



GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO FINAL DE GRADO

---

**Diseño de una arquitectura para la integración  
de información semi-estructurada proveniente  
de páginas web y su mejora con contenido  
semántico.**

---

*Autor:*  
Andrea CALIA

*Supervisor:*  
Ana SANCHIS HUERTAS  
*Tutor académico:*  
Oscar BELMONTE FERNÁNDEZ

Fecha de lectura: 30 de Junio de 2014  
Curso académico 2013/2014

## **Resumen**

En este documento se presenta una propuesta de arquitectura Software para el desarrollo de aplicaciones Web de extracción, transformación y distribución de información a partir de páginas Web. En el proceso de transformación de los datos, éstos se amplían añadiendo contenido semántico (meta-datos). La información ampliada se distribuye en formato XML. La arquitectura propuesta se compone de módulos para la realización de las tareas necesarias.

Con fin demostrativo, se han creado dos aplicaciones Web, una para servir la información de los trenes de la Comunidad Valenciana y la otra para proporcionar información acerca del servicio de préstamo de bicis en Castellón Bicicas. Finalmente, se ha desarrollado un tercer proyecto para la integración de los dos anteriores en una aplicación Web para la planificación de rutas de transporte intermodales.

## **Palabras clave**

Microformatos, REST, Web Service, Semántica, XML, Arquitectura Software, Web Wrapping, Geolocalización

## **Keywords**

Microformats, REST, Web Service, Semantics, XML, Software Architecture, Web Wrapping, Geolocalization

# Índice general

<b>1. Introducción</b>	<b>5</b>
<b>2. Agradecimientos</b>	<b>7</b>
<b>3. Arquitectura del sistema</b>	<b>9</b>
3.1. Justificación . . . . .	9
3.2. Requisitos y alcance . . . . .	12
3.3. Diseño conceptual . . . . .	13
<b>4. Caso de uso: Intermodal Trip Planner</b>	<b>19</b>
4.1. Metodología . . . . .	19
4.1.1. Definición de roles . . . . .	21
4.1.2. Sprints . . . . .	21
4.1.3. Reuniones . . . . .	21
4.1.4. Pizarra Kanban . . . . .	22
4.1.5. Historias de usuario/Test de aceptación . . . . .	23
4.2. Requisitos . . . . .	23
4.2.1. Requisitos funcionales . . . . .	24
4.2.2. Requisitos no funcionales . . . . .	25
4.3. Análisis . . . . .	26

4.3.1. Renfe Web Service . . . . .	27
4.3.2. Bicicas Web Service . . . . .	30
4.3.3. Intermodal Trip Planner . . . . .	33
4.4. Tecnologías usadas . . . . .	36
4.5. Desarrollo . . . . .	42
4.5.1. Scheduled Executor Framework . . . . .	42
4.5.2. Sprints . . . . .	43
<b>5. Conclusiones</b>	<b>53</b>
<b>A. Respuestas de los servicios Web</b>	<b>57</b>
A.1. Renfe Web Service . . . . .	57
A.2. Bicicas Web Service . . . . .	58
<b>B. Tiempo de respuesta de la aplicación</b>	<b>63</b>

# Capítulo 1

## Introducción

El uso de Internet ha crecido mucho en la última década. Gracias a los dispositivos móviles como Smartphone y Tablet, es posible acceder a los contenidos de la Web desde cualquier sitio y momento. Consecuentemente, el contenido generado por los usuarios de Internet ha crecido ocupando la mayor parte del tráfico de la Web. Un ejemplo de este tipo de contenido son las entradas de los Blogs de las personas, los Tweets generados en la plataforma Twitter o los comentarios en la red social Facebook. Este contenidos es la base de la Web 2.0 [3].

La información generada por los usuarios de Internet abarca muchos temas. Entre ellos se puede encontrar comentarios, opiniones o reseñas sobre cualquier tema de interés general. Uno de estos temas es la ciudad donde los usuarios residen. Por ejemplo, en la Web se puede encontrar información sobre los restaurantes de una ciudad o se pueden encontrar consejos sobre que transporte es mejor usar para llegar de un sitio a otro. Esta información entra en un área de investigación muy activa últimamente, Smart Cities.

Smart Cities es un concepto muy amplio. Uno de cuyos objetivos es ofrecer a los usuarios todo tipo de información y/o recomendación sobre las ciudades. Además, se incluye en esta campo de investigación la búsqueda de soluciones para los problemas de las grandes ciudades como el parking o el tráfico de coches. Smart Cities tiene muchos otros objetivos y ramas de investigación, pero todas convergen en el intento de crear ciudades eficientes enérgicamente y económicamente.

La mayor parte de la información disponible en Internet es accesible a sus usuarios, pero su interpretación es muy difícil para agentes Software. Este es un obstáculo muy grande para el análisis de la información.

El proyecto que se presenta en este documento consiste en desarrollar una propuesta de arquitectura Software para añadir contenido semántico a información semi-estructurada en la web. Esta arquitectura permite recopilar información desde recursos en la web, ofrecer estos datos en formato semi-estructurado y, finalmente, añadir estructuras semánticas para el procesamiento de estos datos por parte de agentes Software. Los recursos para la recopilación de información pueden ser páginas web (blogs o páginas institucionales por ejemplo), datos en formato XML o, más en general, cualquier fuente de datos semi-estructurada. Además, si la información proviene

de páginas web, el sistema la ofrece en formato XML para el fácil procesamiento por parte de programas informáticos. Por último, a la información generada en formato XML se le añade una capa de información semántica basada en varios Microformatos[2] publicadas en la web. Esta capa permite a agentes semánticos, o Software, procesar estos datos para interactuar con otros servicios de forma automática o semi-automática.

El hecho de poder proporcionar información existente en la web con formato semi-estructurado y con estructuras semánticas, es la fundación del concepto de Web 3.0. Existen varias definiciones de Web 3.0 y todavía no se ha definido un estándar para dicho concepto. Como se indica en [8], en la Web 3.0 agentes semánticos y humanos pueden utilizar la gran cantidad de datos presentes en la web de hoy en día.

Además de proporcionar una arquitectura abstracta, se ha desarrollado un caso de uso para demostrar la utilidad de dicha arquitectura. Este caso de uso permite enlazar varios servicios disponibles en la ciudad de Castellón. Estos servicios son el transporte público de trenes y el servicio de préstamo de bicis Bicicas.

El caso de uso integra la información extraída de las páginas Web de estos dos servicios para proporcionar al usuario una manera de llegar a la estación de trenes de Castellón usando el servicio de préstamo Bicicas. Esta aplicación Web, localiza el usuario y el sitio de Bicias más cercano, traza la ruta más corta hacia la estación de trenes y muestra al usuario los trenes Castellón-Valencia que podrá coger.

La empresa donde se ha desarrollado el proyecto es Ubik Geospatial Solutions <sup>1</sup>. La empresa ofrece servicios informáticos a clientes particulares y a grandes organizaciones. Los servicios incluyen páginas web y aplicaciones para Smartphone y Tablets en el campo de Smart Cities, Environment y Social Media Integration. La empresa es una spin-off universitaria, de la Universitat Jaume I, recién creada y con sede en el edificio ESPAITEC 2.

Con el permiso de la empresa de estancia en prácticas, se ha decidido ofrecer el código completo del proyecto. Para ello, se ha creado un repositorio público en el servicio GitHub. La dirección es <https://github.com/andreacalia/uji-final-year-project>

Con el objetivo de que la comprensión del caso de uso especificado se ha realizado un vídeo demostrativo de la aplicación Intermodal Trip Planner. Se puede acceder a él en la siguiente dirección: <http://youtu.be/HdGypS9rvb8>

---

<sup>1</sup><http://www.ubikgs.com/>

## Capítulo 2

# Agradecimientos

La realización de este proyecto no hubiera sido posible sin la ayuda de mi familia. El constante apoyo que me han dado ha permitido seguir trabajando de la mejor forma posible. Además, quiero agradecer el soporte que he recibido por el tutor académico y el supervisor de la empresa de la estancia en prácticas. Finalmente, quiero agradecer el banco Santander por haber financiado con la beca *Santander CRUE CEPYME 2014* mi estancia en prácticas.





## Capítulo 3

# Arquitectura del sistema

En este capítulo se va a especificar una propuesta de arquitectura Software para la creación de aplicaciones Web que permiten añadir contenido semántico a información semi-estructurada presente en la Web. El principal objetivo de esta arquitectura es definir una forma estándar para efectuar operaciones de Web Wrapping sobre recursos Web, para transformar los datos en formato semi-estructurado y para añadir información semántica.

La sección 3.1 describe los motivos que han llevado a la creación de la arquitectura. La sección 3.2 describe los requisitos que la arquitectura tiene que satisfacer así como sus objetivos y características. La sección 3.3 explica el diseño conceptual de la propuesta de arquitectura. En ella, se describen todas las partes o módulos que la componen y se explica el flujo de datos entre los varios módulos.

### 3.1. Justificación

Las páginas presentes hoy en día en la Web se crean principalmente para la visualización de la información por parte de seres humanos. El lenguaje usado para la creación de dichas páginas web es el HTML o XHTML.

Actualmente, se está desarrollando la versión 5 el lenguaje HTML [15]. El estado de esta versión no es definitivo y queda todavía mucho trabajo por hacer. Sin embargo, su adopción por parte de los mayores navegadores web ya ha empezado y se pueden utilizar varias funcionalidades de la nueva versión del lenguaje. La quinta iteración del lenguaje HTML incluye muchas características muy novedosas y que permiten sobrepasar los límites de su versión anterior. Las características más destacadas son:

- *Web Workers*: permite a las aplicaciones web ejecutar script muy pesados en segundo plano. Esto permite aumentar la reactividad de las páginas web separando la computación más intensiva de la gestión de la interfaz gráfica.
- *Video*: HTML 5 permite reproducir contenido audio-visual directamente en la página web.

Con las versiones anteriores del lenguaje, esta característica se dejaba en mano a plug-in y complementos externos al navegador y desarrollados por terceros.

- *Canvas*: la nueva versión del lenguaje propone un API para la definición de directivas para el dibujo de formas e imágenes en las páginas Web. Esto permite embeber en los sitios Web contenido tanto 2D como 3D. Además, se puede aprovechar la tecnología WebGL [6] para la creación de contenido 3D complejo.
- *Geolocation*: la geolocalización de los dispositivos conectados a Internet es una de las características más demandada hoy en día. Se pueden crear aplicaciones personalizadas basándose a la localización del usuario por ejemplo. Esta funcionalidad está integrada en HTML 5.
- *Semántica*: la nueva iteración del lenguaje HTML permite describir mejor el contenido de las páginas web. Ahora se pueden usar etiquetas como *footer*, *article*, *etc...* para que los navegadores o agentes Software puedan entender mejor la información que contienen las etiquetas HTML.

Estas características, junto con todas las otras novedades, permiten a los desarrolladores de páginas Web crear aplicaciones cada vez más complejas y pueden encontrar nuevas formas de interacción con el usuario.

La información de las páginas Web se interpreta con un Software para leer el contenido y estructura para poder presentar los datos al usuario final según el estilo especificado por el creador del sitio. Sin embargo, esta visualización está pensada para ser entendida por un ser humano. Un agente Software no es capaz de entender de forma natural el significado de la información. Para este problema no hay una sola solución, pero se han propuesto varios métodos para poder rodear el obstáculo. El sistema que más se usa actualmente es la inclusión en los sitios Web de meta-datos para dar más información sobre el texto y los datos presentes en las etiquetas HTML. Un Software puede trabajar con los datos de una página web usando esta información para poder seleccionar las partes de la página que sean de interés. Hay varias tecnologías para incluir meta-datos en páginas HTML. A continuación se comentan las principales:

- *RDFa*[14]: es una recomendación de la organización W3C para añadir contenido semántico a las páginas XHTML. El modelo de datos que se especifica permite incluir estructuras RDF [11] (sujeto, predicado y objeto) en el mismo código XHTML a través de los atributos de las etiquetas. Siendo RDF un formato de datos basado en XML, la página que incluya RDFa tiene que respetar el estándar XHTML. Este lenguaje presenta las mismas características de HTML con la restricción de que los documentos tienen que ser documentos XML bien formados. Un documento XML bien formado es aquello que respeta las reglas impuestas en la sección 2.1 *Well-Formed XML Documents* de [12].
- *Microformatos*[2]: los microformatos permiten extender la expresividad de el lenguaje HTML a través el uso de atributos de las etiquetas. Existen muchas especificaciones para estos meta-datos, las más populares son *h-event*, para incluir información sobre eventos, *h-geo*, para especificar coordenadas geográficas en formato de latitud y longitud y *h-card*, para publicar información sobre personas, u organizaciones. Una de las características más importante de los microformatos es que no están sujetos a las reglas XML ya que se integran en los atributos que especifica el estándar HTML.

Los meta-datos hacen que el contenido de las páginas Web pueda ser procesado fácilmente por agentes Software hechos a medida. La extracción de la información se puede hacer usando varias técnicas. La más usada es el Web Wrapping. Esta técnica extrae información del DOM (Document Object Model) de los documentos HTML. A partir de las etiquetas extraídas, se puede acceder al contenido textual y de los atributos para extraerlo. En el caso de que la página contenga meta-datos, la selección de la información relevante puede ser simple o incluso trivial. Sin embargo, si no existen dichos meta-datos, un profundo análisis de la estructura de la página Web es necesario para poder formular consultas adecuadas. Estas consultas se pueden escribir en el lenguaje XPath [9]. Con XPath se pueden expresar trayectorias del árbol DOM de las páginas Web para apuntar a las etiquetas HTML que se desea.

Una vez extraída la información requerida, esta se puede redistribuir en un formato que sea más accesible a los programas. El formato que representa un estándar para este tipo de tareas es XML. Los documentos XML generados se pueden publicar con servicios Web.

Los servicios Web se pueden crear a partir de varios esquemas o estilos arquitectónicos muy diferentes. Uno de los más comunes hoy en día es REST (Representational State Transfer) [5]. En este tipo de arquitectura, una URI (Uniform Resource Identifier) representa un sujeto o un recurso del servicio Web. Además, las operaciones que se pueden hacer sobre los recursos en el servicio Web se especifican con el método HTTP con el cual se efectúa la petición. Las características más importantes de la arquitectura REST son:

- *Client-Server*: la arquitectura REST separa los roles de los clientes y del servidor en su modelo.
- *Stateless*: REST no mantiene ninguna sesión del usuario guardada en el servidor. Esto es debido a que, en cada petición, los datos que representan al cliente se tienen que incluir en la URI.
- *Addressable*: todo los recursos que proporciona el Servidor se puede acceder a través de una URI.
- *Uniform Interface*: los recursos del Servidor son accesibles con los métodos del estándar HTTP.
- *Representational Oriented*: el mismo recurso se puede representar con formatos diferentes. Por ejemplo, se puede devolver información de un API REST en formato JSON o XML.
- *Hypermedia*: varios recursos se pueden enlazar entre ellos. Esto es debido a que cada uno esta representado por una URI.

Leonard Richardson[7] ha propuesto una jerarquía para indicar cuanto una aplicación Web respeta los principios REST. Los niveles de la jerarquía son:

- *Level 0*: se proporciona un solo recurso accesible a través del método POST de HTTP.
- *Level 1*: muchos recursos pueden ser accedido con el método POST.
- *Level 2*: se puede acceder a muchos recursos con diferentes métodos HTTP.

- *Level 3*: se usan los enlaces Web para interactuar con el servicio. El cliente descubre nuevas URI a partir de la información devuelta por peticiones anteriores. Este nivel se denomina *Hypermedia As The Engine Of Application State (HATEOAS)*.

El proceso de desarrollo de un sistema de extracción y publicación de información en la Web puede llegar a ser muy caótico. Esto es debido, principalmente, a que dicho sistema se puede implementar de muchas formas diferentes e involucrar muchas tecnologías. Esto conlleva problemas a la hora de mantener el código del sistema así como reduce la eficacia de una actualización futura.

Por estos motivos, se pretende desarrollar una arquitectura que pueda resolver el problema de la extracción de información a partir de páginas web, de manipulación de esta información y de publicación de forma estándar. Esto permite solventar o aliviar los problemas presentados anteriormente.

### 3.2. Requisitos y alcance

Los requisitos generales que la arquitectura tiene que solventar son proporcionados por la empresa donde se ha hecho la estancia en prácticas. El Cuadro 3.1 resume los requisitos recopilados que la arquitectura tiene que incluir.

Número	Descripción requisito
1	La arquitectura tiene que permitir hacer el scraping de diferentes recursos Web, sean ellos en formato XML o HTML
2	Se ha de poder envolver la información extraída para que sea disponible en formato XML
3	Opcionalmente, se tiene que poder guardar la información extraída de forma persistente
4	Se pueden obtener los datos datos envueltos con o sin contenido semántico añadido
5	La interfaz de publicación de los datos extraídos tiene que ser REST

Cuadro 3.1: Cuadro con los requisitos de la arquitectura del sistema

El primer requisito permite dar flexibilidad a la arquitectura. Los datos sobre los cuales el sistema puede actuar pueden estar en formato XML o HTML. Esto es debido a que la arquitectura tiene que poder extraer información de páginas Web como objetivo principal. Dado que las páginas Web han de estar escritas en lenguaje HTML o XHTML, es posible extender el tipo de datos de entrada a cualquier documento XML, siendo HTML y XML derivados de SGML (Standard Generalized Markup Language). El segundo requisito es de fundamental importancia dado que el objetivo final es poder proporcionar la información extraída como servicio Web. XML es el formato de datos más adecuado para este tipo de tareas y es muy fácil incluir información semántica en él. El tercer requisito es opcional y su aplicación depende del tipo de datos que se tienen que extraer. Su propósito es guardar los datos extraídos por el proceso de Web Wrapping

de forma persistente en una base de datos por ejemplo. A este nivel de abstracción, la forma en la cual se guardan los datos no se especifica, siendo ella muy dependiente de los datos extraídos. Este requisito permite diseñar un sistema eficiente cuando los datos de la fuente cambian muy poco o el servicio proporcionado tiene un uso muy frecuente. En el primer caso, si la fuente de los datos no cambia muy a menudo o si cambia sistemáticamente cada cierto tiempo, es una buena idea usar persistencia para no volver a hacer el proceso de Web Wrapping e integración de los datos por cada petición, siendo estos datos iguales para muchas peticiones seguidas. En el segundo caso, si el servicio proporcionado por la arquitectura tienen un tráfico muy alto, el modulo de persistencia puede actuar como *cache* de los datos, para no sobrecargar el sistema y proporcionar un servicio mejor. Estas especificaciones han de considerarse guías, la decisión de como y si implementar la persistencia depende del caso concreto de aplicación. El cuarto requisito especifica que el sistema tiene que devolver los datos procesados y en formato XML con contenido semántico o no. La potencia del sistema es que permite añadir semántica a los datos extraídos con la técnica de Web Wrapping, pero en ocasiones estos meta-datos no son necesarios. En estos casos, se tienen que poder pedir los datos a la aplicación Web sin información semántica para así aumentar el rendimiento del sistema y disminuir la carga de la red. El quinto y último requisito especifica que la aplicación Web resultado de la implementación del sistema tiene que usar el estilo arquitectónico REST. Este estilo permite mejorar el estilo de programación y los clientes del servicio tienen un conjunto de URI claro y conciso.

El alcance de la arquitectura es desde la comunicación y extracción de los datos de la fuente de datos (recurso externo) hasta la publicación del servicio para proporcionar los datos al usuario final.

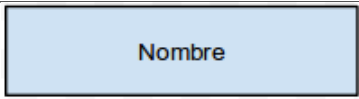
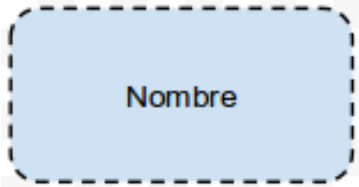


### 3.3. Diseño conceptual

En esta sección, se describe el diseño conceptual de la arquitectura del sistema. El diseño, cumple todos los requisitos especificados en la Sección 3.2. El desarrollo del diseño conceptual del sistema ha sido la primera etapa de la estancia en prácticas. Para ello, se han realizado muchas reuniones con el tutor académico y con el supervisor de la empresa. El objetivo de las reuniones ha sido asegurarse de que el esquema propuesto cumpliera con las expectativas de las partes interesadas manteniendo un nivel de complejidad razonable. El sistema se ha dividido en módulos que encapsulan una funcionalidad específica de la arquitectura. Entre los módulos hay conexiones que muestran el flujo de los datos. Para expresar el esquema de la arquitectura se ha usado un diagrama creado para la ocasión. El Cuadro 3.2 describe todos los componentes del diagrama para que su lectura sea de fácil comprensión.

La Figura 3.1 muestra el esquema de la arquitectura que se ha diseñado. Este esquema incluye todos los componentes necesarios para cumplimentar con los requisitos especificados en la Sección 3.2.

A continuación se explica de forma detallada cada modulo de la arquitectura.

- *External Web Resource*: este componente no es parte de la arquitectura, pero se incluye en el diseño conceptual para representar el recurso Web que representa la fuente de datos para la implementación del sistema. El recurso Web puede ser por ejemplo una página Web

Representación visual	Descripción
	Este componente representa un módulo de la arquitectura. El módulo es obligatorio y se tiene que implementar porque su funcionalidad es un requisito.
	Este componente representa un módulo opcional del sistema. Este tipo de módulos proporcionan mayor funcionalidad y flexibilidad.
	Las flechas conectan dos módulos siguiendo el flujo de los datos. Cada módulo tiene que estar conectado a otro mediante una flecha. No pueden haber módulos aislados del resto del sistema.
	Este tipo de flecha representa un flujo de datos opcional entre componentes. En este caso, por lo menos un componente de los extremos de la flecha tiene que ser opcional.

Cuadro 3.2: Cuadro que describe las partes del diagrama de la arquitectura

(<https://www.google.com>) o un servicio Web (<http://nominatim.openstreetmap.org/reverse?format=xml&lat=52.5487429714954&lon=-1.81602098644987&zoom=18&addressdetails=1>).

- *Web Wrapping*: este componente es de fundamental importancia y encapsula toda la funcionalidad necesaria para hacer el proceso de Web Wrapping de la fuente de datos. En él se ha de obtener el recurso Web haciendo las peticiones necesarias e interpretar la respuesta. Además, incluye los algoritmos necesarios para extraer los datos de interés a partir de la fuente de datos recuperada. A este nivel de abstracción no se especifica la forma para llevar a cabo esta tarea. Esto es debido a que la implementación concreta es muy dependiente de las características de la fuente de datos.
- *XML Wrapper*: el objetivo de este componente es de recubrir la información extraída de la fuente de datos con etiquetas XML. En este módulo se tiene que definir la estructura del documento XML así como su documento de validación, DTD o XSchema, si se desea.
- *Persistence Module*: el módulo de persistencia es un componente opcional de la arquitectura. Se puede incluir o no incluir en la implementación dependiendo del tipo de datos que se manejan y de la frecuencia de acceso al sistema. Este módulo permite almacenar los datos del módulo *XML Wrapper*. Dependiendo del caso de uso, se puede almacenar el documento XML generado por el componente *XML Wrapper* o los datos extraídos de la fuente de datos.
- *REST Observable Service*: este componente es opcional y permite ampliar mucho la flexibilidad del sistema. Su objetivo es proporcionar una interfaz de programación para que

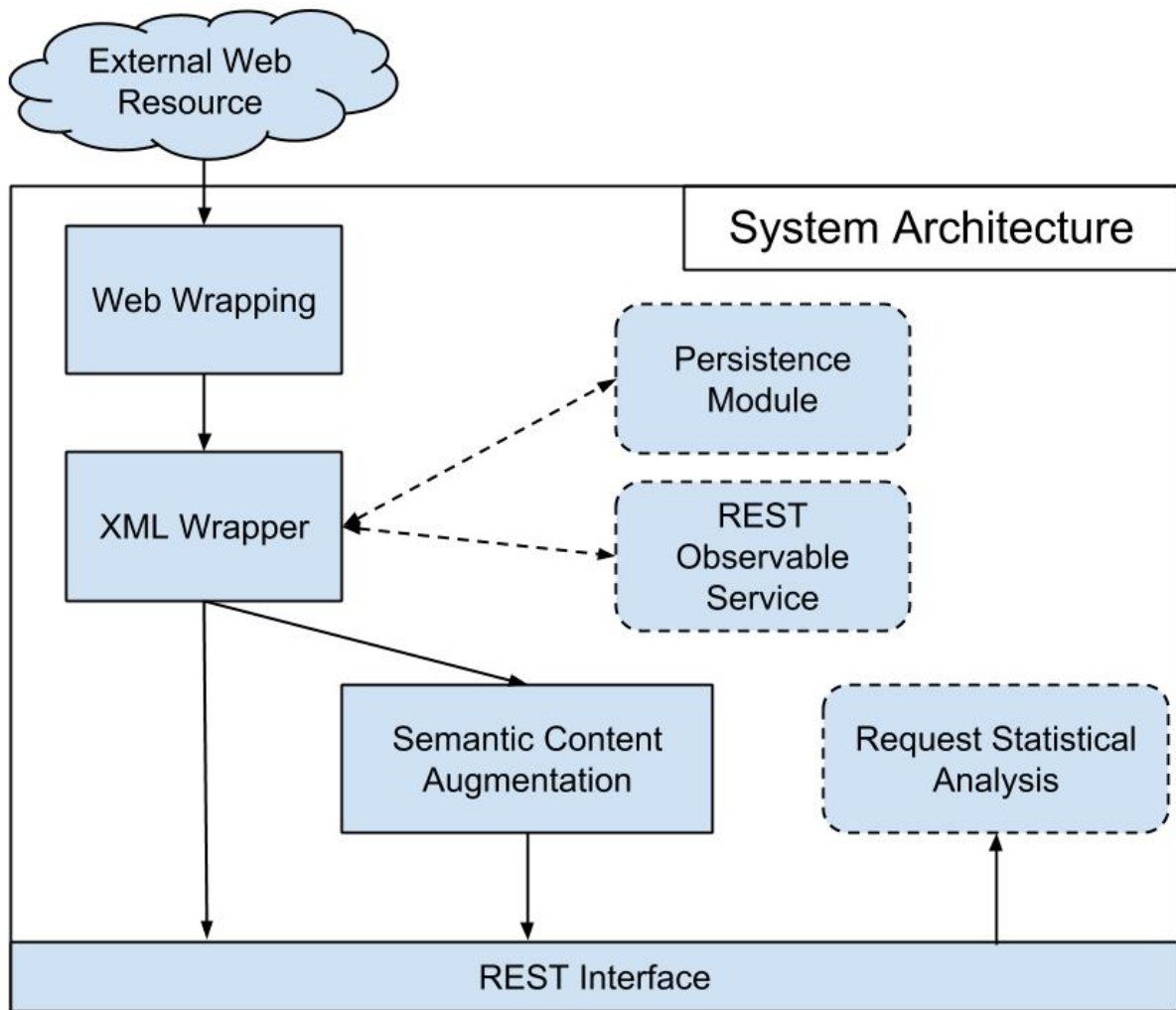


Figura 3.1: Esquema de la arquitectura del sistema

clientes remotos se puedan suscribir a los cambios que se pueden producir en los datos proporcionados por el sistema. Esto permite a clientes remotos (seas ellos otros servidores o aplicaciones de gestión) ser notificados de los cambios y ahorra el coste en términos de computación y ancho de banda de realizar peticiones para detectar cambios en los datos. Este mecanismo aplica la idea del patrón de diseño Observer/Observable a los servicios Web. Una aplicación que puede ser beneficiada por este tipo de componente es una aplicación Web que contiene un Dashboard (o cuadro de comando) para monitorizar el estado del sistema. Es particularmente útil para este caso de uso porque los datos que se visualizan tienen que estar actualizados en tiempo real. Nuevamente, no se especifican las tecnologías para la implementación de este componente. La tecnología SSE (Server-Sent Events) [13] se puede usar para llevar a cabo esta tarea. Los SSE son eventos que el cliente recibe del servidor. Este tipo de comunicación se llama notificaciones *push*. Esta tecnología permite tener una latencia muy baja, debido a una conexión que se establece y permanece activa entre el servidor y el cliente.

- *Semantic Content Augmentation*: este es el componente de la arquitectura que añade la capa de información semántica al documento XML generado a partir de la fuente de

datos. Este componente es obligatorio para cumplir con los objetivos del sistema. El tipo de contenido semántico no se especifica, porque es dependiente de los datos que contiene la fuente de datos del caso de uso particular. Además, el tipo de meta-datos que se añaden también depende del caso de uso. Unos posibles meta-datos que se pueden añadir a un documento XML son Microformatos o meta-datos RDFa. Siendo el documento a ampliar un documento XML, la tecnología XSLT [10] permite llevar a cabo esta tarea de una forma muy eficiente.

- *Request Statistical Analysis*: este componente es opcional y permite almacenar información acerca de las peticiones que el sistema recibe. El propósito de este modulo es el análisis de las peticiones que el sistema recibe para detectar picos de demanda y mejorar el servicio proporcionado.
- *REST Interface*: un requisito de la arquitectura es que la interfaz Web implemente el estilo arquitectónico REST. Este componente se encarga de recibir y tratar las peticiones que recibe el sistema y procesar las respuestas.

En la Figura 3.2 se muestra el flujo de la información que se produce entre los módulos obligatorios de la arquitectura propuesta. Este diagrama ayuda a comprender el funcionamiento y las operaciones que cada módulo implementa.

Se ha usado un diagrama de secuencia porque permite mostrar la interacción entre los módulos a lo largo del tiempo. En este tipo de diagramas el tiempo se desarrolla hacia abajo.



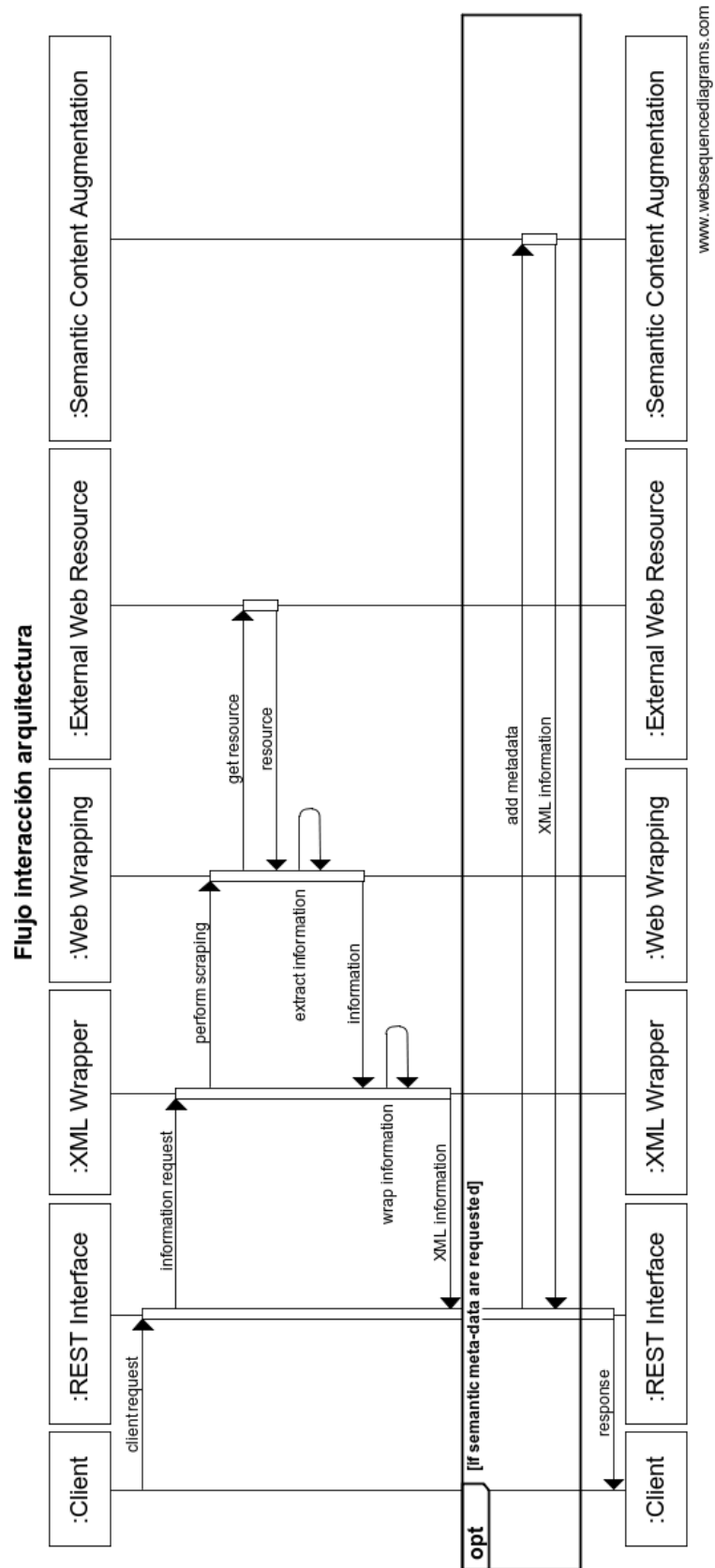


Figura 3.2: Diagrama de secuencia con el flujo de información entre los módulos de la arquitectura propuesta



## Capítulo 4

# Caso de uso: Intermodal Trip Planner

Este capítulo describe el caso de uso que se ha planteado para demostrar el funcionamiento de la arquitectura propuesta en el capítulo 3.

El caso de uso elegido se llama *Intermodal Trip Planner*. Se ha decidido implementar la arquitectura para proporcionar un servicio Web que permite devolver información de la páginas Web de Renfe y Bicicas. Por un lado, de la página de Renfe, se proporciona información acerca de los horarios de los trenes de cercanías de la Comunidad Valenciana. Por otro lado, los datos extraídos de la Web de Bicicas permiten disponer de información de la disponibilidad en tiempo real de las bicis del servicio de préstamo de Castellón. Con estos servicios Web, se pretende desarrollar una aplicación Web que, localizando el usuario, permite encontrar el puesto de Bicicas con bicis disponibles más cercano y calcular la ruta entre la posición del usuario hasta la posición del sitio de Bicicas. Luego, calcular la ruta entre la estación de Bicicas hasta la estación de trenes de Castellón y dar un listado de los trenes que se pueden coger para llegar a Valencia teniendo en cuenta el tiempo del viaje hasta la estación. Además, la aplicación tiene que permitir añadir al calendario de Google del usuario un evento que representa el viaje en tren. La aplicación final se acerca al concepto de *Mash Up*, realizando la integración de información de varias fuentes en una aplicación Web.

Este caso de uso permite demostrar la funcionalidad de la arquitectura propuesta, implementando tanto los componentes obligatorios como algunos opcionales.

### 4.1. Metodología

En esta sección se describe la metodología de desarrollo que se ha utilizado para la implementación de la aplicación de la estancia en prácticas. La aplicación tiene unas características tales que una metodología ágil de desarrollo permite aumentar la eficiencia y la eficacia del desarrollo [1].

A continuación se presentan los doce principios que están a la base de las metodologías ágiles:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para la continuación, ajustar y perfeccionar su comportamiento en consecuencia.

Las motivaciones que han llevado a la decisión de usar una metodología de desarrollo ágil son:

- *Cercanía partes interesadas*: las partes interesadas en el desarrollo del caso de uso son: el estudiante, el supervisor de la empresa y el tutor académico..
- *Equipo de desarrollo pequeño*: el equipo de desarrollo donde se ha incorporado el estudiante en prácticas es pequeño. Esto hace que no sea necesaria la coordinación de muchas personas.

Teniendo en cuenta las motivaciones de arriba, se ha considerado que la metodología de desarrollo más adecuada es una adaptación de la metodología SCRUM. SCRUM es una metodología ágil que permite la creación incremental de un proyecto Software. Además, permite hacer frente a cambios en los requerimientos de forma rápida y eficiente. Esta metodología se ha adaptado a las características del proyecto usando las partes que más valor añadido podían generar. Estas son: definición de roles, Sprints, reuniones y test de aceptación. Además, se ha usado una pizarra Kanban para la organización de las tareas

A continuación se describen en detalle todas estas características.

#### 4.1.1. Definición de roles

La definición de varios roles en el equipo que participa en el proyecto permite aumentar la eficiencia y minimizar el esfuerzo en llevar a cabo el proyecto. En la metodología SCRUM se definen tres tipos de roles: Product Owner, Scrum Master y Equipo de desarrollo. A continuación se muestra la asignación de estos roles:

- *Product Owner*: este rol se ha reservado para el supervisor de la empresa.
- *Scrum Master*: este rol se ha asignado al tutor de la universidad.
- *Equipo de desarrollo*: el equipo de desarrollo en el cual participa el alumno de estancia en prácticas.

El propósito del rol de Product Owner es asegurar que los intereses de los *stakeholders* (partes interesadas) se cumplen en todo momento. La persona que asume este rol tiene que vigilar la evolución del proyecto y que se cumplan los plazos y las expectativas del cliente. El Scrum Master asegura la correcta implementación de la metodología de trabajo. No permite que factores externos impidan cumplir con los objetivos de los Sprints. El equipo de desarrollo lleva a cabo el desarrollo de la aplicación y se asegura de entregar a tiempo los objetivos de cada Sprint.

#### 4.1.2. Sprints

En la metodología ágil SCRUM un Sprint representa una iteración del proceso de desarrollo del proyecto. Es la unidad básica de tiempo e incluye unos objetivos que, una vez cumplimentados todos, permiten la entrega de una versión más o menos funcional de la aplicación.

Los Sprints del proyecto tienen una duración de 15 días cada uno. Al final de cada Sprint, el equipo se reúne para presentar los progresos en el desarrollo y para comprobar que se cumplen los objetivos propuestos.

#### 4.1.3. Reuniones

Las reuniones son una parte fundamental en una metodología ágil. Existen dos tipos de reuniones: reuniones diarias (*Daily Meeting*) y reuniones de cada Sprint (*Sprint Planning Meeting*),

Las reuniones diarias se realizan a primera hora de la mañana cada día. Permiten que todos los miembros del equipo estén al tanto del estado del proyecto y ayudan a centrarse en los objetivos del proyecto y no dispersar esfuerzos.

El segundo tipo de reuniones se realiza al final de cada Sprint. En estas reuniones, cada 15 días, se presenta el trabajo realizado por el equipo de desarrollo y se evalúa el resultado obtenido.

Esto permite que las partes interesadas en el proyecto tomen nota de los avances y comprobar que el proyecto sigue con la plan inicial.

#### 4.1.4. Pizarra Kanban

La pizarra Kanban (o Kanban Board) es una herramienta muy útil en las metodologías ágiles. No es parte de ninguna metodología en concreto, pero sí puede complementar las tareas de organización.

Una pizarra Kanban contiene columnas que representa un estado de una tarea en particular. Cada ítem que se incluye en una columna representa una tarea. Cada tarea puede tener uno o más responsables. Las tareas pasan de columna en columna según el estado en el que se encuentran. En el caso más simple, la pizarra Kanban tiene tres columnas: *To Do*, *In Progress* y *Done*. Idealmente, una tarea empieza su ciclo de vida en el estado *To Do*, pasa a *In Progress* cuando el responsable de la tarea empieza su desarrollo. Finalmente acaba en estado *Done* cuando su desarrollo ha sido terminado.

Normalmente, se tiene una pizarra Kanban por cada Sprint del proyecto. Así pues, las tareas que se encuentran en la pizarra se refieren al mismo Sprint y no distorsionan la visualización de tareas en Sprint futuros.

Las pizarras de Kanban presentan muchas ventajas. Entre ellas, se puede destacar que se puede visualizar el flujo de las tareas en un vistazo y esto permite tener una visión global de las tareas que quedan pendientes en cada Sprint. Además, cada miembro del equipo sabe en todo momento lo que los otros miembros están haciendo, esto permite monitorizar la carga de trabajo del equipo y adaptarse a situaciones imprevistas, como retrasos en una tarea.

Para integrar la pizarra de Kanban en los proyectos existen muchas herramientas online. La que se ha decidido usar para el desarrollo de la aplicación ha sido Trello <sup>1</sup>.

La aplicación Web Trello permite crear una pizarra Kanban permitiendo añadir tareas y columnas según sea necesario. Además, almacena todos los cambios hechos en la nube y es posible revisarlos en cualquier momento. Finalmente, permite compartir pizarras virtuales con otros miembros del equipo, de forma que todos puedan ver y actualizar las tareas. La Figura 4.1 muestra una pizarra Kanban en la aplicación Web Trello.

---

<sup>1</sup><https://trello.com/>

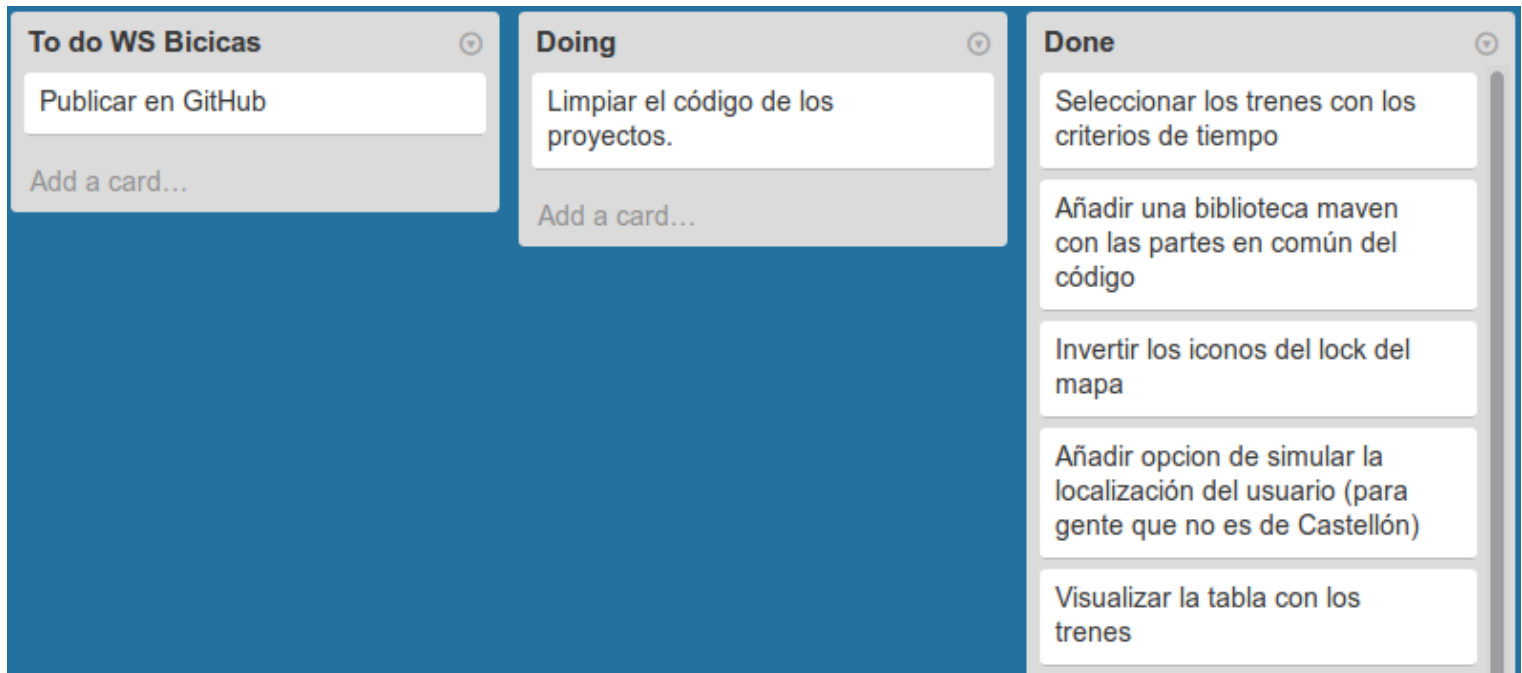


Figura 4.1: Ejemplo de pizarra Kanban en Trello

#### 4.1.5. Historias de usuario/Test de aceptación

Las historias de usuario son una herramienta muy útil a la hora de recopilar requisitos funcionales para el desarrollo de un proyecto Software. En ellas, se describe una funcionalidad que el Software tiene que tener en su versión final.

Esta herramienta permite al equipo de desarrollo tener muy claros los requisitos y las expectativas del cliente.

Los tests de aceptación se crean con el cliente a partir de las historias de usuario. Los tests se generan para que se pueda comprobar cada historia de usuario en cuanto se implemente en el sistema.

## 4.2. Requisitos

En esta sección se describen los requisitos de la aplicación que representa el caso de uso de la arquitectura propuesta. Los requisitos especificados pueden ser de dos tipos: funcionales y no funcionales. Por un lado, los requisitos funcionales corresponden con una funcionalidad que el usuario tiene que percibir en la aplicación final. Por ejemplo, la presencia de uno o varios botones para realizar unas tareas. Por otro lado, los requisitos no funcionales son características que tiene que tener el sistema y que el usuario no percibe directamente. Un ejemplo puede ser el uso de una tecnología en particular u obtener un cierto tiempo de respuesta de la aplicación.

#### 4.2.1. Requisitos funcionales

A continuación se especifican los requisitos funcionales que se han recopilado para el caso de uso. Estos se han expresado con la técnica de las historias de usuario. El Cuadro 4.1 lista los requisitos. Cada requisito se identifica con un código.

Identificador	Historia de usuario
US-1	Como usuario quiero que el sistema me localize para saber mi posición.
US-2	Como usuario quiero que mi posición se muestre en un mapa.
US-3	Como usuario quiero que el sistema me encuentre el puesto de Biciclas más cercano para poder sacar una bici.
US-4	Como usuario quiero que el puesto de Biciclas sugerido por el sistema tenga bicis disponibles para poder sacar una.
US-5	Como usuario quiero que el sistema me indique el camino más corto entre mi posición y la posición del puesto de Biciclas más cercano para poder dirigirme hacia él.
US-6	Como usuario quiero que el sistema me lleve desde el puesto de Biciclas más cercano al puesto de Biciclas de la estación de Renfe para poder coger un tren.
US-7	Como usuario quiero que el sistema calcule el tiempo estimado de llegada a la estación de Renfe.
US-8	Como usuario quiero que el sistema me muestre los trenes Castellón-Valencia que puedo coger teniendo en cuenta el tiempo estimado de llegada a la estación. Esto es para poder elegir el tren a coger.
US-9	Como usuario quiero que el sistema me permita añadir un evento a mi calendario de Google para poder visualizarlo en mi móvil.
US-10	Como analista de datos, quiero que el sistema guarde el estado de los puestos de Biciclas cada cierto intervalo de tiempo. Esto es para poder analizar los datos en un futuro.
US-11	Como analista quiero poder cambiar el intervalo de actualización del estado de los puestos de Biciclas para poder mejorar el análisis.

Cuadro 4.1: Cuadro con las historias de usuario recopiladas

A partir de las historias de usuario recopiladas, se han creado los tests de aceptación que permiten verificar la funcionalidad requerida. El Cuadro 4.2 presenta los tests de aceptación desarrollados.

La aplicación final se considerará acabada cuando todos los tests de aceptación que se han establecido pasarán la prueba con las partes interesadas en el proyecto.



Identificador	Test de Aceptación
TA-1	Localizar el usuario y comprobar la localización.
TA-2	Mostrar un icono en el mapa en la posición del usuario.
TA-3	Comprobar que el puesto de Bicicas más cercano al edificio Espatec es el de la facultad de Humanas.
TA-4-1	Comprobar que el sitio de Bicicas seleccionado tenga bicis disponibles.
TA-4-2	Comprobar que se selecciona el segundo sitio de Bicicas más cercano si en el primero no quedan muy pocas bicis comparado con las que quedan en el segundo.
TA-5	Comprobar que el camino desde la posición del usuario hacia el sitio de Bicicas sea el más corto caminado.
TA-6	Comprobar que el camino desde el sitio de Bicicas hacia la estación de trenes sea el más corto en bici.
TA-7	Comprobar que el tiempo estimado para llegar a la estación de trenes sea correcto.
TA-8	Comprobar que el sistema muestre los trenes que se pueden coger desde la llegada estimada a la estación Renfe.
TA-9	Comprobar que el sistema añada al calendario de Google del usuario un evento con el viaje en tren seleccionado.
TA-10	Comprobar que se guarde correctamente el estado de los puestos de Bicicas.
TA-11	Comprobar funciona correctamente el cambio de periodo de actualización de los sitios de Bicicas.

Cuadro 4.2: Cuadro con las historias de usuario recopiladas

#### 4.2.2. Requisitos no funcionales

Para la realización del caso de uso, se han impuesto unos requisitos no funcionales. Estos requisitos no añaden una funcionalidad al sistema y el usuario de la aplicación no es necesario que los conozca.

Los requisitos no funcionales que el caso de uso tiene que respetar se presentan a continuación:

- *Información semántica*: los servicios Web que se crearán para cumplir los objetivos del caso de uso, tienen que integrar información semántica usando el componente dedicado de la arquitectura propuesta en el Capítulo 3. Los meta-datos semánticos tienen que ser Microformatos y tienen que integrarse en los servicios proporcionados por el caso de uso. El Anexo A muestra la información devuelta por los servicios Web implementados. En ella se pueden notar los meta-datos semánticos incluidos.
- *Tiempo de respuesta*: la aplicación Web final tiene que tener un tiempo de respuesta inferior a 4 segundos. Este tiempo incluye todas las peticiones Web que se tienen que efectuar, así como todas las operaciones necesarias para recopilar los datos que se usan. Este tiempo de respuesta se tiene que medir en la máquina donde se desplegará la aplicación final. En el Anexo B se detallan los tests que se han llevado a cabo sobre los servicios Web

desarrollados.

El siguiente listado muestra los requisitos no funcionales relativos a la tecnología a utilizar en el proyecto:

- *Java EE*<sup>2</sup>: para la realización de los servicios y aplicaciones propuestas se usará el lenguaje de programación orientado a objetos Java. La versión es Enterprise Edition (EE), que incluye los componentes necesarios para poder desarrollar aplicaciones Web.
- *Git*<sup>3</sup>: Git permite la gestión de las versiones de los ficheros de código fuente del proyecto.
- *Maven*<sup>4</sup>: para poder organizar la gestión de las dependencias del proyecto, se usará la herramienta Maven. Ésta permite el control tanto de las dependencias como de el sistema de compilación y despliegue de la aplicación.
- *Jersey*<sup>5</sup>: el framework para la creación de los servicios REST será Jersey.
- *Repositorio compartido*: el código fuente del proyecto tiene que usar el control de versiones Git. Git es un sistema distribuido y una copia del código se tiene que guardar en un repositorio remoto. Todos los miembros del proyecto tienen que tener acceso al repositorio remoto.
- *OpenStreetMap*<sup>6</sup>: el servicio que se usará para la visualización de mapas será OpenStreetMap.

### 4.3. Análisis

En esta sección se procede al análisis de los requisitos funcionales y no funcionales adquiridos en la Sección 4.2.

A partir del Cuadro 3.1, se han detectado tres áreas de desarrollo: extracción de la información acerca del servicio de trasporte en tren de Renfe, información del estado de los sitios de Biciclas y la realización de una aplicación Web que integre estos servicios para la construcción del caso de uso *Intermodal Trip Planner*.

A continuación se muestran los requisitos funcionales agrupados por área de desarrollo:

- Área de desarrollo *Renfe Web Service*: ésta área de desarrollo engloba el proceso necesario para extraer la información de los trenes Castellón-Valencia a partir de la página Web de la empresa Renfe. Para la realización de esta área, se desarrollará un servicio Web que implementa la arquitectura del sistema propuesta en el Capítulo 3. El requisito que permite solventar esta aplicación Web es el US-8.

---

<sup>2</sup><http://www.oracle.com/technetwork/java/javase/overview/index.html>

<sup>3</sup><http://git-scm.com/>

<sup>4</sup><http://maven.apache.org/>

<sup>5</sup><https://jersey.java.net/>

<sup>6</sup><http://www.openstreetmap.org/>

- Área de desarrollo *Bicicas Web Service*: esta área de desarrollo consiste en la creación de una aplicación Web que implementa la arquitectura que se ha explicado en el Capítulo 3. Esta aplicación Web permite el acceso a la información proporcionada por la página Web del servicio de préstamo de bicis Bicicas. Los requisitos que se cumplen con esta aplicación Web son: US-3, US-4, US-10 y US-11.
- Área de desarrollo *Intermodal Trip Planner*: el última área de desarrollo permite solventar los requisitos: US-1, US-2, US-5, US-6, US-7 y US-9. Para ello, se creará una aplicación Web para la integración de la información proporcionada por los otros servicios, Renfe Web Service y Bicicas Web Service.

Las Secciones 4.3.1, 4.3.2 y 4.3.3 describen en detalle el análisis de cada área de desarrollo que se ha creado para desarrollar el caso de uso.

#### 4.3.1. Renfe Web Service

La aplicación Renfe Web Service es una aplicación que implementa la arquitectura Software propuesta en este documento para recuperar información de los trenes Castellón-Valencia a partir de la Web de la compañía Renfe.

La Figura 4.2 muestra los componentes de la arquitectura que se han usado para desarrollar esta aplicación Web. Estos son: Web Wrapping, XML Wrapper, Semantic Content Augmentation, REST Interface y Request Statistical Analysis. Como se puede notar, se han implementado todos los módulos obligatorios de la arquitectura. De forma adicional, se ha incluido el módulo de análisis de las peticiones Web para poder mejorar el servicio ofrecido en un futuro.

Ahora se van a describir las funcionalidades que cada módulo implementa.

El módulo Web Wrapping se encarga de hacer una petición HTTP a la página Web de Renfe cercanías<sup>7</sup>. En la Web, se encuentra un formulario para obtener informaciones sobre los trenes entre varias estaciones de cercanía. En la petición, el módulo tiene que enviar los datos de dicho formulario para que devuelva la información de los trenes entre las estaciones de Castellón y Valencia-Nord. Una vez obtenida la página Web de respuesta, se procede a la limpieza del código HTML. Este proceso consiste en crear un documento XHTML equivalente a la página descargada, pero cumpliendo las reglas del estándar XML. Este paso es necesario para poder manejar el árbol DOM (Document Object Model) de la página. Con la tecnología XPath se extrae la información de los trenes y se proporciona al siguiente módulo.

El componente XML Wrapper recoge la información que el módulo Web Wrapping ha extraído y crea un objeto Java para encapsulara. Luego, se transforma este objeto en un documento XML para que se pueda servir a través del servicio Web.

El módulo Semantic Content Augmentation aplica al documento creado por el módulo XML Wrapper una plantilla XSLT para añadir meta-datos al documento XML. En este caso, se añaden los microformatos necesarios para describir un trayecto en tren como un evento.

---

<sup>7</sup><http://horarios.renfe.com/cer/hjcer310.jsp>

El módulo REST Interface tiene que ofrecer las URL para poder acceder al servicio Web. En este caso, el servicio Web ofrecerá una única URL para la consulta del horario de los trenes entre dos estaciones de cercanía de la comunidad Valenciana entre las fechas especificadas.

La URL en cuestión es `/[departure]/[departureYear]/[departureMonth]/[departureDay]/[departureTime]/[arrival]/[arrivalYear]/[arrivalMonth]/[arrivalDay]/[arrivalTime]/`. Se accede a ella con el método HTTP GET.

A continuación se describe el significado de los parámetros de la URL y las partes de las URL:

- *departure*: es el código de la estación de salida del tren.
- *departureYear*: es el año de salida.
- *departureMonth*: es el mes de salida.
- *departureDay*: es el día de salida.
- *departureTime*: es la hora de salida.
- *arrival*: es el código de la estación de llegada del tren.
- *arrivalYear*: es el año de llegada.
- *arrivalMonth*: es el mes de llegada.
- *arrivalDay*: es el día de llegada.
- *arrivalTime*: es la hora de llegada.

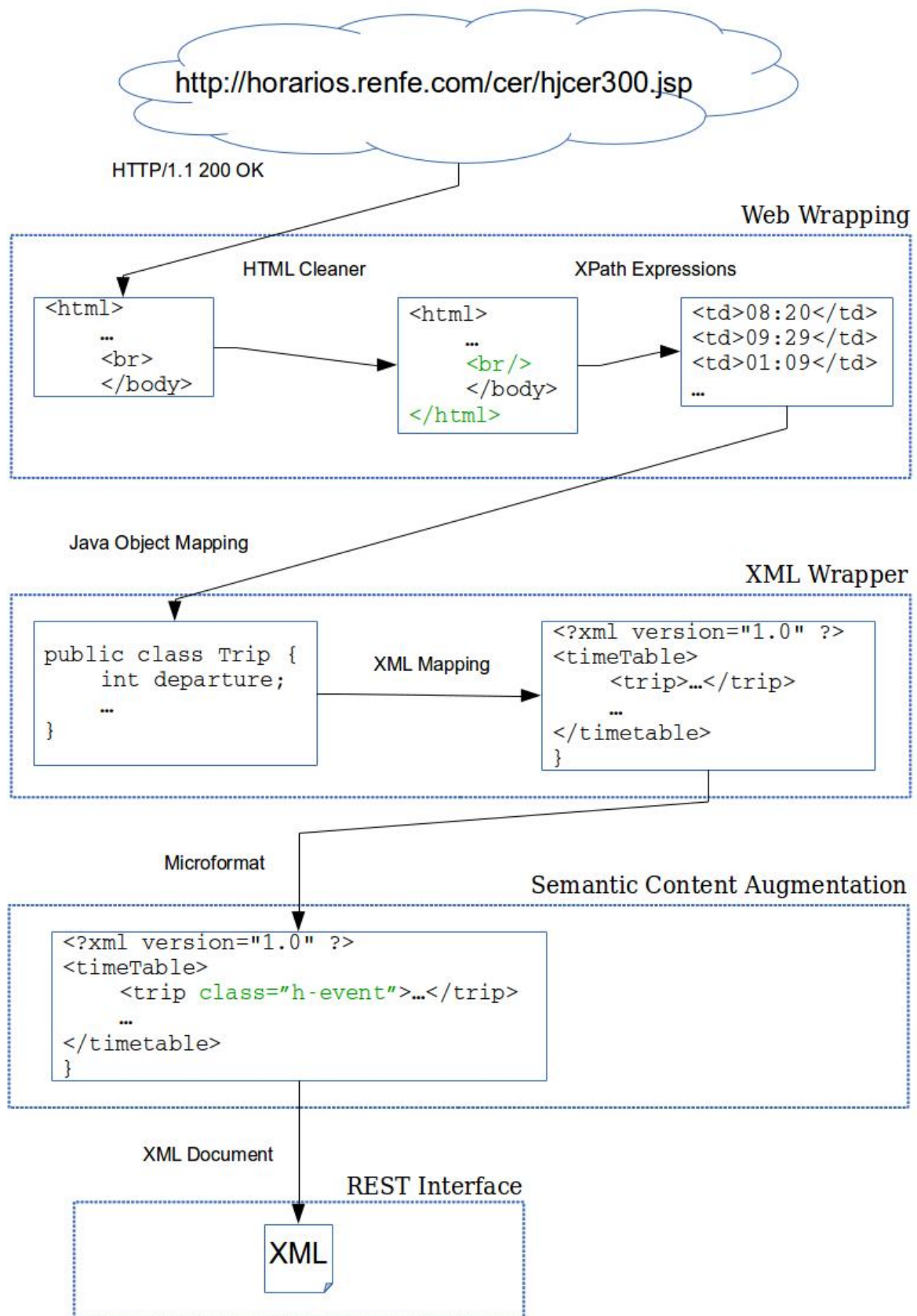


Figura 4.2: Esquema de la arquitectura mostrada en la Figura 3.1 particularizado para la implementación de la aplicación Renfe Web Service

### 4.3.2. Bicicas Web Service

La aplicación Web Bicicas Web Service es una aplicación que implementa la arquitectura Software propuesta en el Capítulo 3. Su tarea es extraer información de la página Web del servicio Bicicas y proporcionar un API para la consulta del estado de las bicis.

La Figura 4.3 muestra los componentes de la arquitectura que se han usado para desarrollar esta aplicación Web. Estos son: Web Wrapping, XML Wrapper, Semantic Content Augmentation, Persistence Module, REST Interface y Request Statistical Analysis. Se han implementado todos los módulos obligatorios de la arquitectura y, además, los módulos opcionales de persistencia y de análisis de las peticiones Web. El primer módulo opcional mencionado sirve para poder guardar el estado de las bicis en los diferentes puntos de Bicicas, además de un historial con los estados por los que ha pasado cada estación de Bicicas. El segundo módulo opcional se ha incluido para poder mejorar el servicio ofrecido en un futuro.

El módulo de persistencia se ha incluido en el desarrollo de la plataforma Web para poder dar soporte al análisis de los datos de las estaciones de Bicicas. Estos análisis no entran en el alcance de esta aplicación Web, pero en un futuro serán la base de partida para otros proyectos.

Ahora se van a describir las funcionalidades que cada módulo implementa.

El módulo Web Wrapping se encargará de realizar la petición HTTP a la página Web donde se encuentra el estado de las estaciones de Bicicas<sup>8</sup>. Una vez obtenida la página Web, se limpiará el código HTML y se usará la tecnología XPath para extraer la información de cada estación de Bicicas.

El módulo XML Wrapper manejará el módulo de persistencia para poder guardar los datos extraídos por el módulo Web Wrapping y para poder recuperarlos. Estos datos se mapearán en una clase Java para luego crear un documento XML para que se puedan enviar a través del servicio Web. El objeto Java creado será el que se pasará al módulo de persistencia para que se guarde en la base de datos.

El módulo Persistence Module recibirá los datos extraídos de la página Web de Bicicas y los guardará en una base de datos MongoDB. Se encargará de detectar si el estado de una estación a guardar en la base de datos es diferente al estado guardado. En ese caso, el módulo creará una entrada en una colección de historial para dar soporte al análisis futuro que se ha requerido.

El módulo Semantic Content Augmentation aplica al documento creado por el módulo XML Wrapper una plantilla XSLT para añadir meta-datos al documento XML. En este caso, se añaden los Microformatos necesarios para describir la localización geográfica de las estaciones de Bicicas.

El módulo REST Interface tiene que proporcionar todas las URL para que los usuarios de la aplicación Web puedan acceder a la información de las estaciones de Bicicas.

La aplicación tendrá una tarea que se ejecuta en segundo plano que se ocupa de obtener y transformar los datos que se extraen de la fuente de datos. De esta forma, la actualización del estado de las estaciones de Bicicas se lleva a cabo cada cierto tiempo.

---

<sup>8</sup><http://www.bicicas.es/estado/EstadoActual.asp>

El siguiente listado muestra las URL que la aplicación Web proporcionará:

- `bicicas/nearest?lat=&lon=`: esta dirección Web permite obtener la estación de Bicicas más cercana a las coordenadas de latitud y longitud especificadas en los parámetros. Método HTTP GET.
- `/bicicas[code]`: devuelve toda la información acerca de la estación de Bicicas especificada con el código en la URL. Se accede con el método HTTP GET.
- `/bicicas`: devuelve toda la información de todas las estaciones de Bicicas. Como en el caso anterior, se accede con el método HTTP GET.
- `/admin/tasks/[name]?period=`: cambia el periodo de ejecución de la tarea especificada con el parámetro *name*. El valor del periodo se especifica como parámetro en la URL. El acceso con el método HTTP POST permite enviar las credenciales de acceso a la parte de administración.
- `/admin/tasks/start`: habilita las tareas que se ejecutan en la aplicación Web. Las credenciales de acceso se envían en el cuerpo de la petición HTTP POST.
- `/admin/tasks/stop`: deshabilita las tareas que se ejecutan en la aplicación Web. Con el método HTTP POST se incluyen las credenciales de acceso a la parte de administración.

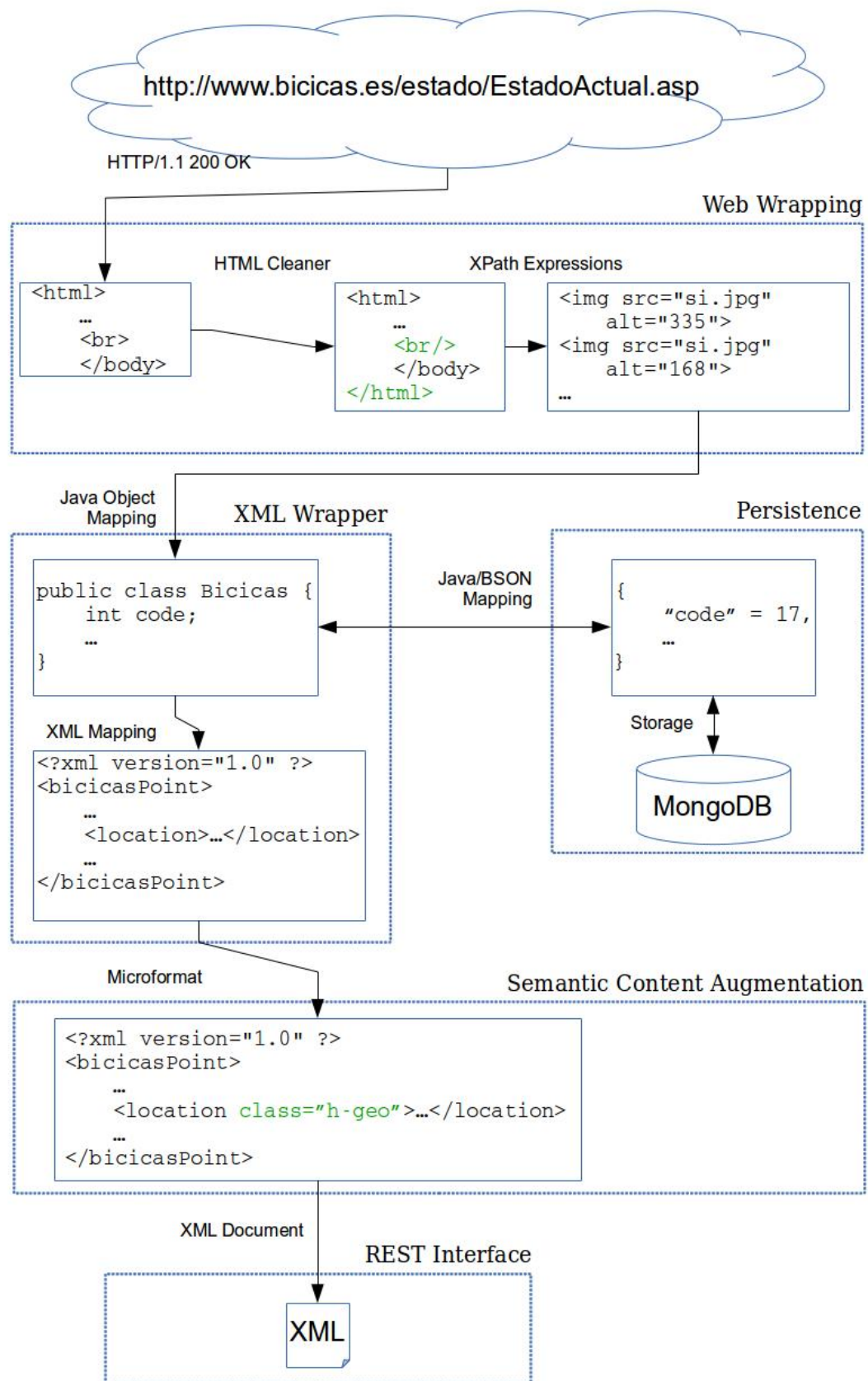


Figura 4.3: Esquema de la arquitectura mostrada en la Figura 3.1 particularizado para la implementación de la aplicación Bicas Web Service



### 4.3.3. Intermodal Trip Planner

La última aplicación Web a desarrollar representa la integración de los servicios generados implementando la arquitectura propuesta. Esta aplicación tiene como objetivo recomendar al usuario la estación de Bicicas más cercana a su posición, trazar una ruta hasta la estación de trenes y sugerir los trenes que se pueden llegar a coger. Para conseguir este objetivo, la aplicación presentará un componente mapa en su interfaz gráfica. En él se dibujará la posición del usuario, del sitio de Bicicas más cercano y la posición de la estación de trenes. Además, se visualizará la ruta sugerida por el sistema para llegar a la estación. Para representar la información de los trenes, se usará una tabla. En ella se incluirá la información de interés tal y como la hora de salida, la hora de llegada y la duración del trayecto.

Esta aplicación no implementa la arquitectura propuesta, sino que será una integración de la información proporcionada por Bicicas Web Service y Renfe Web Service entre otros servicios.

Intermodal Trip Planner hará uso de varios servicios Web para proporcionar la información que se visualizará en pantalla. El siguiente listado describe los servicios usados:

- Bicicas Web Service: servicio propio para obtener los datos de las estaciones de Bicicas. El análisis de esta aplicación Web se puede consultar en la Sección 4.3.2.
- Renfe Web Service: servicio propio para poder consultar los horarios de los trenes de cercanía de la Comunidad Valenciana. En la Sección 4.3.1 se puede consultar el análisis de esta aplicación Web.
- Nominatim<sup>9</sup>: servicio Web proporcionado por OpenStreetMap para realizar operaciones de *reverse geocoding*. Estas operaciones consisten en devolver la dirección más cercana a las coordenadas de latitud y longitud pasadas como parámetro. Este servicio se usará en la aplicación Web para poder encontrar la dirección de donde está usando la aplicación el usuario y la dirección de la estación de Bicicas más cercana. Las coordenadas del usuario son conocidas porque se pedirán al servicio de geolocalización del dispositivo. Las coordenadas de la estación de Bicicas más cercana se extraerán de los meta-datos (Microformato h-geo) que estarán incluidos en la respuesta del servicio Bicicas Web Service.
- Google Calendar API<sup>10</sup>: servicio Web proporcionado por Google que permite interactuar con las funcionalidades del calendario de Google. Se usarán las APIs Web de este servicio para poder añadir eventos al calendario de Google del usuario de la aplicación. Los eventos a añadir serán los viajes en tren Castellón-Valencia que el usuario quiera guardar en su calendario personal. Los datos de los viajes serán extraídos de los meta-datos (Microformato h-event) que se encuentran en la respuesta del servicio Renfe Web Service.

Estos servicios se integrarán en la aplicación de forma transparente al usuario y permitirá ofrecer toda la funcionalidad requerida.

A continuación se presentan los mockups que se han realizado para la aplicación. Éstos se han presentado en una reunión con las partes interesadas en el proyecto y han sido validado al final de la misma.

---

<sup>9</sup><http://wiki.openstreetmap.org/wiki/Nominatim>

<sup>10</sup><https://developers.google.com/apis-explorer/#p/calendar/v3/>

Se han propuesto tres mockups para la aplicación Intermodal Trip Planner. El primero se muestra en la Figura 4.4 y representa el aspecto de la aplicación Web una vez cargada. Al usuario se presenta un botón para empezar el proceso de localización, de búsqueda de la estación de Bicicas más cercana y de la búsqueda de los trenes disponibles.

El mockup mostrado en la Figura 4.5 muestra el estado de la aplicación una vez localizado el usuario, el Bicicas más cercano y la estación de trenes. Estos tres elementos se muestran en un mapa. En la figura, el punto verde (1) muestra la posición del usuario de la aplicación, el punto violeta (2) muestra el sitio de Bicicas más cercano y el punto azul (3) muestra la localización de la estación de Renfe.

El último mockup, Figura 4.6 representa la página Web completamente cargada y con toda la información. A parte del mapa descrito arriba, hay un apartado donde se mostrará información al usuario. Esta información, por ejemplo, será el número de bicis disponibles en el sitio seleccionado así como la distancia entre el usuario y la estación. El último elemento de la página es una tabla donde se muestran los trenes Castellón-Valencia que se estima que se pueden coger, teniendo en cuenta el tiempo de llegada a la estación de Renfe.

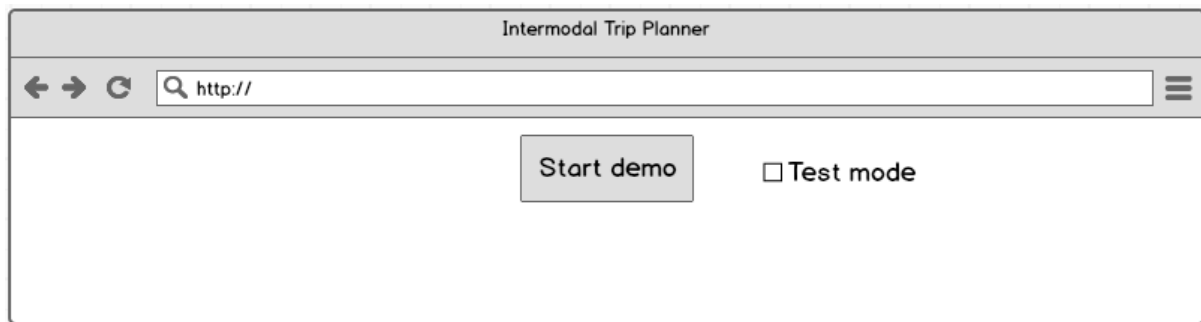


Figura 4.4: Mockup de la página Web cargada

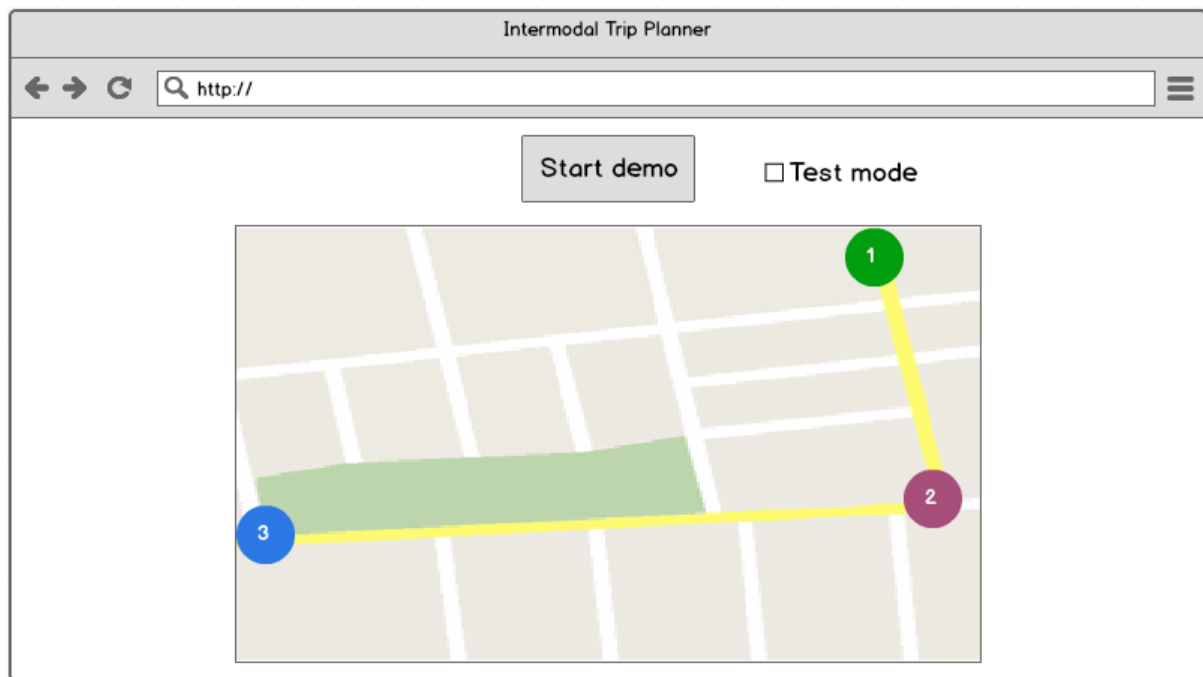


Figura 4.5: Mockup de la página Web mostrando el usuario, la estación de Bicicas más cercana y la estación de Renfe

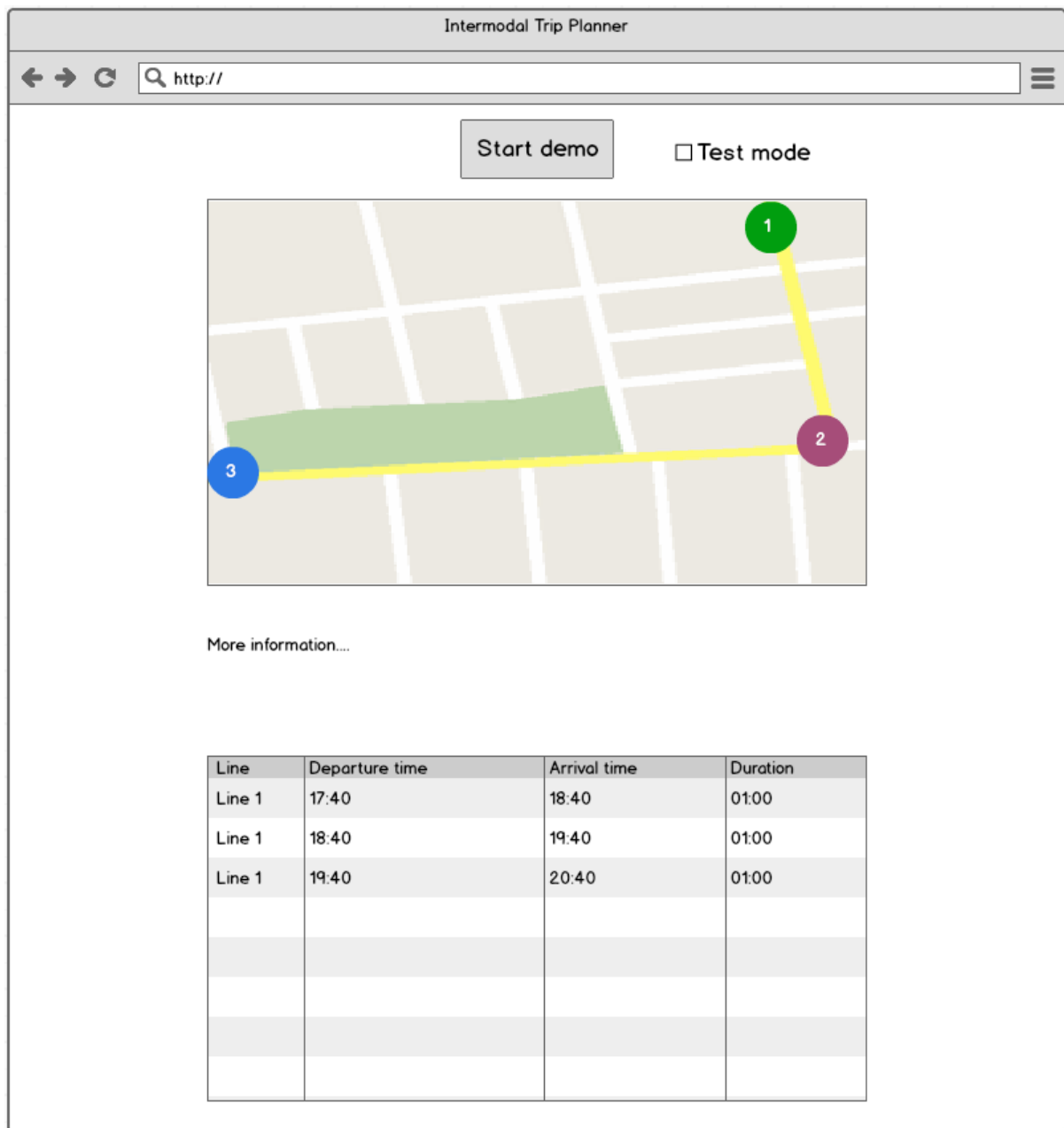


Figura 4.6: Mockup de la página Web completa. Se muestra toda la información que tiene que aparecer, incluso el mapa, la tabla con los trenes y la información adicional

#### 4.4. Tecnologías usadas

En esta sección se describen y justifican las tecnologías usadas para realizar el desarrollo del caso de uso propuesto para la estancia en prácticas. Algunas de estas tecnologías han sido un requisito de la empresa mientras que otras han sido seleccionadas entre las disponibles.

A continuación se describe cada una de ellas.

## Java EE

Java es un lenguaje de programación orientado a objetos muy usado en la actualidad. Es el lenguaje que se ha impartido en las principales asignaturas de programación en el Grado de Ingeniería Informática. Java permite desarrollar aplicaciones que se ejecutan tanto en local como en remoto. Recientemente, es el lenguaje de programación usado para diseñar aplicaciones para el sistema operativo móvil Android<sup>11</sup>. La plataforma Java EE, Java Enterprise Edition, permite el desarrollo rápido y escalable de aplicaciones Web. Estas aplicaciones se pueden desplegar en muchos servidores, tanto en máquinas virtuales propias como en la nube. Además, el lenguaje, el servidor de aplicaciones Web y las herramientas para desarrollar en Java son Open Source.

## Apache Tomcat

Apache Tomcat<sup>12</sup> es una implementación Open Source de la tecnología Java Servlet y Java Server Pages. Permite la ejecución de aplicaciones Web desarrolladas con tecnología Java EE.

Se ha elegido como contenedor de Servlet para el proyecto porque es uno de los más usados por la comunidad de desarrolladores.

## Git

Git es un sistema de control de versiones distribuido Open Source. Se ha creado en el 2005 por Linus Torvalds para el proyecto Kernel Linux. Este sistema permite mantener un historial de los cambios en los ficheros de un proyecto. Aunque se puede para mantener versiones de proyectos de cualquier naturaleza, es muy común usarlo para proyectos Software. Esto es debido a que es una herramienta muy potente y flexible cuando se usa con ficheros de texto, como los del código fuente de aplicaciones. Git es una herramienta muy avanzada que permite a varias personas trabajar a la vez en un proyecto y permite ayudar a integrar el código escrito por cada miembro del equipo.

Existen muchas herramientas para el control de versiones. Sin embargo, las soluciones más usadas en las empresas son los sistemas de control de versiones distribuidos. Éstos conllevan grandes ventajas frente a los sistemas centralizados, como por ejemplo que cada usuario del sistema tiene una copia de todo el repositorio y no es necesario estar conectado al servidor centralizado para poder trabajar. Git tiene una comunidad de usuarios y desarrolladores muy amplia, lo cual garantiza futuro al proyecto.

## Maven

Maven es un Software para la automatización de la compilación de proyectos informáticos, particularmente usado para aplicaciones Java. En su uso básico, permite organizar las dependen-

---

<sup>11</sup><http://www.android.com/>

<sup>12</sup><http://tomcat.apache.org/>

cias de los proyectos Software y automatizar la compilación y ejecución de tests unitarios. Es un Software Open Source desarrollado por Apache Software Foundation. Las reglas de construcción de un proyecto (compilación y ejecución) se especifican en un fichero XML denominado Project Object Model (POM). Cada proyecto Maven tiene que tener un fichero POM donde se describen las dependencias y reglas de compilación. A través de un sistema de plugins permite aumentar sus funcionalidades. Entre los plugins más útiles se encuentran los que permiten compilar y desplegar las aplicaciones de Servidor desarrolladas en Java.

## Jersey

Jersey es un framework Java que permite el desarrollo de aplicaciones Web RESTFul. Es la implementación oficial de la especificación JAX-RS API<sup>13</sup> (JSR 311 y JSR 339). Es un Software Open Source desarrollado por Oracle. Este framework proporciona una interfaz de programación sencilla para abstraer la complejidad al trabajar con los Servlet de la plataforma Java. Además, permite aumentar la flexibilidad a la hora de la programación de aplicaciones Web proporcionando clases y métodos de utilidad para el desarrollo específico de aplicaciones Web 2.0 y servicios REST. Existen muchos otros framework para el desarrollo de aplicaciones Web en la plataforma Java, como JBoss RESTeasy<sup>14</sup> o Spring<sup>15</sup>. Sin embargo, se ha elegido Jersey para este proyecto porque es la implementación oficial del estándar JAX-RS.

## OpenStreetMap

OpenStreetMap<sup>16</sup> es un proyecto libre para la creación y el mantenimiento de un mapa de todo el mundo. Los usuarios pueden insertar o modificar la posición, por ejemplo, de edificios u obras de arte para actualizar los mapas. El uso de estos mapas es libre y su contenido es Open Source.

## HTML5/CSS3

HTML5[15] es la quinta iteración del estándar HTML. Este lenguaje permite describir la estructura de las páginas Web con etiquetas y atributos. La quinta versión todavía no está en versión definitiva, pero las empresas desarrolladoras de navegadores Web ya empiezan a implementar partes del estándar.

CSS3 es el lenguaje para la especificación de estilo de las páginas Web escritas en HTML. Tiene una sintaxis propia (no se basa en XML) y permite modificar el aspecto y el posicionamiento de las etiquetas HTML.

---

<sup>13</sup><https://jax-rs-spec.java.net>

<sup>14</sup><http://resteasy.jboss.org/>

<sup>15</sup><https://spring.io/>

<sup>16</sup><http://www.openstreetmap.org/>

## Javascript

Javascript es una implementación del estándar ECMA Script[4]. Javascript permite desarrollar código que se ejecuta en los navegadores Web para implementar funcionalidad avanzada, como validación de datos o carga de contenido asíncrona (AJAX). Es un lenguaje ligero e interpretado, es multiplataforma y orientado a objetos.

## jQuery

jQuery<sup>17</sup> es una biblioteca para Javascript ampliamente usada hoy en día para el desarrollo de aplicaciones Web complejas. Proporciona métodos y objetos para, por ejemplo, una fácil manipulación del árbol DOM, para manejar eventos, animaciones y peticiones AJAX. La biblioteca es Open Source.

Se ha decidido incluir esta biblioteca en el proyecto para reducir el tiempo de desarrollo de la aplicación Intermodal Trip Planner.

## Leaflet

Leaflet<sup>18</sup> es una biblioteca Javascript Open Source que permite el manejo de los mapas en páginas Web. Proporciona métodos para incluir mapas de diferentes proveedores de forma muy fácil e intuitiva. Tiene soporte por defecto por los mapas del servicio OpenStreetMap.

Se ha decidido usar esta biblioteca porque tiene soporte nativo para los mapas de OpenStreetMap y porque tiene una comunidad muy activa que contribuye al proyecto.

## MongoDB

MongoDB<sup>19</sup> es una base de datos Open Source NoSQL basada en documentos. Los documentos que se guardan de forma persistente no tienen un esquema fijo (no es necesario especificarlo a priori) y tienen una sintaxis similar al formato JSON. Soporta la creación de índices, está optimizado para escalar horizontalmente y el API para las consultas es muy simple e intuitiva. Permite crear índices 2D y 3D sobre atributos y se pueden hacer consultas geográficas sobre esos atributos.

Se ha elegido este tipo de base de datos porque permite guardar los objetos de los lenguajes de programación de forma muy simple. Además, tiene prestaciones excelentes y es uno de los motores de base de datos NoSQL más avanzados hoy en día.

---

<sup>17</sup><http://jquery.com/>

<sup>18</sup><http://leafletjs.com/>

<sup>19</sup><http://www.mongodb.org/>

## Microformato h-event

h-event<sup>20</sup> contiene algunas clases que permiten describir eventos de muchos tipos. Las clases usadas son:

- *h-event*: esta clase indica que se esta empezando a definir un evento.
- *p-name*: el contenido de la etiqueta marcada con esta clase incluye el nombre del evento.
- *p-category*: con esta clase se describe la categoría del evento que se esta describiendo.
- *dt-start*: esta clase indica la hora y fecha en la cual empieza el evento. El contenido de la etiqueta marcada con esta clase tiene que expresarse en formato ISO8601<sup>21</sup> para que sea legible de forma universal.
- *dt-end*: esta clase indica la hora y fecha de finalización del evento. El contenido de la etiqueta marcada con esta clase es recomendable que expresarse en formato ISO8601.
- *dt-duration*: el contenido de la etiqueta marcada con esta clase describe la duración del evento.

Existen muchos formatos para definir eventos. Uno de los más usados es h-calendar<sup>22</sup>. Para este proyecto se ha elegido usar h-event porque representa la segunda versión del microformato h-calendar.

## Microformato h-geo

h-geo<sup>23</sup> es un Microformato que contiene algunas clases que permiten describir coordenadas geográficas. Las clases usadas son:

- *h-geo*: esta clase indica que se esta definiendo una localización.
- *p-latitude*: el contenido de la etiqueta marcada con esta clase contiene una coordenada de latitud.
- *p-longitude*: con esta clase se describe el valor de longitud.

## HTML Cleaner

HTML Cleaner<sup>24</sup> es una biblioteca para el lenguaje de programación Java que permite interactuar con documento HTML o XML. En presencia de un documento HTML, la biblioteca

---

<sup>20</sup><http://microformats.org/wiki/h-event>

<sup>21</sup><http://www.iso.org/iso/iso8601>

<sup>22</sup><http://microformats.org/wiki/hcalendar>

<sup>23</sup><http://microformats.org/wiki/h-geo>

<sup>24</sup><http://htmlcleaner.sourceforge.net/>



limpia el código del documento produciendo un documento XML equivalente, que tiene un árbol DOM (Document Object Model) válido. Una vez hecho este pre-procesamiento, la biblioteca es capaz de recorrer el árbol DOM y ejecutar consultas XPath.

Esta biblioteca es Open Source y la comunidad es muy activa. Además tiene un soporte nativo para el sistema de gestión de dependencia Maven.

## **XPath**

XPath [9] es un estándar del W3C<sup>25</sup> (World Wide Web Consortium) que especifica un lenguaje de consultas para documentos XML. Este lenguaje permite especificar las trayectorias hacia las etiquetas XML de un documento.

Esta tecnología se ha elegido por ser el método estándar para consultar etiquetas de un documento XML.

## **XSLT**

XSLT [10] es un lenguaje estándar de la asociación W3C que permite transformar un documento XML en otro documento XML.

El documento que especifica la transformación es también XML, por lo tanto este lenguaje es totalmente independiente de la arquitectura o lenguaje de programación usado. Para especificar trayectorias a etiquetas a manipular usa el lenguaje de consulta XPath.

Se ha elegido esta tecnología para manipular y transformar documentos XML porque es la manera estándar para hacer este proceso y porque es independiente del lenguaje de programación usado.

## **XML**

XML [12] es un lenguaje de marcado que permite definir documentos que contienen información accesible a humanos y máquinas. Esto es posible gracias a su sintaxis muy comprensible.

Es un lenguaje que se ha transformado en el estándar de-facto para el envío de información en la Web debido a su gran flexibilidad en la sintaxis. Esto permite poder representar cualquier tipo de dato o objeto de forma fácil.

Otra característica muy importante de XML es que se puede especificar un documento de validación para validar los documentos XML y verificar que contienen la información correcta. El lenguaje más usado para hacer esta tarea es XSchema<sup>26</sup>.

---

<sup>25</sup><http://www.w3c.com/es/>

<sup>26</sup><https://schema.org/>

## YourNavigation

YourNavigation<sup>27</sup> es un sistema Open Source para la planificación de rutas. Usa el sistema de mapas de OpenStreetMap y algoritmos específicos para encontrar rutas entre dos puntos en un mapa.

Se ha elegido usar este servicio de rutas por ser gratuito y Open Source.

## 4.5. Desarrollo

En esta sección se describe la fase de desarrollo del proyecto. La realización del proyecto se ha dividido en varios Sprints con duración de 15 días cada uno. El Cuadro 4.3 muestra la planificación de los Sprints de la estancia en prácticas. Además, se muestra la duración real que ha llevado acabar la tarea de cada Sprint. El horario de trabajo consiste en 8 horas diarias y las fechas de la estancia en prácticas son del 01/04/2014 al 30/06/2014.

Sprint	Duración estimada	Duración real
Sprint 1	15 días	20 días
Sprint 2	15 días	10 días
Sprint 3	15 días	15 días
Sprint 4	15 días	19 días
Sprint 5	15 días	11 días
Sprint 6	15 días	14 días

Cuadro 4.3: Cuadro que resume la duración estimada y real de cada Sprint del proyecto

La Sección 4.5.1 describe el desarrollo de una biblioteca que se encarga de la ejecución de tareas en segundo plano. Esta biblioteca se ha creado para este proyecto, pero en un futuro se puede integrar en otros. La Sección 4.5.2 muestra en detalle la descripción de todos los Sprints del proyecto.

### 4.5.1. Scheduled Executor Framework

En esta sección se presenta el trabajo que se ha hecho para desarrollar una biblioteca para la gestión de tareas programadas en la plataforma Java. El nombre de esta biblioteca es Scheduled Executor Framework.

La necesidad que cubre esta biblioteca es de poder ejecutar tareas cada cierto intervalo de tiempo. Cada tarea tiene un intervalo mínimo de tiempo entre cada ejecución y se puede iniciar y detener.

La biblioteca se encarga de la distribución de las tareas en varios hilos de ejecución para

---

<sup>27</sup><http://www.yournavigation.org/>

aprovechar los recursos de la máquina donde se ejecuta el código. Esta gestión es totalmente transparente al usuario (desarrollador) de la biblioteca.

Para la implementación de Scheduled Executor Framework, se ha aprovechado de las clases que proporciona el lenguaje Java para el manejo asistido de hilos. En particular, se usa una clase que se llama *ScheduledThreadPoolExecutor*<sup>28</sup>. Esta clase permite la ejecución de tareas (objetos que implementan la interfaz *Runnable*<sup>29</sup> o *Future<V>*<sup>30</sup>) especificando un tiempo mínimo que tiene que pasar entre cada ejecución de la misma tarea. Además, este objeto optimiza el número y el comportamiento de los hilos de ejecución para aprovechar al máximo los recursos Hardware.

La biblioteca Scheduled Executor Framework proporciona una clase abstracta (*AbstractScheduledTask*) para que se puedan crear diferentes tareas a ejecutar.

La clase *ScheduledExecutor* es la clase principal de la biblioteca y es la que se encarga de gestionar todas las tareas en ejecución. Esta clase se ha implementado de manera que sea tolerante a la concurrencia (Thread-safe), esto quiere decir que diferentes hilos de ejecución puede compartir la misma instancia de la clase sin que haya ningún problema de concurrencia o condiciones de carrera.

#### 4.5.2. Sprints

En esta sección se van a describir los Sprint que se han realizado para llevar a cabo el proyecto de prácticas. Casa Sprint tiene una duración de 15 días. Al final de cada uno se ha realizado una reunión con todas las partes involucradas en el proyecto para presentar los avances con respecto al Sprint anterior.

La planificación inicial de los Sprints se ha incluido en la propuesta técnica de la estancia en prácticas. Sin embargo, tras las reuniones iniciales del proyecto se ha decidido cambiar algunos Sprints para mejorar la organización del proyecto. A continuación se resume la planificación:

- Sprint 1: diseño conceptual de la arquitectura del sistema.
- Sprint 2: definición del caso de uso para la demostración del funcionamiento de la arquitectura.
- Sprint 3: realización del área de desarrollo Renfe Web Service.
- Sprint 4: realización del área de desarrollo Bicicas Web Service.
- Sprint 5: realización del área de desarrollo Intermodal Trip Planner.
- Sprint 6: revisión del sistema entero en funcionamiento, puesta en marcha y revisión de la memoria de prácticas.

A continuación se van describiendo en detalle todos los Sprints que se han realizado.

---

<sup>28</sup><http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/ScheduledThreadPoolExecutor.html>

<sup>29</sup><http://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html>

<sup>30</sup><http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/Future.html>

## **Sprint 1**

El primer Sprint del proyecto se ha centrado en la creación del diseño conceptual de la arquitectura que se va a proponer para la extracción de información de recursos Web y la distribución a través de API REST.

En este Sprint se han realizado muchas reuniones con las partes interesadas del proyecto, el equipo de desarrollo, el Product Owner y el Scrum Master. En estas reuniones la empresa ha expresado los objetivos que se quieren conseguir con la arquitectura y con la estancia en prácticas.

El resultado de este Sprint ha sido el esquema que representa el diagrama conceptual de la arquitectura. La Figura 3.1 muestra el esquema final. En el Capítulo 3 se describe en detalle el resultado de este Sprint. La duración de este Sprint ha sido de 20 días, 5 más de la planificación. Esto es porque se han tenido que realizar refinamientos sobre la arquitectura propuesta que han llevado a un alargamiento de la duración de este Sprint.

## **Sprint 2**

El objetivo del segundo Sprint del proyecto ha sido la definición del caso de uso que se tendrá que implementar para demostrar las funcionalidades de la arquitectura diseñada en el primer Sprint.

Para conseguir este objetivo se han hecho muchas reuniones entre el equipo de desarrollo, el Product Owner y el Scrum Master. En ellas se han barajado los posibles casos de uso que podrían verse beneficiados de servicio Web con metadatos semánticos. Entre ellos, se ha evaluado la posibilidad de continuar uno de los proyectos en marcha en la empresa.

Al final de las reuniones, se ha decidido implementar un caso de uso que use la información de las estaciones de Biciclas de Castellón. Este tema está en la línea de desarrollo de la empresa siendo directamente relacionado con el concepto de Smart Cities. La empresa está desarrollando varios proyectos para las ciudades inteligentes y el desarrollo de un prototipo de aplicación para la consulta de datos del servicio Biciclas puede ser muy útil en un futuro. Por ejemplo, se puede integrar este servicio en la plataforma Smart Campus, actualmente en desarrollo. En la versión actual, esta plataforma permite acceder a las informaciones del campus de la UJI y poder buscar recursos (despacho o aulas). Debido al diseño modular de dicha aplicación, en futuro se ampliarán sus funcionalidades.

El resultado de este Sprint es el listado de los requisitos funcionales y no funcionales así como la descripción del caso de uso a desarrollar. Los requisitos se pueden consultar en la Sección 4.2 de este documento. Este Sprint ha tenido una duración de 10 días, 5 días menos que la planificación.

## Sprint 3

El tercer Sprint del proyecto se ha dedicado al desarrollo de la aplicación Renfe Web Service. El análisis de esta aplicación Web se muestra en la Sección 4.3.1.

Para el desarrollo de Renfe Web Service se ha creado un proyecto Maven con todas las dependencias que requiere el framework Jersey. El siguiente listado muestra las dependencias del proyecto:

- jersey-container-servlet, versión 2.8
- jersey-server, versión 2.8
- jersey-media-json-jackson, versión 2.8
- jongo, versión 1.1
- mongo-java-driver, versión 2.11.4
- jersey-client, versión 2.8
- htmlcleaner, versión 2.8

Lo primero que se ha hecho antes de empezar el desarrollo, es analizar la fuente de los datos de los trenes de cercanías de la Comunidad Valenciana. Los datos se pueden encontrar en la Web de Renfe en la sección de cercanías de Valencia<sup>31</sup>

Para poder acceder al listado de los trenes, hay que rellenar un formulario. En él se tiene que especificar la estación de origen y destino, la fecha del viaje y el intervalo de horas. Una vez enviado el formulario, el sistema de Renfe genera la página Web con la tabla que contiene el detalle de los trenes que coinciden con los parámetros especificados.

Para hacer esta operación desde la aplicación Renfe Web Service, se ha tenido que estudiar el código fuente de la página Web que contiene el formulario para poder deducir los parámetros que hay que enviar para obtener el listado de los trenes. Éstos parámetros son:

- regionCode.
- i.
- cp.
- departureStationCode.
- arrivalStationCode.
- departureDateTime.
- arrivalDateTime.

---

<sup>31</sup><http://www.renfe.com/viajeros/cercanias/valencia/index.html>

- TXTInfo.

Algunos de los parámetros son necesarios para que la aplicación en el servidor sea capaz de interpretar correctamente los datos, pero otros son los que se tienen que cambiar dinámicamente para simular la selección de los campos del formulario. Éstos campos son:

- `departurStationCode`: es el código de la estación de salida de los trenes. Para el caso de uso que se ha desarrollado, el código de la estación de salida será 65300, representando la estación de Castellón.
- `arrivalStationCode`: es el código de la estación de llegada de los trenes. Para el caso de uso que se ha desarrollado la estación de llegada será Valencia-Nord, cuyo código es 65000.
- `departureDateTime`: es la mínima hora de salida de los trenes. Se especifica en milisegundos.
- `arrivalDateTime`: es la máxima hora de llegada de los trenes. Se especifica en milisegundos.

La variación de estos parámetros permite a Renfe Web Service devolver el horario de los trenes de la Comunidad Valenciana.

Una vez rellenado por código el formulario, éste se envía a la siguiente URL: `http://horarios.renfe.com/cer/hjcer310.jsp`

La página Web devuelta contiene el resultado de la búsqueda de trenes con el horario de los que respetan las condiciones especificadas. Esta página se tiene que analizar para obtener las trayectorias XPath para poder extraer la información de dicha tabla.

Una vez encontradas las consultas XPath para obtener la información requerida, se procede a aplicarlas sobre la página Web. Para hacer esta operación, es necesario usar la biblioteca HTML Cleaner. Esta biblioteca Java procesa y limpia un documento HTML y devuelve un documento XML bien formado de tal manera que se pueda navegar en su DOM (Document Object Model). Con los métodos de los objetos de la biblioteca HTML Cleaner se aplican las trayectorias XPath y se construye un objeto Java que encapsula la información de los viajes en tren.

Este objeto Java ya se puede servir a través del servicio Web. Esto es gracias a que la versión de Java usada para el desarrollo de la aplicación es la 7, que implementa internamente la especificación JAXB<sup>32</sup>. Esta característica de Java permite crear un documento XML a partir de un objeto Java. De esta forma, el framework Jersey puede devolver la información de los trenes en formato XML automáticamente.

En el caso de que el usuario haya solicitado la información de los trenes con contenido semántico, se procede a añadir las etiquetas y atributos necesarios. Para hacer esto, se genera un documento XML a partir de la clase Java que encapsula la información de los trenes, luego se aplica una plantilla de transformación XSLT al documento para que se añadan los metadatos necesarios. En el caso de los viajes en tren, se tienen que añadir metadatos para describir cada viaje como un evento. Para ello, se usa el Microformato h-event. Este Microformato define

---

<sup>32</sup><https://jaxb.java.net/>

algunos atributos que se tienen que añadir a las etiquetas XML. En la Sección 4.4 se puede consultar la descripción de este Microformato.

La implementación del módulo Request Statistical Analysis se lleva a cabo con una tarea programada en Scheduled Executor Framework. Esta tarea guarda periódicamente en una colección MongoDB los estadísticos de las peticiones HTTP que se hacen en el módulo REST Interface.

Al final de este Sprint, se ha entregado la aplicación Renfe Web Service. El tiempo de desarrollo de esta aplicación ha sido el estimado, 15 días.

## **Sprint 4**

El objetivo del cuarto Sprint del proyecto ha sido la realización de la aplicación Bicicas Web Service. En la Sección 4.3.2 se muestra el análisis de esta aplicación Web.

Para el desarrollo de Bicicas Web Service se ha creado un proyecto Maven separado. Para este proyecto se han especificado las siguientes dependencias:

- jersey-container-servlet, versión 2.8
- jersey-server, versión 2.8
- jersey-media-json-jackson, versión 2.8
- jongo, versión 1.1
- mongo-java-driver, versión 2.11.4
- jersey-client, versión 2.8
- htmlcleaner, versión 2.8

Este proyecto implementa, además de los componentes obligatorios, el módulo de persistencia de la arquitectura descrita en el Capítulo 3. En esta estructura de módulos, el objeto Java construido por el módulo XML Wrapper se guarda de forma persistente. La información requerida por el módulo REST Interface se recupera del módulo Persistence Module, no se obtiene de la fuente de datos original. De esta forma, la actualización de los datos en el módulo Persistence Module se tiene que hacer cada cierto tiempo. Para conseguir este objetivo, se ha desarrollado una tarea periódica que se ejecuta en el Scheduled Executor Framework desarrollado para este proyecto.

La página Web donde se encuentra la información del estado de las estaciones de Bicicas es: <http://www.bicicas.es/estado/EstadoActual.asp>. El primer paso consiste en analizar la estructura de dicha página Web para encontrar las etiquetas HTML que contiene la información de interés.

El análisis del código HTML produce las trayectorias XPath que se usan para extraer los datos del estado de las estaciones de Bicicas.

Una vez obtenidas las consultas XPath, se procede a limpiar y procesar el documento HTML con la biblioteca HTML Cleaner. Después de que la biblioteca haya creado una estructura DOM válida, se aplican las consultas XPath para extraer información del documento.

Con la información extraída se crea un objeto Java por cada estación de Biciclas, este almacena el estado de la estación. El estado de las estaciones tiene en cuenta el número de bicis disponible, la posición de las mismas en los anclajes y las bicis que han salido y entrado en la estación desde el último estado. Este detalle, permite tener traza de los movimientos de cada bicicleta en las estaciones de Castellón. Esto es posible porque en la tabla que muestra el estado de las estaciones en la página Web de Biciclas, contiene el identificador de la bicicleta en cada anclaje.

El objeto creado para encapsular el estado de las estaciones se guarda en una colección MongoDB para garantizar la persistencia. Para mantener un historial de los estados, antes de actualizar el estado de una estación se comprueba si éste ha cambiado. En caso positivo, el estado antiguo se mueve a una colección de historial y el estado corriente se actualiza.

Esta operación de consulta y recopilación de información acerca de las estaciones de Biciclas se ejecuta periódicamente. La frecuencia de actualización se puede modificar, pero por defecto es de un minuto. Este tiempo se ha establecido porque es un intervalo de actualización razonable que permite mantener los datos suficientemente actualizados y no sobrecargar la máquina donde se despliega la aplicación.

Cuando se recibe una petición de datos, el módulo REST Interface pide la información al módulo XML Wrapper. Éste recoge la información de la colección MongoDB (módulo Persistence Module). En el caso de que la petición incluya meta-datos semánticos, el módulo Semantic Content Augmentation aplica una hoja de transformación XSLT al documento generado por el módulo XML Wrapper. Durante la transformación se incluye el Microformato h-geo para añadir información geográfica a las estaciones de Biciclas. En la Sección 4.4 se puede consultar la descripción del Microformato h-geo.

La implementación del módulo Request Statistical Analysis se lleva a cabo con una tarea programada en Scheduled Executor Framework. Esta tarea guarda periódicamente en una colección MongoDB los estadísticos de las peticiones HTTP que se hacen en el módulo REST Interface.

Al final de este Sprint, se ha entregado la aplicación Biciclas Web Service. Esta aplicación se ha desarrollado en 19 días, 4 más que la planificación. Esto es debido a la complejidad añadida del desarrollo del módulo de persistencia, con la introducción de una base de datos MongoDB.

## **Sprint 5**

El quinto Sprint se ha dedicado al desarrollo de la aplicación Intermodal Trip Planner. El análisis de esta aplicación se detalla en la Sección 4.3.3.

El código de la aplicación Intermodal Trip Planner esta dentro un proyecto Maven con las siguientes dependencias:



- jersey-container-servlet, versión 2.8
- jersey-server, versión 2.8
- jersey-media-json-jackson, versión 2.8
- jongo, versión 1.1
- mongo-java-driver, versión 2.11.4
- jersey-client, versión 2.8

Esta aplicación representa la integración de los dos servicios Web que se han desarrollado (Bicicas Web Service y Renfe Web Service). Además, se usan los servicios Web de terceros Nominatim (para hacer el reverse-geocode de coordenadas geográficas), Google Calendar API (para interactuar con la aplicación Calendar de Google) y YourNavigation para la consulta de rutas.

Considerando los requisitos de esta aplicación, se ha dividido la funcionalidad en siete áreas de desarrollo:

- Localizar el usuario.
- Localizar la estación de Bicicas más cercana.
- Localizar la estación de trenes de Renfe.
- Calcular la ruta desde la posición del usuario hasta la estación de Renfe.
- Obtener el listado con el detalle de los trenes.
- Obtener información reverse-geocode.
- Añadir evento a Google Calendar.

Estas áreas se han implementado en la parte cliente de la página Web de Intermodal Trip Planner, por lo tanto se han desarrollado con el lenguaje de programación Javascript. Para facilitar el desarrollo del código Javascript se ha usado la biblioteca jQuery. De esta forma, la interacción con el DOM del documento HTML y la realización de las peticiones AJAX han sido muy simples de desarrollar. Además, se ha usado la biblioteca Leaflet para añadir a la página Web un mapa usando el servicio OpenStreetMap. La biblioteca permite modificar el estado del mapa y añadir marcadores o rutas.

A continuación se explica en el detalle cada área de desarrollo.

### **Localizar el usuario.**

La aplicación Web necesita localizar el usuario para poder funcionar. El estándar HTML5 tiene una funcionalidad de geolocalización de dispositivos. Para esta tarea, se han añadido al

lenguaje Javascript varios métodos para poder determinar la posición del usuario. El método para hacer eso se llama `navigator.geolocation.getCurrentPosition(...)`<sup>33</sup>. En la llamada se pueden incluir varias opciones tales y como la antigüedad de la última posición capturada o si la localización tiene que ser de alta precisión.

Esta funcionalidad de estándar HTML5 todavía no ha sido implementada por todos los navegadores disponibles. Además, la implementación varía mucho y en los dispositivos móviles, el navegador Web no utiliza el posicionamiento GPS para devolver la localización con exactitud donde es posible hacerlo. Estos problemas son debidos a que esta tecnología todavía no es suficientemente madura. Por contra, todos los mayores navegadores Web están introduciendo esta funcionalidad en sus nuevas versiones. En el momento de hacer el test de la aplicación (junio 2013) el navegador Google Chrome<sup>34</sup> es el único que, en su versión móvil y desktop, consigue localizar al usuario.

### **Localizar la estación de Bicicas más cercana.**

Para localizar la estación de Bicicas más cercana a la posición del usuario se realiza una petición AJAX al Servidor de la aplicación Intermodal Trip Planner. Éste hace una consulta al servicio Web Bicicas Web Service. La respuesta se devuelve al cliente y se marca la posición del sitio de Bicicas en el mapa en la página Web.

### **Localizar la estación de trenes de Renfe.**

La posición de la estación de Renfe está disponible en el Servidor de la aplicación Intermodal Trip Planner. El cliente simplemente hace una petición AJAX para obtenerla. Al finalizar la petición, el cliente muestra la posición de la estación en el mapa.

### **Calcular la ruta desde la posición del usuario hasta la estación de Renfe.**

En este área de desarrollo se quiere obtener la ruta desde la posición del usuario hasta el sitio de Bicicas seleccionado y, luego, hasta la estación de trenes. Para hacer eso, se hacen dos llamadas AJAX al Servidor de la aplicación (Intermodal Trip Planner). Éste consulta el servicio Web YourNavigation para obtener las rutas. Las dos rutas obtenidas se dibujan en el mapa de la página Web.

### **Obtener el listado con el detalle de los trenes.**

La tabla con el horario y el detalle de los trenes se obtiene del servicio Web Renfe Web Service. Antes de hacer la petición, se calcula la hora estimada de llegada del usuario a la estación Renfe. Esto se hace teniendo en cuenta los datos de las rutas devueltas por el servicio YourNavigation.

---

<sup>33</sup>// <https://developer.mozilla.org/en-US/docs/Web/API/Geolocation.getCurrentPosition>

<sup>34</sup><http://www.google.com/intl/es/chrome/browser/>

Una vez calculada la hora de llegada, se consulta Renfe Web Service por los trenes desde la hora estimada de llegada hasta el final del día (23:59 horas). El resultado devuelto por el servicio Web se usa para construir una tabla HTML para mostrarla en la página Web.

### **Obtener información reverse-geocode.**

El servicio de reverse-geocode se ha introducido para demostrar las posibilidades que surgen al incluir metadatos geográficos (Microformato h-geo en este caso) en una página Web. El servicio que proporciona esta funcionalidad es Nominatim (OpenStreetMap). Con esto, se consigue obtener la dirección donde se encuentra el usuario y donde está la estación de Biciclas más cercana a su posición. Una vez obtenida esta información, se muestra en la página Web.

### **Añadir evento a Google Calendar.**

Añadir eventos al calendario de Google permite demostrar la utilidad de insertar metadatos (Microformato h-event en este caso) en la página Web. Se deja la posibilidad al usuario de añadir los trayectos en tren a su calendario Google. En la tabla que muestra el detalle de los trayectos, se ha insertado un botón que permite llamar al usuario autenticarse en su cuenta Google y añadir el evento del trayecto seleccionado a su calendario. Esta funcionalidad se lleva a cabo usando la biblioteca Javascript que Google proporciona para acceder a sus APIs.

El resultado de este Sprint es la aplicación Intermodal Trip Planner. La duración del desarrollo ha sido menor de la planificada, ocupando 11 días.

## **Sprint 6**

El sexto Sprint se ha centrado en la finalización del proyecto de prácticas. Se han hecho muchas reuniones con las partes interesadas (Product Owner, Scrum Master y equipo de desarrollo) para revisar todas las aplicaciones en funcionamiento. Esta revisión se ha realizado con el fin de garantizar que el producto final cumpla con las expectativas del cliente. Esta evaluación se centra en el conjunto de las tres aplicaciones Web (Biciclas Web Service, Renfe Web Service y Intermodal Trip Planner) funcionando en conjunto. La evaluación de las singulas aplicaciones se ha llevado a cabo en la reunión de finalización de los Sprints 3, 4 y 5.

Una vez comprobado que el producto final cumple con todos los requisitos y las expectativas del cliente, se ha procedido al despliegue de las tres aplicaciones Web. Por ello, se ha solicitado la creación de una máquina virtual en la red de la UJI. Dicha máquina es *inti.init.uji.es*.

En la máquina virtual se ha instalado el contenedor de Servlet Apache Tomcat y el motor de base de datos MongoDB. Después de haber comprobado el correcto funcionamiento de las aplicaciones instaladas, se han desplegado los tres proyectos Web en el servidor. El siguiente listado muestra la dirección URL base de cada una de ellas:

- Biciclas Web Service: <http://inti.init.uji.es:8080/BiciclasWebService/>

- Renfe Web Service: <http://inti.init.uji.es:8080/RenfeWebService/>
- Intermodal Trip Planner: <http://inti.init.uji.es:8080/TripPlanner/>

Como recordatorio, la URL completa para acceder a la aplicación Web Intermodal Trip Planner es <http://inti.init.uji.es:8080/TripPlanner/ws/index>.

El Sprint se ha finalizado con la revisión final de la memoria de prácticas, con el tutor académico, y su entrega. La duración de este Sprint ha sido de 14 días, 1 día menos de lo estimado.

## Capítulo 5

# Conclusiones

El trabajo llevado a cabo en el marco de la asignatura de prácticas *EI1054 - Prácticas Externas y Proyecto Final de Grado* ha permitido ampliar los conocimientos adquiridos en las asignaturas cursadas en los cuatro años de carrera. Se han podido aplicar muchos de los conceptos que se han impartido en la intensificación de *Sistemas de Información*. Además, se han podido explorar en profundidad aspectos de las tecnologías Web mencionados en las clases teóricas de la carrera. Esto ha sido posible gracias a la colaboración de la empresa donde se ha hecho la estancia en prácticas, la cual ha permitido investigar las nuevas tecnologías y evaluar su posible inclusión en el proyecto a desarrollar.

La experiencia de trabajar en una empresa ha sido muy interesante. Además de haber adquirido conocimientos a nivel técnico, se ha adquirido experiencia en el desarrollo de un proyecto, desde la planificación hasta el desarrollo y despliegue. Esta experiencia ha permitido aprender a trabajar en un equipo de personas con competencias multidisciplinares.

El proyecto desarrollado ha sido motivador por los temas que abarca. Sobre todo, por el uso de tecnologías muy avanzadas y novedosas. La flexibilidad que se ha tenido a la hora de elegir las tecnologías y la manera de implementar las varias funcionalidades ha permitido trabajar a gusto y sin presiones, lo cual ha permitido obtener una mejor calidad del trabajo. Además, se ha aprendido a desarrollar aplicaciones Web de calidad profesional.

El sistema desarrollado es solo el punto de partida para proyectos futuros. El área de investigación que más puede beneficiar de este proyecto es el estudio de la base de datos sobre los estados de las estaciones de Bicicas. Sobre estos datos, se pueden aplicar algoritmos de Minería de Datos para poder encontrar patrones de comportamiento y clases de pertenencias. Además, es una motivación más la posible integración de los servicios Web de Bicicas y Renfe en una plataforma Smart Cities. Por ejemplo, se pueden integrar estos servicios en la plataforma Smart Campus<sup>1</sup> que el grupo de investigación de la UJI Geotec<sup>2</sup> está desarrollando.

---

<sup>1</sup><http://smart.uji.es/>

<sup>2</sup><http://www.geotec.uji.es/>



# Bibliografía

- [1] Agile Manifesto. <http://agilemanifesto.org/>. [Consulta: 24 de Mayo de 2014].
- [2] Microformats. <http://microformats.org/>. [Consulta: 24 de Mayo de 2014].
- [3] Paul Andreson. What is web 2.0? ideas, technologies and implications for education. *JISC Technology and Standards Watch (Tech Watch)*, Febrero 2007.
- [4] ECMA International. ECMA Script. <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>, Junio 2011.
- [5] Roy T. Fielding and Richard N. Taylor. Principled design of a modern web architecture. *ACM Transactions on Internet Technology*, Mayo 2002.
- [6] KHRONOS Group. WebGL. <https://www.khronos.org/registry/webgl/specs/1.0.2/>, Marzo 2013.
- [7] L. Richardson and S.Ruby. Restful Web Services, 2007.
- [8] O. Lassila and J. Hendler. Embracing "web 3.0". *Internet Computing, IEEE*, 11(3):90–93, Mayo 2007.
- [9] W3C. XPath. <http://www.w3.org/TR/1999/REC-xpath-19991116>, Noviembre 1999.
- [10] W3C. XSLT. <http://www.w3.org/TR/1999/REC-xslt-19991116>, Noviembre 1999.
- [11] W3C. RDF. <http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/>, Febrero 2004.
- [12] W3C. XML. <http://www.w3.org/TR/2004/REC-xml11-20040204/>, Febrero 2004.
- [13] W3C. Server-Sent Event. <http://www.w3.org/TR/2009/WD-eventsource-20091029/>, Octubre 2010.
- [14] W3C. RDFa. <http://www.w3.org/TR/2013/REC-rdfa-core-20130822/>, Agosto 2013.
- [15] W3C. HTML 5. <http://www.w3.org/TR/2014/CR-html5-20140429/>, Abril 2014.





## Anexo A

# Respuestas de los servicios Web

En este anexo, se incluyen las respuesta de los servicios Web creados para el proyecto de prácticas. Por cada servicio Web, Bicicas Web Service y Renfe Web Service, se muestra la respuesta a las URL que se han creado para servir información acerca de la fuente de datos asociada.

Las siguientes secciones muestran los documentos XML generados por cada servicio Web.

### A.1. Renfe Web Service

El Listado A.1 muestra el resultado de la petición HTTP `http://inti.init.uji.es:8080/RenfeWebService/ws/timetable/65000/2014/06/29/12/65300/2014/06/29/13/?semantic=h-event`. En ella, se especifica que se quiere obtener el detalle de los trenes entre la estación de Castellón (código 65000) y la estación de Valencia-Nord (código 65300). Se quieren obtener los trenes a partir de las 12:00 del día domingo 29/06/2014 hasta las 13:59 horas del mismo día. Se puede notar como se especifica el tipo de metadatos semánticos que se quieren añadir al documento XML. Por el momento, se permite solo añadir *h-event*.

```
<?xml version="1.0" encoding="UTF-8"?>
<timeTable>
  <departureStationCode>65000</departureStationCode>
  <arrivalStationCode>65300</arrivalStationCode>
  <departureDateTime>2014-06-30T12:00:00+02:00</departureDateTime>
  <arrivalDateTime>2014-06-30T13:59:00+02:00</arrivalDateTime>
  <trips>
    <trip class="h-event">
      <name class="p-name">Train trip from Castellón de la Plana to Valencia Nord</name>
      <category class="p-category">Train trip</category>
      <departureDateTime class="dt-start">2014-06-30T12:20:00+02:00</departureDateTime>
      <arrivalDateTime class="dt-end">2014-06-30T13:29:00+02:00</arrivalDateTime>
      <tripTime class="dt-duration">01:09</tripTime>
      <line>C6 </line>
      <civis>>false</civis>
    </trip>
  </trips>
</timeTable>
```

```

</trip>
<trip class="h-event">
  <name class="p-name">Train trip from Castellón de la Plana to Valencia
    Nord</name>
  <category class="p-category">Train trip</category>
  <departureDateTime class="dt-start">2014-06-30 T12:50:00+02:00</
departureDateTime>
  <arrivalDateTime class="dt-end">2014-06-30 T13:59:00+02:00</arrivalDateTime>
  <tripTime class="dt-duration">01:09</tripTime>
  <line>C6 </line>
  <civis>false</civis>
</trip>
<trip class="h-event">
  <name class="p-name">Train trip from Castellón de la Plana to Valencia
    Nord</name>
  <category class="p-category">Train trip</category>
  <departureDateTime class="dt-start">2014-06-30 T13:20:00+02:00</
departureDateTime>
  <arrivalDateTime class="dt-end">2014-06-30 T14:29:00+02:00</arrivalDateTime>
  <tripTime class="dt-duration">01:09</tripTime>
  <line>C6 </line>
  <civis>false</civis>
</trip>
<trip class="h-event">
  <name class="p-name">Train trip from Castellón de la Plana to Valencia
    Nord</name>
  <category class="p-category">Train trip</category>
  <departureDateTime class="dt-start">2014-06-30 T13:50:00+02:00</
departureDateTime>
  <arrivalDateTime class="dt-end">2014-06-30 T14:59:00+02:00</arrivalDateTime>
  <tripTime class="dt-duration">01:09</tripTime>
  <line>C6 </line>
  <civis>false</civis>
</trip>
</trips>
</timeTable>

```

Listing A.1: XML devuelto por el servicio Web Renfe Web Service

## A.2. Bicicas Web Service

El Listado A.2 muestra la información que se obtiene pidiendo el estado de la estación Bicicas con código 40. El documento se obtiene accediendo a la siguiente URL: <http://inti.init.uji.es:8080/BicicasWebService/ws/bicicas/40>.

En el Listado A.3 se muestra la estación de Bicicas más cercana a las coordenadas geográficas especificadas en la URL de la petición. Las coordenadas se refieren a la Avenida Perez Galdos número 17 y la estación de Bicicas encontrada es la número 2, Tenencia de Alcaldía del Oeste. La URL para obtener esta información es: <http://inti.init.uji.es:8080/BicicasWebService/ws/bicicas/nearest?lat=39.9864399&lon=-0.045919299999999996>

```

<?xml version="1.0" encoding="UTF-8"?>
<bicicasPoint>
  <code>40</code>

```

[illegible]

```

        <bikeCode>789</bikeCode>
    </dock>
    <dock>
        <bikeCode>-1</bikeCode>
    </dock>
    <dock>
        <bikeCode>214</bikeCode>
    </dock>
</docks>
</currentState>
</biciclasPoint>

```

Listing A.2: XML devuelto por el servicio Web Biciclas Web Service con la informacion de la estacion de Biciclas con codigo 40

```

<?xml version="1.0" encoding="UTF-8"?>
<biciclasPoint>
    <code>13</code>
    <name>13.- Tenencia alcaldía oeste</name>
    <location class="h-geo">
        <lat class="p-latitude">39.987175</lat>
        <lon class="p-longitude">-0.047486</lon>
    </location>
    <bicycleNumber>16</bicycleNumber>
    <currentState>
        <updated>2014-06-22T07:08:28.987+02:00</updated>
        <online>true</online>
        <bicycleAvailable>5</bicycleAvailable>
        <ingoing>
            <dock>
                <bikeCode>537</bikeCode>
            </dock>
            <dock>
                <bikeCode>720</bikeCode>
            </dock>
            <dock>
                <bikeCode>251</bikeCode>
            </dock>
            <dock>
                <bikeCode>646</bikeCode>
            </dock>
            <dock>
                <bikeCode>666</bikeCode>
            </dock>
        </ingoing>
        <outgoing />
    </currentState>
    <docks>
        <dock>
            <bikeCode>537</bikeCode>
        </dock>
        <dock>
            <bikeCode>-1</bikeCode>
        </dock>
        <dock>
            <bikeCode>-1</bikeCode>
        </dock>
        <dock>
            <bikeCode>-1</bikeCode>
        </dock>
        <dock>
            <bikeCode>-1</bikeCode>
        </dock>
    </docks>

```

```

    <bikeCode>720</bikeCode>
  </dock>
  <dock>
    <bikeCode>-1</bikeCode>
  </dock>
  <dock>
    <bikeCode>-1</bikeCode>
  </dock>
  <dock>
    <bikeCode>-1</bikeCode>
  </dock>
  <dock>
    <bikeCode>-1</bikeCode>
  </dock>
  <dock>
    <bikeCode>251</bikeCode>
  </dock>
  <dock>
    <bikeCode>646</bikeCode>
  </dock>
  <dock>
    <bikeCode>-1</bikeCode>
  </dock>
  <dock>
    <bikeCode>666</bikeCode>
  </dock>
  <dock>
    <bikeCode>-1</bikeCode>
  </dock>
  <dock>
    <bikeCode>-1</bikeCode>
  </dock>
  <dock>
    <bikeCode>-1</bikeCode>
  </dock>
  <dock>
    <bikeCode>-1</bikeCode>
  </dock>
  </docks>
</currentState>
</bicicasPoint>

```

Listing A.3: XML devuelto por el servicio Web Bicicas Web Service con la informacion de la estacion de Bicicas mas cercana a la Avenida Perez Galods numero 17



## Anexo B

# Tiempo de respuesta de la aplicación

En esta sección se presentan los tests del tiempo de respuesta de los servicios Web creados. Como se muestra en la Sección 4.2, uno de los requisitos del caso de uso es que el tiempo de respuesta sea inferior a 4 segundos.

Para comprobar este requisito, se ha usado una herramienta para el test de cargas en servidores Web. La herramienta usada es Pylot<sup>1</sup>. Los tests se han efectuado con una configuración de 20 agentes haciendo 20 peticiones por segundo a lo largo de 60 segundos. Un agente es un concepto específico de la herramienta usada y simula un usuario real del servicio Web.

Para cada servicio Web desarrollado, Renfe Web Service y Bicicas Web Service, se han lanzado los tests con las URL que se han visto en el Anexo A. El cuadro B.1 muestra un resumen de los resultados de los tests. El primer test *Bicicas Web Service - Nearest Station Test* se refiere al tiempo de respuesta del servicio Web Bicicas Web Service solicitando la estación de Bicicas más cercana a las coordenadas geográficas de usuario. La gráfica que muestra el detalle del test es la Figura B.3. El segundo test tiene en consideración el tiempo que el servicio Web Bicicas Web Service tarda en devolver el estado de una estación de Bicicas especificada por su código. La gráfica con el resultado se muestra en la Figura B.2. El último test comprueba el comportamiento del servicio Web Renfe Web Service. Se pide al servicio los horarios de los trenes entre dos fechas. La Figura B.1 muestra el resultado.

Como se puede notar de los resultados de los tests, el requisito queda satisfecho, en cuando ningún servicio Web tiene un tiempo de respuesta cercano al límite.

Test	Numero peticiones	Media tiempo respuesta (s)
Bicicas Web Service - Nearest Station Test	1042	1.157
Bicicas Web Service - Station Status Test	2904	0.371
Renfe Web Service - Timetable Test	2197	0.548

Cuadro B.1: Cuadro que resume los resultados de los tests efectuados sobre los servicios Renfe Web Service y Bicicas Web Service

---

<sup>1</sup><http://www.pylot.org/>

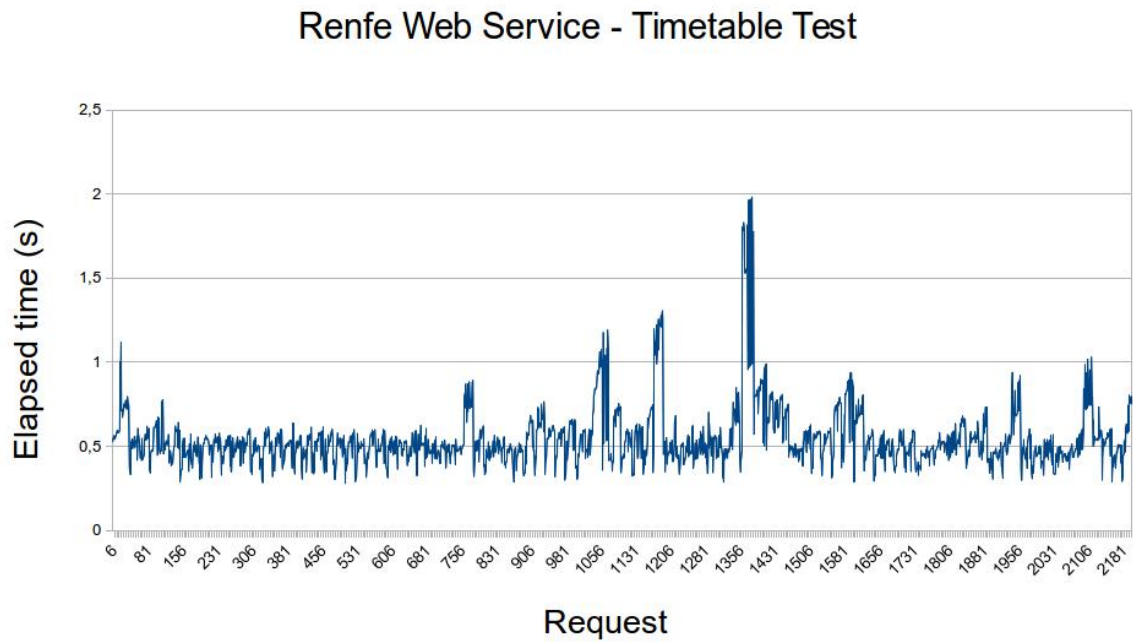


Figura B.1: Gráfico que muestra el resultado del test de tiempo de respuesta del servicio Renfe Web Service

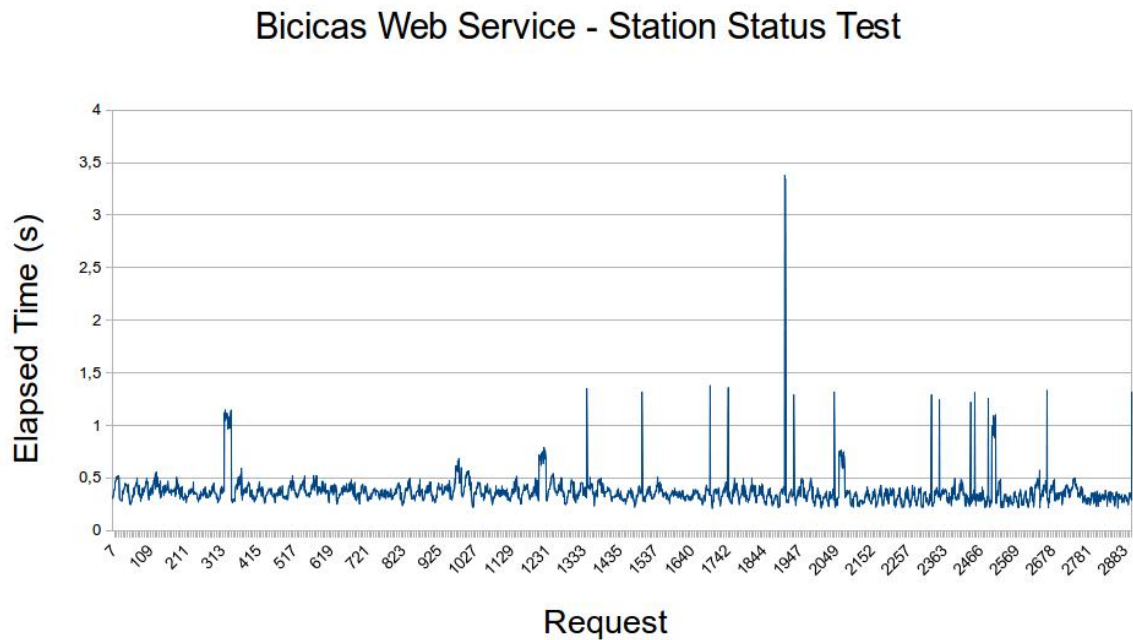


Figura B.2: Gráfico que muestra el resultado del test de tiempo de respuesta del servicio Bicicas Web Service obteniendo el estado de una estación de Bicicas



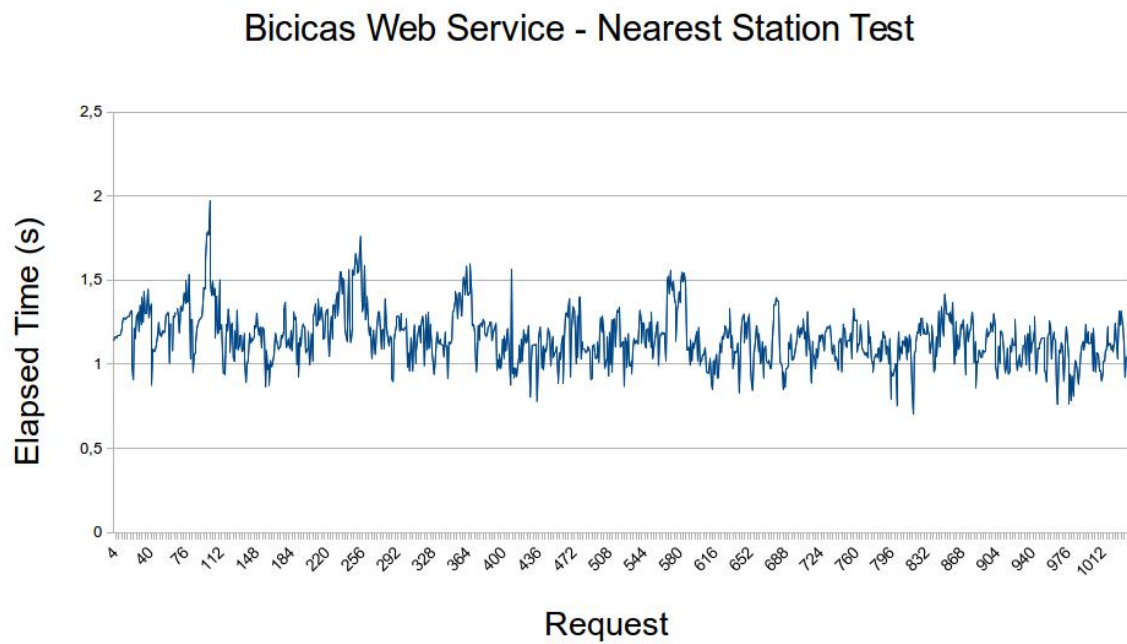


Figura B.3: Gráfico que muestra el resultado del test de tiempo de respuesta del servicio Bicicas Web Service obteniendo sitio de Bicicas más cercano a la posición del usuario