaaa.

Nombre	**Tipo de dato**	**Formato**	**Descripción**
Abonados anuales nuevos	ENTERO	#.###	Nº de clientes que han obtenido un abono anual en el rango de búsqueda.
Abonados actuales activos	ENTERO	#.###	Nº de clientes con un abono activo en el rango de búsqueda.
Ratio de renovación	REAL	#,## %	Porcentaje de renovaciones realizadas durante el rango de búsqueda.
Abonados anuales renovados	ENTERO	#.###	Nº de abonados que han renovado su abono anual durante el rango de búsqueda.
Abonados anuales cancelados	ENTERO	#.###	Nº de abonados que han cancelado su abono anual durante el rango de búsqueda.
Abonados con tarjeta operativa	ENTERO	#.###	Nº de abonados que disponen de una tarjeta operativa durante el rango de búsqueda.
Ratio de tarjetas operativas	REAL	#,## %	Porcentaje de tarjetas operativas durante el rango de búsqueda.
Abonados temporales nuevos	ENTERO	#.###	Nº de abonos temporales expedidos durante el rango de búsqueda.
Abonados temporales activos	ENTERO	#.###	Nº de abonos temporales activos durante el rango de búsqueda.

Tabla A.9: Resultado búsqueda Abonados. Durante el rango de búsqueda.

Perfil - Sexo

Formulario de búsqueda: "Fecha Fin" y "Usuarios".

Resultados:

– Perfil - Sexo al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Usuarios abonados/activos	ENTERO	#.###	Nº de usuarios abonados/activos en el sistema hasta la fecha seleccionada.
Sexo	CADENA	"Masculino", "Femenino" o "Desconocido"	Sexos que reconoce el sistema.
Total	ENTERO	#.###	Total de usuarios de un sexo abonados/activos en el sistema.
%Total	REAL	#,## %	Porcentaje de usuarios de un sexo abonados/activos en el sistema.

Tabla A.10: Resultado búsqueda Perfil - Sexo.

Perfil - Edades

Formulario de búsqueda: "Fecha Fin", "Usuarios", "Edad inicial", "Edad Final" y "Rango de edades".

Resultados:

- Perfil - Edad al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Usuarios abonados/activos	ENTERO	#.###	Nº de usuarios abonados/activos en el sistema hasta la fecha seleccionada.
Edad no especificada	ENTERO	#.###	En caso de no haber especificado una edad.
Edad	RANGO	"[Edad no especificada]" o "[rango definido]"	Rangos de edad que se han definido para dar el resultado de usuarios abonados/activos.
Total	ENTERO	#.###	Total de usuarios de un rango de edad abonados/activos en el sistema.
%Total	REAL	#,##%	Porcentaje de usuarios de un rango de edad abonados/activos en el sistema.

Tabla A.11: Resultado búsqueda Perfil - Edades.

El sistema no hace ningún tipo de comprobación para confirmar que los datos introducidos son aceptables. En caso de introducir en "Rango de Edades" un 0 el sistema se queda colgado y no da respuesta. En caso de que el "Rango de Edades" sea superior a la edad final, mostrará el conjunto completo desde "Edad inicio" hasta "Edad final". En todos los casos muestra una edad calificada como "[Edad no especificada]" con valor 1. El sistema no comprueba que "Edad inicio" sea menor o igual que "Edad final" y da como resultado "[Edad no especificada]" con el valor 1. A la vista del poco control sobre los datos de entrada y las salidas que se producen, datos aparentemente inconsistentes, no parece una tabla que se deba incluir en esta solución.

Perfil - Profesión

Formulario de búsqueda: "Fecha Fin" y "Usuarios".

Resultados:

– Perfil - Profesión al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Usuarios abonados/activos	ENTERO	#.###	Nº de usuarios abonados/activos en el sistema hasta la fecha seleccionada.
Profesión	CADENA	[Profesión]	Profesiones incluidas en el sistema.
Total	ENTERO	#.###	Total de usuarios de una profesión abonados/activos en el sistema.
%Total	REAL	#,##%	Porcentaje de usuarios de una profesión abonados/activos en el sistema.

Tabla A.12: Resultado búsqueda Perfil - Profesión.

Perfil - Postal Code

Formulario de búsqueda: "Fecha Fin" y "Usuarios".

Resultados:

– Perfil - Código postal al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Usuarios abonados/activos	ENTERO	#.###	Nº de usuarios abonados/activos en el sistema hasta la fecha seleccionada.
Código postal	???	???	Códigos postales incluidos en el sistema.
Total	ENTERO	#.###	Total de usuarios de un código postal abonados/activos en el sistema.
%Total	REAL	#,##%	Porcentaje de usuarios de un código postal abonados/activos en el sistema.

Tabla A.13: Resultado búsqueda Perfil - Postal Code.

Existen códigos postales que incluyen números y letras, solamente cadena de caracteres, o desconocidas. Códigos postales que no concuerdan con el estándar

español, por lo que se puede tratar de equivocaciones o códigos de otros paises. Mirando los datos puede que algunos se traten de errores como por ejemplo: 500q5, ZARAGOZA, 50180utebo.

El nombrado de la sección en la interfaz web rompe con el resto de nombrados, ya que se realiza en inglés, mientras que el resto está completamente en español.

A.3. Uso de las estaciones

Usos de las estaciones

Formulario de búsqueda: "Fecha Inicio", "Fecha Fin", "Día", "Tipo Área", "Distrito", "Área" y "Estación". **Resultados:**

- Usos de las estaciones del dd/MM/aaaa al dd/MM/aaaa.

Nombre	Tipo de dato	Formato	Descripción
Estación	CADENA	"ID - Nombre de estación"	Nombre de la estación que incluye su id y como nombre la dirección donde se localiza.
Intervalo de tiempo	???	"Todos los horarios" o hh:00	Intervalos horarios donde se ha utilizado la estación.
Devoluciones Total	ENTERO	#.###	Total de devoluciones de bicis realizadas en la estación dentro del rango temporal.
Devoluciones Media	REAL	#,##	Media de las devoluciones realizadas en la estación dentro del rango temporal.
Retiradas Total	ENTERO	#.###	Total de retiradas de bicis realizadas en la estación dentro del rango temporal.
Retiradas Media	REAL	#,##	Media de retiradas realizadas en la estación dentro del rango temporal.
Neto (D-R)	REAL	#,##	Diferencia entre el número de bicis devueltas y las retiradas.
Total	REAL	#,##	Total de operaciones realizadas en la estación.

Tabla A.14: Resultado búsqueda Usos de las estaciones.

Los filtros de búsqueda no están bien depurados, existen filtros que devuelven lo mismo. En el filtro "Tipo de Área", escoger "Todos" y "Crown" devuelve el mismo resultado. En caso de escoger el atributo "Zone" es imposible seleccionar otros parámetros por lo que la búsqueda da como resultado error por falta de parámetros. Existe la posibilidad de realizar búsquedas por "Cluster", o "SubCluster", que delimita el rango de búsqueda a zonas concretas de la ciudad, definididas por el sistema. Devuelven la misma información que si se realiza la búsqueda con el parámetro "Todos".

Los atributos a seleccionar en el campo "Distrito" son redundante, "Todos" y "Todos los distritos", ya que el resultado de la búsqueda no varia al seleccionar uno u

otro.

En el campo “Área” existen los valores “001” y “Prueba” que no ofrecen ningún tipo de resultado.

Los resultados no varian en información según los filtros de búsqueda, todas las búsquedas ofrecen los campos descritos en la tabla.

Usos de las estaciones por días de la semana

Formulario de búsqueda: “Fecha Inicio”, “Fecha Fin”, “Uso”, “Tipo Área”, “Distrito”, “Área” y “Área” (En el resto de formularios se presenta como “Estación”).

Resultados: Usos de las bicis por día de la semana del dd/MM/aaaa al dd/MM/aaaa.

– Total por días de la semana

Nombre	Tipo de dato	Formato	Descripción
Sin cabecera	CADENA	“Lunes” ··· “Domingo”	Día de la semana a analizar.
Devoluciones	ENTERO	#.###	Devoluciones realizadas en el rango de búsqueda.
Media	REAL	#,##	Media de devoluciones realizadas en el rango de búsqueda.
Recogidas	ENTERO	#.###	Recogidas realizadas en el rango de búsqueda.
Media	REAL	#,##	Media de recogidas realizadas en el rango de búsqueda.
Total	ENTERO	#.###	Total de operaciones realizadas en la estación en el rango de búsqueda.
Media	REAL	#,##	Media del total de operaciones realizadas en la estación en el rango de búsqueda.

Tabla A.15: Resultado búsqueda Usos de las estaciones por días de la semana. Total.

– Media por estación y día de la semana

Nombre	Tipo de dato	Formato	Descripción
Sin cabecera	CADENA	"Lunes" ··· "Domingo"	Día de la semana a analizar.
Devoluciones	REAL	#,##	Media de devoluciones realizadas en el rango de búsqueda.
Media	REAL	#,##	Media de devoluciones realizadas en el rango de búsqueda.
Recogidas	REAL	#,##	Media de recogidas realizadas en el rango de búsqueda.
Media	REAL	#,##	Media de recogidas realizadas en el rango de búsqueda.
Total	REAL	#,##	Media del total de operaciones realizadas en el rango de búsqueda.
Media	REAL	#,##	Media del total de operaciones realizadas en el rango de búsqueda.

Tabla A.16: Resultado búsqueda Usos de las estaciones por días de la semana. Media.

El formulario de búsqueda no sigue el estándar marcado por la interfaz, ya que lo que en el resto de formularios se conoce como "Estación" en este formulario aparece como "Área", lo que conlleva a tener dos campos de búsqueda con el mismo nombre pero con distinto significado y contenido.

Los resultados obtenidos se pueden obtener a partir de la búsqueda "Usos de estaciones", ya que la descarga de ficheros se hace con un rango temporal de un día.

Los resultados de búsqueda siempre tienen el mismo formato, en caso de elegir "Devoluciones" o "Recogidas" rellena con 0's los campos que no necesita.

La segunda tabla es totalmente redundante consigo misma, ya que los campos "Devolución"/"Recogida"/"Total" ya representan la media, por lo que los tres campos "Media" existentes son replicas.

La información contenida en la segunda tabla se podría considerar irrelevante, ya que es derivada de la primera. No es fácil deducir las operaciones realizadas para los resultados obtenidos.

Número de usos en cada estación por intervalo horario

Formulario de búsqueda: "Fecha Inicio", "Fecha Fin", "Tipo Área", "Distrito", "Área" y "Estación".

Resultados:

– Número de usos en cada estación por intervalo horario del dd/MM/aaaa al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Hora	DURACIÓN	hh:00	Intervalos horarios desde las 0:00 hasta las 23:00.
Devoluciones	ENTERO	#.###	Devoluciones realizadas dentro del intervalo horario.
Media	REAL	#,##	Media de devoluciones realizadas.
Retiradas	ENTERO	#.###	Retiradas realizadas dentro del intervalo horario.
Media	REAL	#,##	Media de retiradas realizadas.
Total	ENTERO	#.###	Total de operaciones realizadas dentro del intervalo horario.
Media	REAL	#,##	Media del total de operaciones realizadas.

Tabla A.17: Resultado búsqueda Número de usos en cada estación por intervalo horario

La información obtenida se puede obtener de la tabla "Usos de las estaciones", por lo que no aporta nada nuevo. El nombre de la sección no es correcto teniendo en cuenta el resultado de la búsqueda, ya que no devuelve el resultado para cada estación por separado, sino que devuelve el sumatorio de las operaciones realizadas en cada estación.

Devoluciones fallidas

Formulario de búsqueda: "Fecha Inicio", "Fecha Fin", "Tipo Área", "Distrito", "Área" y "Estación".

Resultados:

- Número de usos en cada estación por intervalo horario del dd/MM/aaaa al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Estación	CADENA	"Nombre de la estación"	Nombre de la estación en la que se produjo el fallo.
Devoluciones fallidas	REAL	#,##	Número de devoluciones fallidas en el rango de búsqueda.
Media	REAL	#,##	Media de devoluciones fallidas.
Porcentaje %	REAL	#,##	Devoluciones fallidas en relación a todos los fallos que se han producido en el servicio.

Tabla A.18: Resultado búsqueda Número de usos en cada estación por intervalo horario

No especifica el motivo de porque la devolución ha fallado.

No cumple con el estándar de nombrado de las estaciones, como en el resto de resultados, no concatena el id de la estación con su nombre. Tratandose de resultados de tipo entero, los muestra como reales.

Las medias no están bien calculadas, por ejemplo: Devolución fallida: 1 — Media: 0.50.

No da el dato de cuantas devoluciones se han hecho a lo largo del día. Al comprobarlo a partir de otra tabla se observa que la media tampoco se deduce a partir del total de devoluciones.

Matriz de movimientos

Formulario de búsqueda: "Fecha Inicio", "Fecha Fin","Hora inicio", "Hora Fin", "Día", "Tipo Área", "Distrito", "Área" y "Estación".

Resultados:

– TOP 20

Nombre	Tipo de dato	Formato	Descripción
Estación de origen	CADENA	"Nombre de la estación"	Nombre de la estación de origen.
Estación de destino	CADENA	"Nombre de la estación"	Nombre de la estación de destino.
Movimientos	ENTERO	#.###	Movimientos que se han realizado en el rango de búsqueda.

Tabla A.19: Resultado búsqueda Matriz de movimiento

Es posible filtrar por intervalos de hora, pero en el resultado final no se refleja el intervalo donde se ha realizado el movimiento, ni si se tiene en cuenta en que hora se coge o deja la bicicleta.

La interfaz solo da opción de elegir la estación de destino.

Vuelve a haber inconsistencia en los resultados con el nombrado de las estaciones, ya que vuelven a no incluir el id de la estación.

Los resultados están acotados a los 20 movimientos más repetidos, en caso de empate no se sabe que estrategia se sigue.

La interfaz no controla que la hora de inicio sea menor o igual que la hora de fin, lo que da como resultado un conjunto vacio. Al igual que no comprueba si el día seleccionado se encuentra dentro del rango de búsqueda.

Si el rango de búsqueda es muy grande el sistema no es capaz de dar un resultado

Informe de ocupación

Formulario de búsqueda: "Fecha Inicio", "Fecha Fin", "Tipo Área", "Distrito", "Área" y "Estación".

Resultados:

– Informe de ocupación del dd/MM/aaaa al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Estación	CADENA	"ID - Nombre de la estación"	Nombre de la estación que incluye su ID.
% tiempo llena	REAL	#,## %	Porcentaje de tiempo que la estación ha estado llena.
Tiempo	DURACIÓN	#h #m	Tiempo total en el que la estación estuvo completa.
Tiempo/Veces llena	DURACIÓN	#h #m	Tiempo medio en el que la estación extuvo completa.
T. Máx. Estación Llena	DURACIÓN	#h #m	Tiempo máximo en el que la estación estuvo completa.
Veces llena	ENTERO	#.###	Número de veces que la estación se encontraba llena en el rango de búsqueda.
Media de ocupación	ENTERO	#.###	Media de slots ocupados en la estación.
Slots	ENTERO	#.###	Puestos de la estación para enganchar las bicicletas.

Tabla A.20: Resultado búsqueda Informe de ocupación

A.4. Nivel del Servicio

Disponibilidad de las bicis

Formulario de búsqueda: "Fecha Inicio", "Fecha Fin", "Bicis contratadas".
Resultados:

– Disponibilidad de las Bicis durante el período del dd/MM/aaaa al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Bicicletas teóricas	NUMÉRICO	#.### ó #,##	Cifra introducida en el campo de búsqueda "Bicis contratadas".
Horas de funcionamiento teóricas	DURACIÓN	hh:00	Horas que las bicicletas están disponibles.
Disponibilidad teórica	REAL	#,##	Disponibilidad calculada a partir del nº de bicicletas teóricas y las horas de funcionamiento teóricas.
Disponibilidad real	REAL	#,##	Disponibilidad real de bicicletas durante el rango de búsqueda.
Ratio de disponibilidad	REAL	#,## %	Porcentaje de disponibilidad calculado a partir de la disponibilidad teórica y la real.

Tabla A.21: Resultado búsqueda Disponibilidad de las bicis.

El resultado de la búsqueda es un resultado teórico en el cual no se tiene en cuenta si los datos introducidos son reales, se puede poner un número negativo o incluso en número real en el campo "Bicis contratadas".

Disponibilidad de las estaciones

Formulario de búsqueda: "Fecha Inicio", "Fecha Fin".

Resultados:

– Disponibilidad de las estaciones durante el período del dd/MM/aaaa al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Media de estaciones disponibles	REAL	#,##	Media de las estaciones disponibles dentro del rango de búsqueda.
Horas de funcionamiento teóricas	DURACIÓN	hh:00	Horas que las estaciones están disponibles.
Disponibilidad teórica	REAL	#,##	Disponibilidad calculada a partir del nº de bicicletas teóricas y las horas de funcionamiento teóricas.
Disponibilidad real	REAL	#,##	Disponibilidad real de estaciones durante el rango de búsqueda.
Ratio de disponibilidad	REAL	#,## %	Porcentaje de disponibilidad calculado a partir de la disponibilidad teórica y la real.

Tabla A.22: Resultado búsqueda Disponibilidad de las estaciones.

El factor de ajuste para calcular la disponibilidad real es variable.

Disponibilidad de los puntos de estacionamiento

Formulario de búsqueda: "Fecha Inicio", "Fecha Fin".

Resultados:

– Disponibilidad de los anclajes durante el período del dd/MM/aaaa al dd/MM/aaaa

Nombre	Tipo de dato	Formato	Descripción
Slots en servicio	REAL	#,##	Slots que han estado en servicio durante el rango de búsqueda.
Horas de funcionamiento teóricas	CADENA	hh:00	Horas que los slots están disponibles.
Disponibilidad teórica	REAL	#,##	Disponibilidad calculada a partir del nº de slots en servicio y las horas de funcionamiento teóricas.
Disponibilidad real	REAL	#,##	Disponibilidad real de slots durante el rango de búsqueda.
Ratio de disponibilidad	REAL	#,##%	Porcentaje de disponibilidad calculado a partir de la disponibilidad teórica y la real.

Tabla A.23: Resultado búsqueda Disponibilidad de los puntos de estacionamiento.

"Slots en servicio" se representa con un real, esto podría indicar que un slot puede haber estado funcionando parte del tiempo y parte no.

El factor de ajuste es variable según la búsqueda.

Anexos B

Modelado BPMN

B.1. Introducción al modelado con BPMN

Incluir explicación de BPMN, por que se ha utilizado, añadir referencias de los elementos usados y explicacion, referenciar a la bibliografia y la la seccion de diseño del sistema 3

Business Process Model and Notation (BPMN), en español Modelo y Notación de Procesos de Negocio, es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo (*workflow*). *BPMN* fue inicialmente desarrollada por la organización *Business Process Management Initiative (BPMI)*, y es actualmente mantenida por el *Object Management Group (OMG)*, después de la fusión de las dos organizaciones en el año 2005.

Su versión actual, es la v2.0.2 publicada en 2013, que contiene una mejora menor sobre la version del 2011 v2.0, respecto a formatos de intercambio.

El principal objetivo de *BPMN* es proporcionar una notación gráfica estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio (*stakeholders*). Entre estos interesados están los analistas de negocio (quienes definen y redefinen los procesos), los desarrolladores técnicos (responsables de implementar los procesos) y los gerentes y administradores del negocio (quienes monitorizan y gestionan los procesos). En síntesis, BPMN tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación, esto facilitará una mejor comprensión de lo que se realiza.

La gestión por procesos se confirma como uno de los mejores sistemas de

organización empresarial para conseguir índices de calidad, productividad y excelencia. En un contexto empresarial y económico tan complejo, globalizado y competitivo como el actual, la gestión de procesos se ha convertido en una necesidad para las empresas, no para tener éxito, sino incluso también para subsistir.

Actualmente hay una amplia variedad de lenguajes, herramientas y metodologías para el modelado de procesos de negocio. La adopción cada vez mayor de la notación BPMN como estándar, ayudará a unificar la expresión de conceptos básicos de procesos de negocio (por ejemplo: procesos públicos y privados, orquestación, coreografía, etcétera) así como conceptos avanzados de modelado (por ejemplo: manejo de excepciones, compensación de transacciones, entre otros).

B.2. Uso de BPMN en el proyecto

El estándar *BPMN* ha sido utilizado en este proyecto durante la fase de diseño del proyecto (Capítulo 3), ya que se ha entendido que es la manera más clara de representar el ciclo de vida de los procesos y las fases por las que pasa cada uno de ellos de una manera que cualquier lector pueda comprender de que se trata sin realizar un gran esfuerzo para comprenderlo.

Los elementos *BPMN* que se han utilizado son los siguientes:

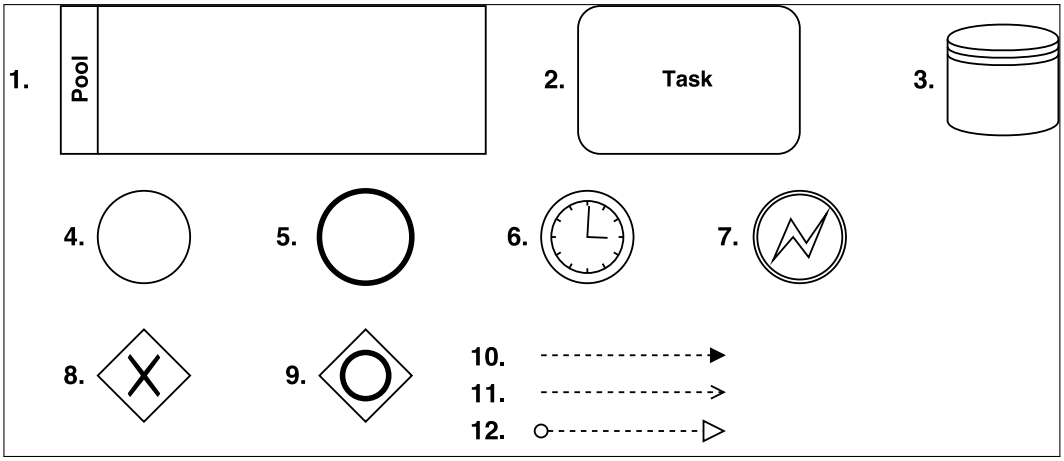

Figura B.1: Elementos BPMN usados en el proyecto

1. ***Pool*:** Representa los participantes principales de un proceso, por lo general, separados por las diferentes organizaciones.

2. ***Task*:** Representa una sola unidad de trabajo que no es o no se puede dividir a

un mayor nivel de detalle de procesos de negocio sin diagramación de los pasos de un procedimiento.

3. ***Data Store*:** Representa un almacenamiento de datos, como puede ser una base de datos.

4. ***General start*:** Actúa como un disparador de un proceso.

5. ***General End*:** Indica el final de un proceso.

6. ***Timer Standard*:** Actúa como un disparador de un proceso al que se le asigna un componente temporal, para indicar cuando debe ser disparado.

7. ***Error Boundary Interrupting*:** Utilizado para representar que se ha producido un error en un proceso.

8. ***Exclusive Gateway*:** También conocida como *XOR Gateway*. Se utiliza para modelar la bifurcación de un proceso en base a una decisión.

9. ***Inclusive Gateway*:** Se utiliza para indicar para crear una bifurcación de caminos que pueden continuar su ejecución de forma paralela.

10. ***Sequence Flow*:** Muestra el orden en que las actividades se llevarán a cabo. El flujo de secuencia puede tener un símbolo al inicio, un pequeño diamante indica uno de un número de flujos condicionales desde una actividad, mientras que una barra diagonal indica el flujo por defecto desde una decisión o actividad con flujos condicionales.

11. ***Association*:** Se suele usar para conectar artefactos o un texto a un objeto de flujo y puede indicar muchas direccionabilidades usando una punta de flecha no rellena (hacia el artefacto para representar a un resultado, desde el artefacto para representar una entrada, y los dos para indicar que se lee y se actualiza).

12. ***Message Flow*:** Está representado por una línea discontinua con un círculo no relleno al inicio y una punta de flecha no rellena al final. Esto nos dice, que el flujo de mensaje atraviesa la frontera organizativa (por ejemplo, entre *Pools*).

Anexos C

Ficheros gestionados en el proyecto

En este anexo se quiere mostrar el formato y contenido de los ficheros que participan en el proceso *ETL* (Sección 4.2.3). Tanto los ficheros obtenidos desde la web legada, que son tratados para extraer la información y generar a partir de ellos ficheros *CSV*, como el fichero de configuración utilizado para extraer información que puede ser modificada por el administrador de la infraestructura sin necesidad de realizar modificaciones a bajo nivel.

C.1. Ficheros gestionados en el proceso ETL

En primer lugar, los ficheros descargados desde la plataforma web legada tienen formato *XLS*, o formato de *Hoja de Excel*. Estos ficheros están diseñados para la lectura directa por un humano, por lo que son díficil de tratar automáticamente para obtener la información que contienen.

bizi 13/11/2017

Usos de las estaciones del 12/11/2017 al 12/11/2017

Estación	Intervalo de tiempo	Devoluciones		Retiradas		Neto (D-R)	Total
		Total	Media	Total	Media		
1 - Avda. Pablo Ruiz Picasso - Torre del Agua	Todos los horarios	13	13,00	14	14,00	-1,00	27,00
	8:00	1	1,00	0	0,00	1,00	1,00
	9:00	0	0,00	1	1,00	-1,00	1,00
	10:00	2	2,00	0	0,00	2,00	2,00
	11:00	3	3,00	6	6,00	-3,00	9,00
	12:00	2	2,00	0	0,00	2,00	2,00
	13:00	2	2,00	0	0,00	2,00	2,00
	14:00	0	0,00	3	3,00	-3,00	3,00
	15:00	0	0,00	1	1,00	-1,00	1,00
	16:00	1	1,00	0	0,00	1,00	1,00
	17:00	1	1,00	0	0,00	1,00	1,00
	18:00	1	1,00	2	2,00	-1,00	3,00
	19:00	0	0,00	1	1,00	-1,00	1,00
2 - Avda. Ranillas - Avda. Pablo Ruiz Picasso	Todos los horarios	24	24,00	24	24,00	0,00	48,00

Figura C.1: Ejemplo del contenido de los ficheros descargados.

Como se puede observar en el pequeño fragmento del documento (Figura C.1) se dispone de la información de las estaciones desglosadas según la franja horaria donde se ha realizado cada operación, indicadas en la parte superior del documento.

Este fichero es tratado por un proceso automátizado que extrae la información a partir de la localización de las celdas del documento y a partir de lo extraido se genera un fichero *CSV* que contiene toda la información relevante de este documento.

idEstacion	nombreEstacion	fechaDeUso	intervaloDeTiempo	devolucionTotal	devolucionMedia	retiradasTotal	retiradasMedia	neto	total
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	Todos los horarios	10	10.00	10	10.00	0.00	20.00
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	0:00	1	1.00	0	0.00	1.00	1.00
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	11:00	1	1.00	2	2.00	-1.00	3.00
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	13:00	1	1.00	2	2.00	-1.00	3.00
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	14:00	2	2.00	1	1.00	1.00	3.00
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	16:00	0	0.00	1	1.00	-1.00	1.00
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	18:00	1	1.00	0	0.00	1.00	1.00
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	19:00	2	2.00	1	1.00	1.00	3.00
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	20:00	2	2.00	2	2.00	0.00	4.00
58	Avda. Valle de Zuriza - Avda. Cataluña	2017-11-12	22:00	0	0.00	1	1.00	-1.00	1.00
78	Avda. San Juan de la Peña - C/ Somport	2017-11-12	Todos los horarios	11	11.00	8	8.00	3.00	19.00
78	Avda. San Juan de la Peña - C/ Somport	2017-11-12	0:00	0	0.00	1	1.00	-1.00	1.00
78	Avda. San Juan de la Peña - C/ Somport	2017-11-12	9:00	1	1.00	0	0.00	1.00	1.00
78	Avda. San Juan de la Peña - C/ Somport	2017-11-12	10:00	1	1.00	2	2.00	-1.00	3.00
78	Avda. San Juan de la Peña - C/ Somport	2017-11-12	11:00	0	0.00	1	1.00	-1.00	1.00

Figura C.2: Ejemplo del contenido de los ficheros tratados.

En la Figura C.2 se puede ver parte del contenido del fichero una vez tratado, se ha utilizado un visor de *CSV* para poder ver los campos con mayor comodidad. Este fichero ahora es más sencillo de manipular y de cargar su contenido dentro en la base de datos, ya que Apache Hive dispone de la posibilidad de hacer cargas masivas de datos a partir de ficheros *CSV*.

C.2. Fichero de configuración interno

Para la puesta en marcha del sistema es necesario disponer de un fichero de configuración donde se recogen los diferentes parametros de configuración que pueden ser modificados por el administrador a la hora de poner en funcionamiento el proyecto. El fichero *myConf.json* (*Listing* 5) da la posibilidad de realizar estas configuraciones de manera externa al sistema por lo que es facilmente manipulable y facilita que el sistema pueda ser desplegado en diferentes sistemas.

```
{
   "fileLocation": {
       "chromeDriverLocation" : "/TFG/herramientas/chromedriver",
       "downloadLocation" : "/descargas",
       "csvDirectory" : "/csvFiles",
       "dockerSharedDirectory":"/docker-hive/compartida"
   },
   "legacySystem": {
       "baseURL" : "http://reportingportal-zar.clearchannel.com/",
       "user" : "",
       "password" : ""
   },
   "connectionHiveDB":{
       "driverName" : "org.apache.hive.jdbc.HiveDriver",
       "user" : "",
       "password" : "",
       "jdbc" : "jdbc:hive2://",
       "host" : "localhost",
       "port" : "10000",
       "database" : "default"
   },
   "connectionMysqlDB":{
       "driverName" : "com.mysql.jdbc.Driver",
       "user" : "",
       "password" : "",
       "jdbc" : "jdbc:mysql://",
       "host" : "localhost",
       "port" : "3306",
       "database" : "tareas"
   }
}
```

Listado 5: Contenido del fichero myConf.json

File location

En este elemento *JSON* (*Listing* 6) se define donde se encuentra el ejecutable *chromeDriver* (chromeDriverLocation), necesario para la ejecución de las tareas de descarga automática del sistema web legado. Se definen las carpetas en el sistema de ficheros donde se guardaran los ficheros descargados (*downloadLocation*) y tratatados (csvDirectory). Por último se indica donde se encuentra la carpeta que se utiliza como enlace entre el sistema local y el sistema desplegado en *Docker* (dockerSharedDirectory).

```
{
   "fileLocation": {
       "chromeDriverLocation" : "/TFG/herramientas/chromedriver",
       "downloadLocation" : "/descargas",
       "csvDirectory" : "/csvFiles",
       "dockerSharedDirectory":"/docker-hive/compartida"
   }
}
```

Listado 6: myConf.json: fileLocation

Legacy System

En este elemento *JSON* (*Listing* 7) se define la url del sistema web legado del que se van a realizar las descargas automatizadas y las credenciales necesarias para el acceso.

```
{
   "legacySystem": {
       "baseURL" : "http://reportingportal-zar.clearchannel.com/",
       "user" : "",
       "password" : ""
   }
}
```

Listado 7: myConf.json: legacySystem

Connection Hive DB y Connection MySQL DB

En estos elementos *JSON* (*Listing* 8 y *Listing* 9) se define lo necesario para poder generar una conexión a partir de un conector *JDBC*. Se define el driver necesario para

realizar la conexión (*driverName*), las credenciales necesarias para la conexión (*user, password*), los elementos necesarios para generar la cadene *JDBC* para la conexión (*jdbc,host,port* y *database*).

```
{
   "connectionHiveDB":{
       "driverName" : "org.apache.hive.jdbc.HiveDriver",
       "user" : "",
       "password" : "",
       "jdbc" : "jdbc:hive2://",
       "host" : "localhost",
       "port" : "10000",
       "database" : "default"
   }
}
```

Listado 8: myConf.json: connectionHiveDB

```
{
   "connectionMysqlDB":{
       "driverName" : "com.mysql.jdbc.Driver",
       "user" : "",
       "password" : "",
       "jdbc" : "jdbc:mysql://",
       "host" : "localhost",
       "port" : "3306",
       "database" : "tareas"
   }
}
```

Listado 9: myConf.json: connectionMysqlDB

Anexos D

API del sistema

La Transferencia de Estado Representacional (en inglés *Representational State Transfer*) o *REST* es un estilo de arquitectura software para sistemas hipermedia distribuidos como la *World Wide Web* [10]. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

En este proyecto se ha utilizado *API REST* para el intercambio de datos entre cliente y servidor, haciendo posible realizar peticiones de información que el cliente web recibe como objetos *JSON* y puede procesar facilmente. Para documentar las operaciones realizadas con *API REST* se ha utilizado *Swagger*, que por medio de documentación de código es capaz de generar una interfaz en la que se puede ver y probar cada uno de los *endpoints*.

La aplicación cuenta con los siguientes recursos a los que puede realizar peticiones:

- **Cuentas:** Disponible para realizar peticiones referentes a las cuentas de usuario (crear nueva cuenta, obtener información sobre las cuentas, modificar contraseñas, etc...).
- **Usuario:** Disponible para realizar operaciones referentes a los usuarios del sistema (Crear nuevos usuarios, obtener información de los usuarios disponibles, borrar usuarios, etc...).
- **Controlador de usuarios JWT:** Recurso disponible para realizar la autenticación de usuarios.
- **Descarga:** Disponible para peticiones referente a la información de las tareas gestionadas por el proceso *ETL* de extracción de datos.

- **Tratamiento:** Disponible para peticiones referente a la información de las tareas gestionadas por el proceso *ETL* de tratamiento de datos.
- **Insercion:** Disponible para peticiones referente a la información de las tareas gestionadas por el proceso *ETL* de inserción de datos.
- **Usoestacion:** Disponible para realizar peticiones sobre la información disponible acerca del servicio *Bizi Zaragoza*.

A continuación se adjuntan un ejemplo de uno de los *endpoints* (Figura D.1) documentado con *Swagger*, siguiendo el estándar *OpenAPI*.

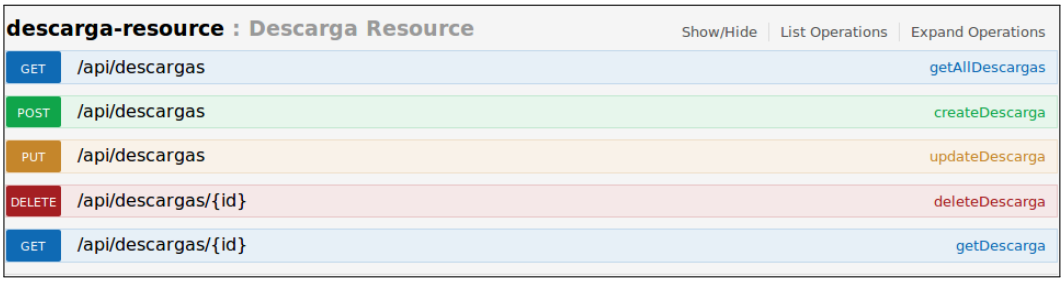

Figura D.1: Documentación del *endpoint descarga-resource*.

La operacion ***GET*** dispone de dos variantes, es posible realizar la petición de toda la información (Figura D.2) o realizar peticiones de un recurso en concreto a partir de su identificador (Figura D.2).

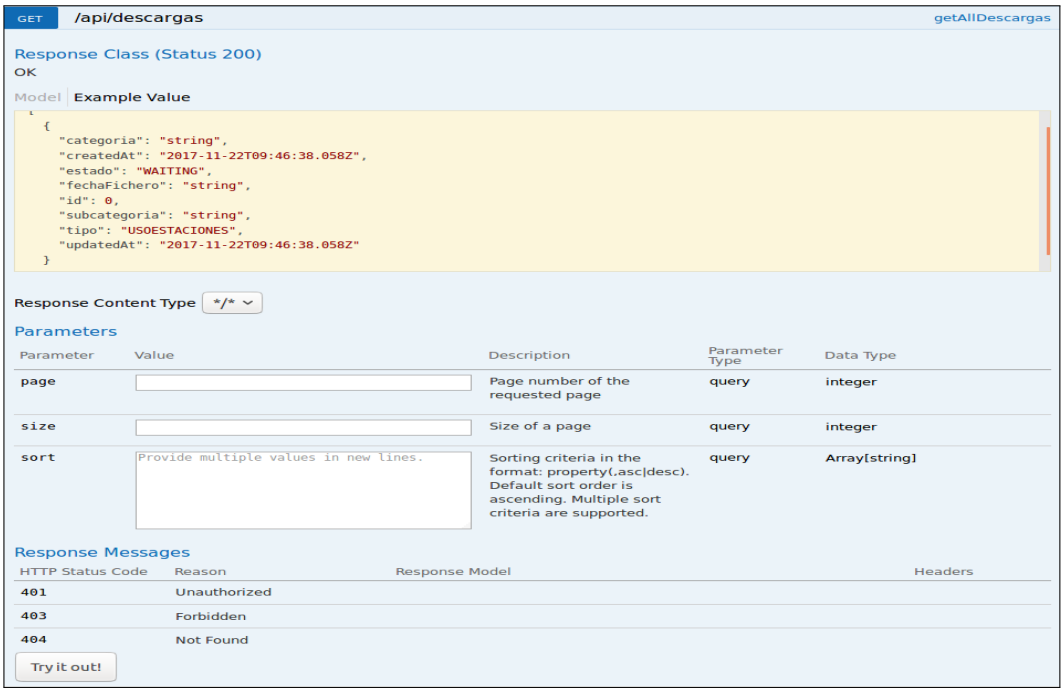

Figura D.2: Operación GET para la obtención de todos los recursos disponibles.

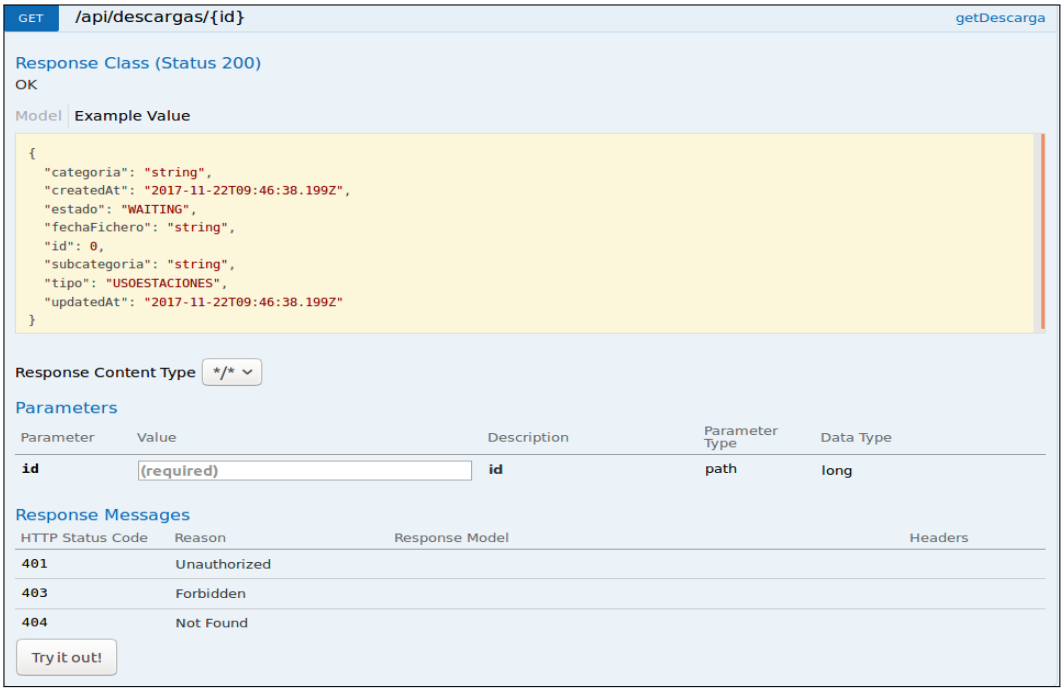

Figura D.3: Operación GET para la obtención de un recurso por medio de su identificador.

La operación ***POST*** (Figura D.4) permite la creación de nuevos recursos. Esta operación ha sido incluida aunque no está disponible en la solución final del proyecto, ya que, en este caso, no se cree conveniente que se puedan insertar datos manualmente.

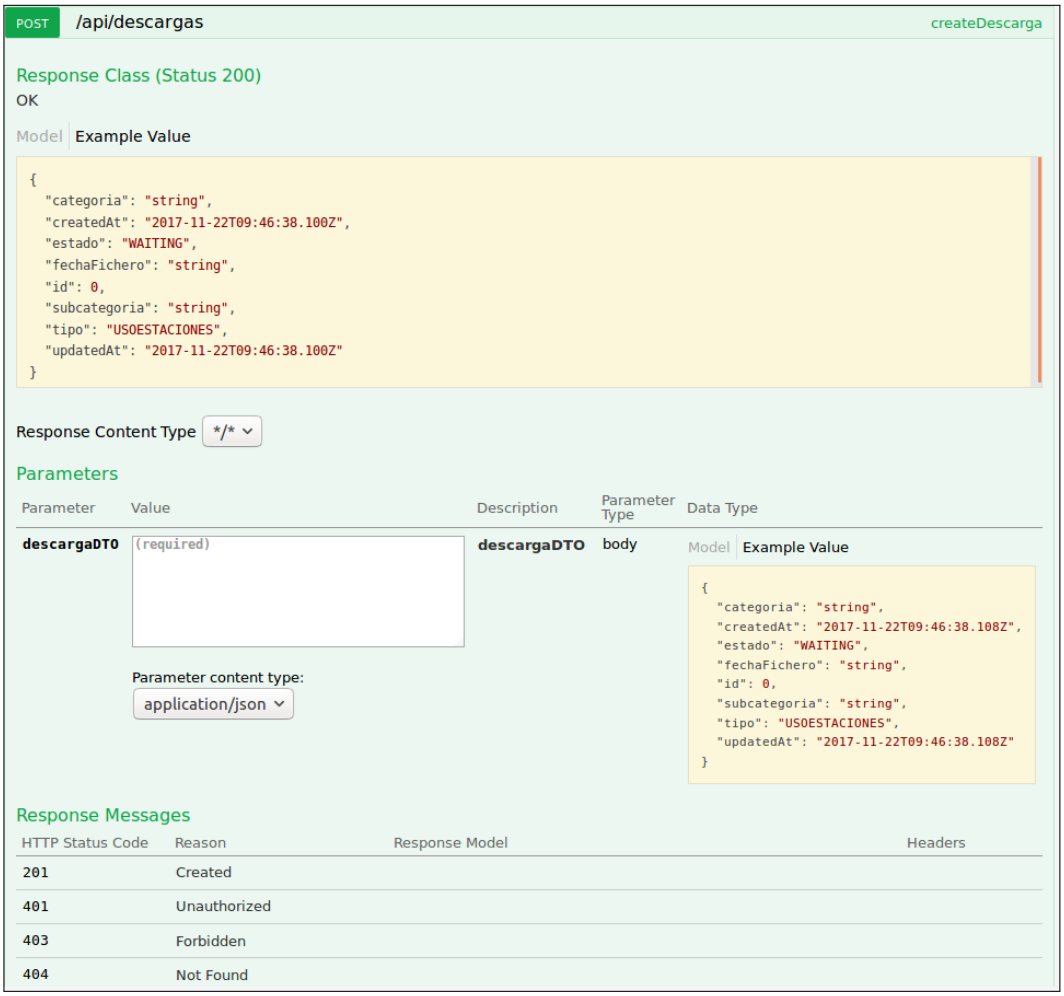

Figura D.4: Operación POST para la creación de nuevos recursos.

La operación ***PUT*** (Figura D.5) ofrece la posibilidad de modificar la información de un recurso. En la aplicación final se ha restringido que la única modificación posible sea el estado del recurso.

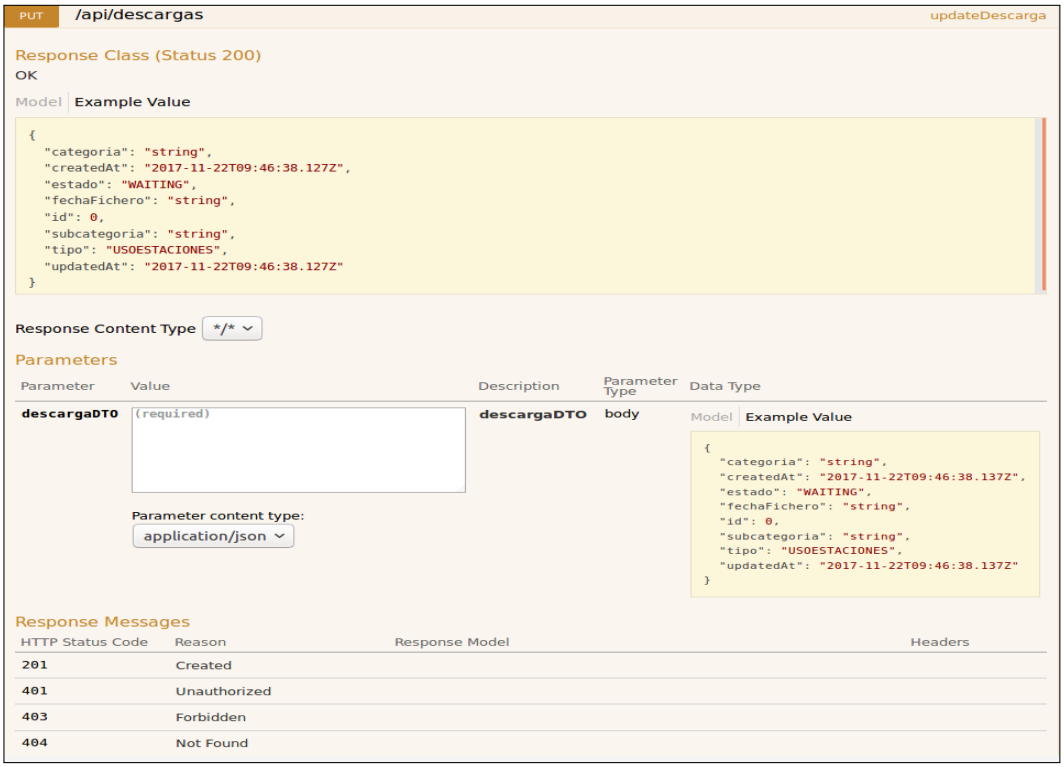

Figura D.5: Operación PUT para la modificación de un recurso.

La operación ***DELETE*** (Figura D.6) permite eliminar, a partir de su identificador, un recurso. Aunque se ha implementado esta operación, en la solución final no está disponible ya que no se contempla que se puedan realizar borrado de recursos de las tareas.

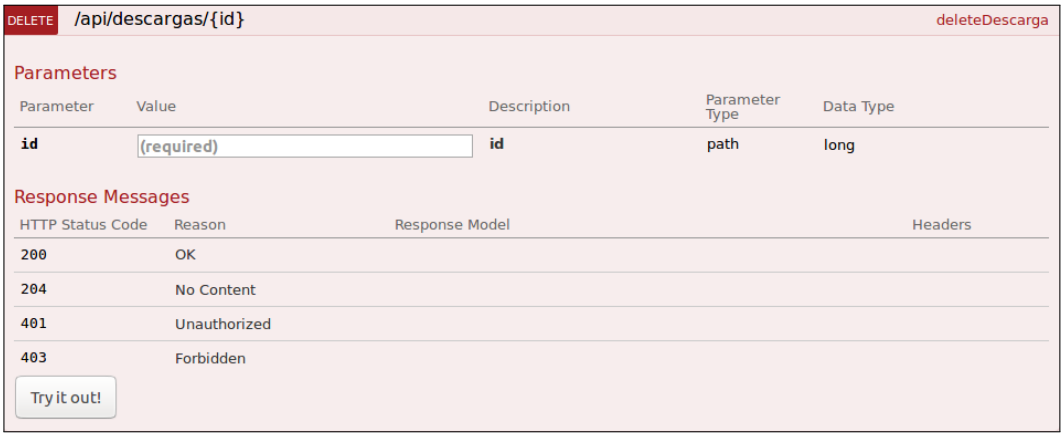

Figura D.6: Operación DELETE para la eliminación de un recurso.

Anexos E

Cálculo del coste de la hora de trabajo

Para entender como se ha generado el presupuesto presentado en la Sección 5.4, se presenta como se ha llevado a cabo el cálculo del coste que debe tener la hora de trabajo. Para ayudar a la comprensión de las etapas se añade el ejemplo realizado para este proyecto. En la Tabla E.1 se muestran los valores con los cuales se ha llegado a la obtención del coste calculado.

Concepto	**Valor**
Jornada laboral	8 horas/día
Días a la semana	5 días
Salario mensual	1.200,00 €
Días de vacaciones	21 días
Días festivos	14 días
Días previstos de incapacidad	7 días
% de tiempo administrativo	50 %
Gastos mensuales	620,00 €
% de beneficio	20 %

Tabla E.1: Datos para el cálculo del precio por horas.

1. **Calcular el salario bruto anual:**

   Salario mensual × 12 meses

   1,200.00 × 12 = 14,400.00

2. **Hallar el precio básico por hora:**

   Salario anual ÷ 2,080 horas posibles (8 horas × 5 días × 52 semanas)

   14,400.00 ÷ 2,080 = 6.92

3. **Sumar el precio de las horas que no se trabajan:**

   14 días festivos × 8 hs. = 112 hs. +

   21 días de vacaciones × 8 = 168 hs. +

   7 días de incapacidad × 8 hs. = 56 hs.

   336 hs. × 6.92 de coste básico = 2,326.15

4. **Calcular el valor del tiempo que se dedica a presupuestos, reuniones, venta, formación... (tiempo administrativo):**

   50 % (2,080 hs. posibles - 336 horas que no trabajarás) × 6.92 coste básico.

   50 % × 1,744 hs. × 6.92 = 6,036.92

5. **Calcular el total de los gastos fijos:**

   Alquiler mensual: 400.00 +

   Servicios mensuales: 10.00 +

   Otros gastos fijos mensuales: 210.00

   Gastos fijos mensuales 620.00 × 12 meses = 7,440.00 fijos anuales

6. **Sumar el valor de las horas que no se trabajan más el valor del tiempo de administración y el de los gastos fijos para obtener el precio extra anual por tu trabajo:**

   2,326.15 + 6,036.92 + 7,440.00 = 15,803.08 de precio extra anual

7. **Calculamos lo que se ganará al año:**

   2,080 hs. posibles al año - 336 hs. de vacaciones, festivos e incapacidad - 872 hs. de reuniones y presupuestos × 6.92 coste básico

   872 horas de trabajo × 6.92 = 6,036.92 de beneficio anual.

8. **Calcular el porcentaje de rentabilidad dividiendo el coste entre el beneficio:**

   15,803.08 ÷ 6,036.92 = 261.774

9. **Por último, para calcular la hora de trabajo se suma el porcentaje de rentabilidad y el porcentaje de beneficio deseado al precio básico:**

   6.92 + 6.92 × 261.774 % + 6.92 × 20 % = **26.43 por hora de trabajo.**