Universidad del País Vasco
Euskal Herriko Unibertsitatea

Facultad de Informática

Grado de Ingeniería Informática

▪ **P**royecto **F**in de **G**rado ▪

Ingeniería del Software

dBizi-bizi: una aplicación móvil multiplataforma y colaborativa para usuarios de dBizi

---

Adrián González Elosegui

Septiembre 2017

informatika fakultatea
facultad de informática

Resumen

Este documento contiene la memoria del proyecto Fin de Grado (TFG) sobre la aplicación dBizi-bizi desarrollada por Adrián González en 2017.

Esta memoria abarca todo lo relacionado con el desarrollo del proyecto, esto es, las tecnologías que se han implementado, la arquitectura escogida, las pruebas realizadas, el control y seguimiento del proyecto, las conclusiones personales y las del proyecto, la bibliografía y el desarrollo del mismo.

dBizi-bizi es una aplicación móvil multiplataforma diseñada para Android, iOS y Windows Phone que trata de facilitar el uso del sistema público dBizi, el servicio de préstamo de bicicletas eléctricas ofrecido en Donostia-San Sebastián.

Esta aplicación nace fruto de una necesidad real, hasta el momento, no existe ningún software que las cubra las necesidades que el sistema dBizi requiere. Por consiguiente, dBizi-bizi trata de cubrir la falta de información respecto al estado de las estaciones y las bicicletas, proporcionar una interacción entre los usuarios del servicio y facilitar su uso.

Al tratarse de una aplicación multiplataforma dBizi-bizi es accesible a un mayor número de usuarios. De esta manera, se extiende su uso y se mejora la experiencia de la aplicación para cada usuario.

Índice

Índice de Figuras

Índice de tablas

1 INTRODUCCIÓN

dBizi-bizi es una aplicación móvil multiplataforma que trata de facilitar el uso diario del sistema público dBizi ofertado en Donostia y añadirle funcionalidades que anteriormente no disponía.

En este capítulo se mostrarán los antecedentes del proyecto y un estudio de mercado tanto de la zona donde va a actuar el producto como de las aplicaciones con funcionalidades similares que tienen su zona de actuación fuera de la ciudad. Por otro lado, también se expone una comparativa de las posibles tecnologías que podrían utilizarse para el desarrollo de la aplicación junto a la elección de la tecnología a implementar y las razones por las que se ha escogido. Por último, se explicará la metodología ágil que se ha escogido para el desarrollo del proyecto: Scrum.

1.1 LICENCIA

1.2 ANTECEDENTES

Los Smartphone han supuesto una revolución en la vida cotidiana de la gente pues la mayor parte de ella dispone de uno y hace uso de él de manera asidua mayoritariamente. Éstos han sufrido muchos cambios y avances, razón por la cual tienen la capacidad de aportar una gran cantidad de funcionalidades que pueden variar de un simple reloj a un complejo dispositivo para videojuegos. Sin embargo, dos de las funcionalidades más comunes entre los usuarios de estos dispositivos son, por un lado, la búsqueda de información y, por otro, la comunicación, a través de voz o de texto. Todo ello sin olvidar en ningún momento la búsqueda de, comodidad y facilidad. Es decir, se aspira a obtener dichas funcionalidades de la forma más sencilla y rápida posible.

dBizi-bizi pretende aunar tanto esas dos funcionalidades como esa características en un ámbito concreto mientras ofrece a los usuarios información sobre un sistema público, dBizi, y les otorga la oportunidad de compartir su experiencia con el resto de usuarios.

1.3 ESTUDIO DE MERCADO

Para analizar la viabilidad del proyecto y obtener información sobre las funcionalidades ya existentes pero mejorables, se ha realizado un estudio de mercado tanto de las aplicaciones similares que actúen en Donostia como de las aplicaciones que no ejerzan en la ciudad pero las cuales tengan un objetivo similar y están orientadas a ofrecer el mismo tipo de servicio.

1.3.1 Aplicaciones en Donostia

1.3.1.1 dBizi

La única opción disponible con unas características similares es la aplicación dBizi [1] que se encuentra en Google Play Store. Sin embargo, esta aplicación recibió su última actualización en julio de 2012 y los datos que muestra no son correctos. En esta aplicación se muestra un mapa en el que se ubica las estaciones de las bicicletas, un listado con las estaciones

existentes en el último momento de su actualización y un panel de preferencias en el que se puede alternar el idioma.

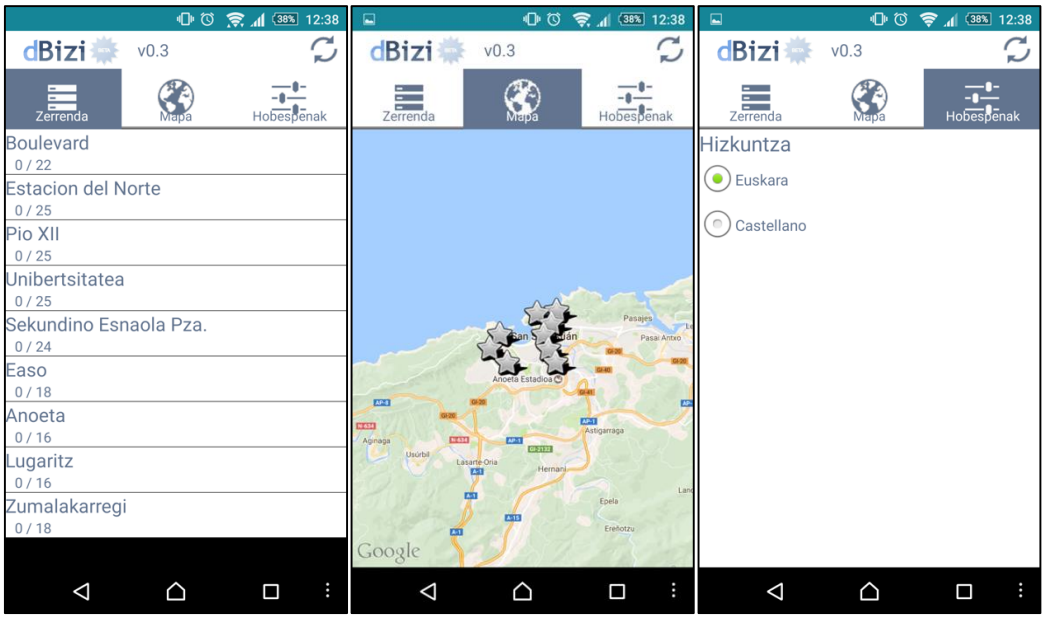


Figura 1.1: dBizi

1.3.2 Aplicaciones fuera de Donostia

1.3.2.1 BiciMad

BiciMad [2] es la aplicación oficial del servicio público de bicicletas de alquiler de Madrid. Este servicio está gestionado por la misma empresa a cargo del servicio de Donostia, Bonopark S.L.. Esta aplicación tiene una valoración muy negativa debido a su rendimiento a pesar de poseer funcionalidades e interfaz atractivas para el usuario. Entre las funcionalidades encontramos:

- Un mapa con las rutas y estaciones.
- Un pronóstico meteorológico.
- Un historial de recorridos.
- Acceso al perfil de usuario.
- Varios datos sobre calorías, emisiones CO_2 y KM recorridos.

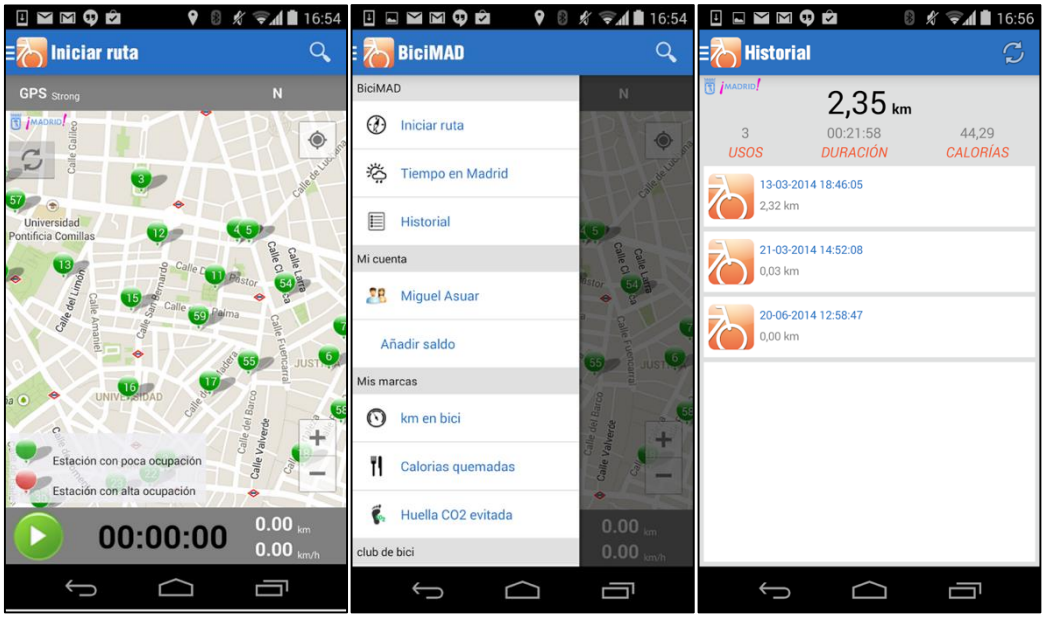


Figura 1.2: BiciMad

1.3.2.2 Cicleta – info BiciMAD

Cicleta [3] es una aplicación que muestra información del sistema público de Madrid. Al no ser oficial posee menos funcionalidades que BiciMAD. Sin embargo, cuenta con una mejor valoración de los usuarios y una interfaz más sencilla e intuitiva. Entre sus funcionalidades se encuentran:

- Un mapa con las rutas y estaciones.
- Marcar estaciones como favoritas.
- Búsqueda de cualquier dirección en el mapa y posibilidad de encontrar la estación más cercana.

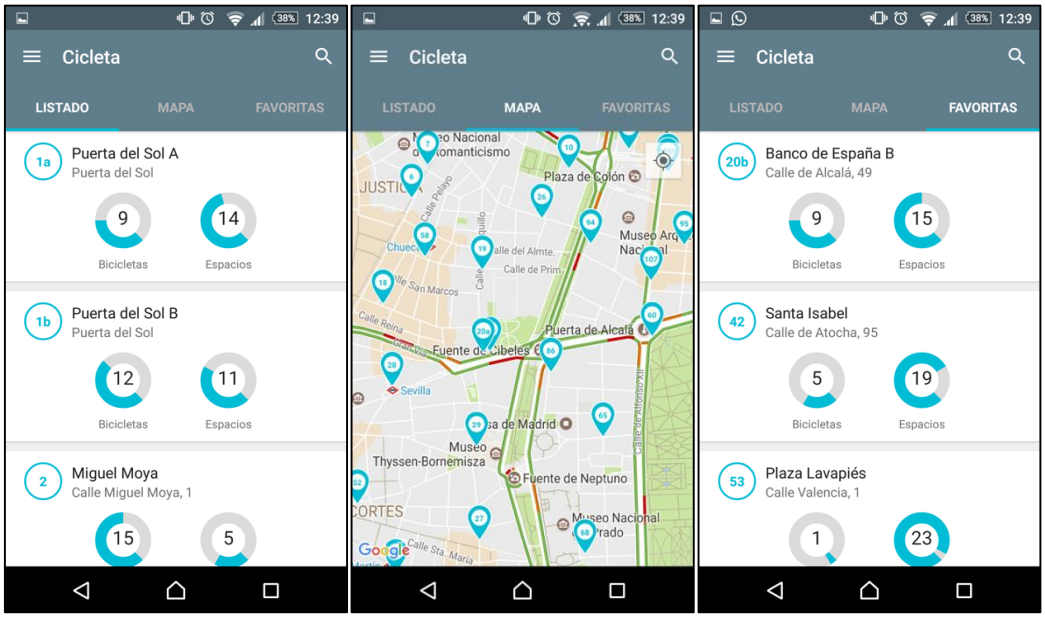


Figura 1.3: Cicleta

1.3.2.3 MADbike

MADBike [4], al igual que la anterior, se trata de una aplicación no oficial del sistema público BiciMAD. Fue ganadora de una subvención de cuarenta mil euros por parte de FB Start. Cuenta con una gran variedad de funcionalidades y un mapa interactivo práctico y original en el que las estaciones se van agrupando conforme alejas el zoom del mapa a fin de que el mapa no se sature con los marcadores. Entre sus funcionalidades se encuentran:

- Mapa interactivo con las estaciones de dBizi.
- La calidad del aire en Madrid obtenida desde un portal de datos abiertos (datos.madrid.es).
- Noticias publicadas en Twitter con el hashtag #bicimad o #MADBike.
- Capacidad para enviar propuestas, informar de incidencias y contactar con los desarrolladores de la aplicación.

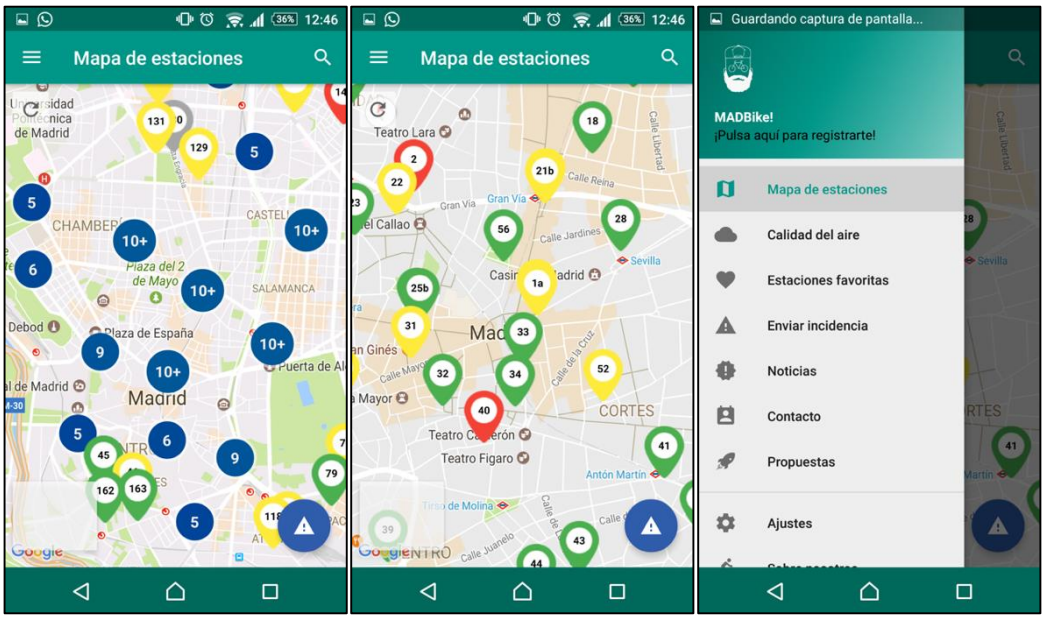


Figura 1.4: MADbike

1.4 MOTIVACIÓN

Una vez realizado el estudio de mercado, se puede observar que no existe una solución real que proporcione a los usuarios de dBizi una mejor experiencia del sistema. Por ello, se decide desarrollar una nueva aplicación que sea capaz de ofrecer las funcionalidades que no eran accesibles para los usuarios de dBizi y poder así cubrir las necesidades que demandaban.

De la misma manera, el desarrollo de esta aplicación móvil capaz de tener un impacto directo en la gente puede tratarse de una buena oportunidad para poner en práctica lo aprendido a lo largo del grado. Igualmente, posibilita la opción de especializarse en un área que suscita especial interés personal por las posibilidades futuras que pueden llegar a ofrecer al alumno. Asimismo, se trata de una buena oportunidad para poder enfrentarse de manera individual

y competente a las incidencias que pueden surgir durante el desarrollo completo de un proyecto.

1.5 APP NATIVA, WEB O HÍBRIDA

Actualmente, existen tres tipos distintos de aplicaciones que pueden desarrollarse para los dispositivos móviles. En primer lugar, se encuentran las aplicaciones nativas. Estas se desarrollan directamente en el lenguaje de programación de cada dispositivo, que en el caso de los más actuales se trataría de Java para android, Objective-C para iOS y .NET para Windows Phone. En segundo lugar, se hallan las aplicaciones web, también denominadas webapp, que se desarrollan en los lenguajes HTML (HyperText Markup Language), CSS (Cascading Style Sheets) y JavaScript. Estas se ejecutan dentro del propio navegador web del dispositivo a través de una URL. Por último, están las aplicaciones híbridas que son una combinación de las dos anteriores. Se desarrollan en los mismos lenguajes que las aplicaciones web, lo cual permite su uso en diferentes plataformas y posibilita el acceso a la mayoría de las características del hardware gracias a *frameworks* específicamente diseñados para facilitar esa tarea.

1.5.1 Ventajas e inconvenientes

El desarrollo de una aplicación nativa ofrece numerosas ventajas en cuanto al rendimiento, un control total sobre el dispositivo y la opción de utilizarla sin tener un acceso a la red. Sin embargo, requiere del conocimiento de diferentes lenguajes de programación debido a que el código no es reutilizable entre las diferentes plataformas, lo cual incrementa mucho el coste de desarrollo de la aplicación.

Esto último no sucede en una aplicación web, pues se desarrolla independientemente del sistema operativo en el que se usará la aplicación. No obstante, su acceso a las características hardware del dispositivo es muy limitado, casi nulo, y requiere de una conexión a Internet para su inicialización al menos.

Por último, las aplicaciones híbridas son adaptables a casi cualquier plataforma, permitiendo de esta manera el acceso a un gran número de usuarios. Estas aplicaciones mantienen el mismo código base entre las distintas plataformas por lo que se reduce el tiempo de desarrollo y permite, a su vez, el acceso a distintos recursos del dispositivo. Además, no requiere de ningún tipo de conexión a la red.

1.6 TECNOLOGÍAS

1.6.1 Tecnologías del lado del cliente

Debido a los objetivos que quieren alcanzarse en este proyecto queda descartada tanto la aplicación web, por falta de control sobre el dispositivo, como la aplicación nativa, por falta de tiempo para el estudio y desarrollo de las tecnologías necesarias. Por lo tanto la opción escogida es hacer una aplicación híbrida.

Existen varios *frameworks* con los que se pueden desarrollar el proyecto. Debido a esto último y a mi propia inexperiencia en el desarrollo de aplicaciones para dispositivos móviles mediante estas tecnologías se ha realizado un estudio de las más utilizadas actualmente en este campo con el objetivo de escoger la que más se adecue a las características del proyecto teniendo en cuenta las limitaciones de tiempo y recursos propias de un TFG.

1.6.1.1 Jquery Mobile

Este *framework* [5] es un proyecto de la fundación jQuery [6]. Es de código abierto y una de las opciones más populares tanto por la sencillez de su aprendizaje como por la ideología *write less, do more* que también está presente en su producto más famoso, la librería de JavaScript jQuery.

Una de las principales particularidades de Jquery Mobile es que la codificación de todas las páginas de la interfaz de usuario se realiza en un único fichero. De este modo, cada página de la aplicación está dentro de una capa de HTML y la transición entre ellas se realiza mediante una consulta AJAX [7] que puede ser invocada tanto mediante eventos de pulsación como de los distintos *widgets* [8] disponibles.

1.6.1.2 Ionic / Ionic 2 [9]

Hoy en día, esta es la versión más utilizada para crear aplicaciones móvil híbridas con Cordova [10] como base. Utiliza Angular JS para gestionar las aplicaciones, lo cual asegura que sean escalables puesto que esta tecnología sigue un patrón Modelo-Vista-Controlador (MVC) [11]. Actualmente, se encuentra en una fase de transición a la versión dos que incluye mejoras sustanciales respecto a su antecesora. Sin embargo, esta nueva versión se encuentra en su fase alfa y, por lo tanto su estabilidad no está garantizada. Posee una herramienta llamada Ionic Creator [12] que permite crear la interfaz mediante el método de arrastrar y soltar. Mientras Ionic soporta dispositivos con Android 4.1+ o iOS 7+, Ionic2 soporta además Windows Phone 10+.

1.6.1.3 PhoneGap

PhoneGap es un *framework* para el desarrollo de aplicaciones móviles producido por Nitobi [13] y comprado posteriormente por Adobe Systems [14].Al igual que Ionic, PhoneGap [15] es una distribución de Apache Cordova y tiene un largo recorrido en el mercado puesto que su lanzamiento inicial es del 2008 y cuenta además con una gran comunidad detrás.

Las API de PhoneGap permiten el acceso a elementos del dispositivo como puede ser el caso del acelerómetro, el GPS, la cámara, las notificaciones... Asimismo, debido a la tecnología en la que se desarrolla PhoneGap, se admiten la inclusión de otras librerías como jQuery, por ejemplo.

Además, desde la adquisición de PhoneGap por parte de Adobe Systems existe un servicio de compilación en la nube proporcionado por Adobe Creative Cloud que facilita la compilación y el firmado de la aplicación.

1.6.2 Tecnologías del lado del servidor

Al igual que en el lado del cliente, para el *back-end* del proyecto se han estudiado diversos *frameworks,* todos ellos basados en el lenguaje PHP [16] puesto que es uno de los más extendidos. Aunque existe un amplio abanico de posibilidades este estudio se ha centrado solo en los principales y más importantes *frameworks* del mercado.

1.6.2.1 CodeIgniter

Al igual que el resto de *frameworks,* CodeIgniter se basa en la arquitectura MVC. Sin embargo, la principal característica de este es su sencillez, tanto a la hora de instalarlo como de usarlo. En caso de tener algún obstáculo existen múltiples ejemplos en su documentación oficial y abundante información en la red.

El núcleo de CodeIgniter es muy ligero lo que permite que las páginas y las peticiones se carguen rápidamente. Además, al ser menos rígido que otras soluciones aporta una mayor flexibilidad. Es decir, define una manera de trabajar que puede ser modificada en caso de así necesitarlo. Funciona bajo la licencia MIT y su versión estable actual es la 3.1.5.

1.6.2.2 Symfony

Uno de los *frameworks* más importantes del mundo es Symfony. Salió en octubre de 2005 por lo que lleva varios años de desarrollo. El objetivo principal de Symfony es el de crear aplicaciones robustas en un contexto empresarial porque les da control total a los desarrolladores sobre la configuración, la estructura de los ficheros… Sin embargo, debido a la gran cantidad de opciones personalizables que ofrece Symfony la curva de aprendizaje se vuelve más lenta que la de otros *frameworks.* Aun así, cuenta con una buena documentación, tanto oficial como extraoficial, y una gran comunidad dispuesta a ayudar. Symfony funciona bajo la licencia MIT y su versión estable actual, la 3.2, salió en noviembre de 2016.Está previsto que termine su ciclo de mantenimiento para julio de 2017.

1.6.2.3 Laravel

Laravel es un *framework* relativamente nuevo (salió en 2011), es gratuito y de código abierto. Fue creado para el desarrollo de aplicaciones web siguiendo la arquitectura MVC.

Sus creadores lo definen como un *framework* con una curva de aprendizaje baja en comparación con los otros disponibles. Además, posee una abundante fuente de información en la documentación oficial. También, tiene su propia plataforma de *hosting* y despliegue junto con un sistema propio de plantillas llamado Blade, el cual facilita tareas cotidianas como la autenticación, gestión de sesiones... Funciona bajo la licencia MIT [17] y su actual versión más estable es la 5.4 que salió en enero de 2017.

1.6.2.4 Zend

Zend es un estable y robusto *framework* de PHP con una gran cantidad de opciones de configuración. Por lo tanto, no es recomendable para proyectos pequeños, pero es una muy buena opción para los más complejos.

Posee unas características muy interesantes, como es el caso de un editor de arrastrar y soltar y herramientas criptográficas entre otras. Su versión actual soporta PHP 7.0. Funciona bajo la licencia *new BSD license* [18] y su actual versión más estable es la 3.0.

1.7 ELECCIÓN DE TECNOLOGÍAS

Con el estudio de tecnologías realizado y teniendo claros tanto los objetivos como el tiempo disponible para realizar el proyecto se han tomado las decisiones que se explicarán a continuación.

Para el lado del cliente o *front-end* la elección es PhoneGap tanto por la documentación oficial como por la información existente en la red. Ambas razones hacen que este *framework* sea una elección consistente y segura a la hora de realizar el proyecto. También se puede contar con su estabilidad puesto que lleva muchos años de desarrollo y tiene, además, numerosos *pluggins* creados por la comunidad y por la empresa desarrolladora del proyecto.

Otra de las razones por las que se ha tomado dicha decisión es la facilidad que proporciona Adobe Creative Cloud, en este caso, para compilar y firmar las aplicaciones creadas con este *framework* con el servicio en la nube PhoneGap Build.

Por último, para el *back-end* la elección es CodeIgniter debido a su sencillez y a la flexibilidad que ofrece al seguir una arquitectura MVC. Aun así, aunque respecto al *back-end* cualquier otra selección hubiera cumplido los mismos objetivos CodeIgniter agilizara el desarrollo de la parte del servidor gracias a su rápida curva de aprendizaje.

1.8 METODOLOGÍA

Para el desarrollo de este proyecto se ha decidido seguir una metodología ágil, Scrum [20] en concreto. Aunque esta metodología este orientada a equipos de trabajo y requiera de un cliente se ha estimado que las ventajas de trabajar con esta metodología supera a los inconvenientes. Estos últimos han sido solucionados de la manera que se explica a continuación.

1.8.1 Scrum

Se trata de una metodología ágil en la que se aplican un conjunto de buenas prácticas para trabajar en equipo y obtener el mejor resultado posible en un proyecto.

En Scrum se realizan entregas parciales y regulares del producto, a poder ser en un intervalo no superior a tres o cuatro semanas. De esta manera, se aporta de forma tangible al cliente una visión de la evolución y el estado del proyecto. Esta metodología está especialmente indicada para proyectos que necesitan obtener resultados pronto, donde los requisitos son cambiantes o están poco definidos y donde la flexibilidad y la productividad son fundamentales.

Scrum se ejecuta en bloques temporales cortos y fijos que varían, normalmente, de dos a cuatro semanas. Estos bloques temporales se denominan Sprints [21] o iteraciones y cada uno de ellos ha de proporcionar un resultado completo. Esto es, un incremento del producto que es susceptible a ser entregado al cliente.

El proceso de desarrollo de un Sprint consta de tres partes: planificación, ejecución e inspección. Asimismo, parte de una lista de objetivos (Product Backlog) que el cliente (Product Owner) prioriza en función del valor que estos añaden al proyecto y el coste que supone realizarlos.

En el primer día de la iteración se realiza una reunión con el cliente donde este presenta al equipo la lista de objetivos. También, se resuelven las posibles dudas que pueda haber por cualquiera de las dos partes y el equipo selecciona los requisitos más importantes y los que se compromete a completar en dicho Sprint. Después, el equipo elabora la lista de tareas que serán necesarias para desarrollar lo pactado con el cliente y se asignan dichas tareas.

Durante la ejecución del Sprint, el equipo realiza cada día una reunión de sincronización de aproximadamente quince minutos. Cada miembro inspecciona el trabajo que el resto de sus compañeros está realizando para así poder resolver cuestiones como qué se ha hecho desde la última reunión de sincronización, qué va a hacer cada uno a partir de ese momento o qué impedimentos u obstáculos van a poder surgir.

Durante este periodo el Scrum Master es el encargado de que el equipo pueda cumplir con los objetivos asignados eliminando obstáculos que el equipo no puede resolver por sí mismo o protegiendo y aislando al equipo de interrupciones externas que pudiesen mermar su productividad.

El último día del Sprint se realiza una reunión de revisión que consta de dos partes: la demostración y la retrospectiva. En la demostración el equipo presenta al cliente los requisitos completados durante la iteración. Por el contrario, en la parte retrospectiva el equipo analiza cómo ha sido la manera de trabajar, los problemas que se han presentado y qué mejoras podrían realizarse. Para ello, el Scrum Master será el encargado de eliminar los obstáculos identificados.

1.8.2 Implantación de Scrum en dBizi

Aunque las metodologías ágiles, y en concreto Scrum, están diseñadas para equipos de trabajo por tratarse de un proyecto individual la asignación de los roles especificados anteriormente se ajustarán a la logística de un Trabajo de Fin de Grado. De esta manera, el director del proyecto será el Product Owner y yo ejerceré los roles del Scrum Master y del del Scrum Team.

Por las mismas razones, se omitirán las reuniones de sincronización porque el conjunto global del equipo es identificado como una sola persona. De esta manera, se agiliza el proceso al hacer una sola reunión para la demostración de un Sprint y para planificar el siguiente Sprint, siempre y cuando el Product Owner quede satisfecho con los resultados mostrados en dicha reunión.

1.8.3 Diagramas UML

Aunque el proyecto vaya a ser desarrollado con una metodología ágil para una mejor visualización de las funcionalidades implementadas y del sistema en general, se crearán varios diagramas de casos de uso junto a su correspondiente flujo de eventos.

2 PLANIFICACIÓN

En este capítulo se expondrán los objetivos que forman parte del desarrollo de la aplicación donde, primero, se establece el alcance del proyecto junto con sus exclusiones para que no den lugar a ambigüedades. También, se realiza una planificación inicial de las tareas que se van a realizar detallando su comienzo, duración y final.

Después, se presenta un diagrama EDT que muestra los paquetes de trabajo en los que se ha descompuesto el proyecto. De igual manera, se presenta un diagrama Gantt con los tiempos de dichos paquetes de trabajo.

Por último, se exponen también los riesgos que pueden alterar los tiempos de desarrollo del proyecto junto a un plan de contingencia para evitar, en la medida de lo posible, alterar el desarrollo del proyecto.

2.1 OBJETIVOS

dBizi-bizi será una aplicación que proporcionará a sus usuarios información precisa y en tiempo real mediante la inclusión de un mapa del estado de las estaciones repartidas por San Sebastián además de un *feedback* comunitario del estado de las bicicletas. Mediante esta aplicación se pretende dar la posibilidad a la gente de consultar toda esta información de una forma sencilla y cómoda al mismo tiempo que mejora su experiencia con el servicio público dBizi. Contará, además, con un chat en el que los usuarios podrán intercambiar opiniones y hablar entre ellos.

Asimismo, será una aplicación multiplataforma que podrá lanzarse en dispositivos con sistema operativo Android, iOS o Windows Phone. De esta manera, se podrá cubrir casi todo el espectro de potenciales usuarios de la aplicación.

2.2 ALCANCE

El alcance de este producto incluye tanto la aplicación *front-end*, desarrollada en PhoneGap, como el *back-end*, desarrollado en CodeIgniter. Se incluye, también, la configuración y el alojamiento de la parte *back-end* en algún servicio de *hosting*. De igual modo, se deben cumplir los siguientes para conseguir un producto de calidad:

1. La aplicación debe ser intuitiva. Dado que el público que hace uso del sistema dBizi es muy heterogéneo y debe poder ser utilizada por todos.
2. La aplicación ha de ser rápida. No debe sobrecargarse con un uso excesivo de librerías de terceros u otros factores que mermen su velocidad puesto que la rapidez de carga es un factor importante en esta aplicación.
3. La aplicación debe llegar a ser accesible a todo el mundo. Por lo tanto, se utilizará alguna plataforma como Google Play Store para su distribución.

2.3 EXCLUSIONES

Debido al coste adicional de tiempo se excluye de este proyecto la traducción de la aplicación a varios idiomas. También queda excluida la implementación de una seguridad exhaustiva tanto en el lado del cliente como en el lado del servidor y por último, quedan excluidas la realización de pruebas en *tablets* y otros dispositivos similares. Tampoco se desarrollará la aplicación en otras plataformas que no sean las citadas en los objetivos.

2.4 WBS / EDT

La estructura de descomposición del trabajo (EDT) [22] consiste en la descomposición jerarquizada del proyecto para cumplir con los objetivos del mismo. La descomposición de este proyecto se ha efectuado en cuatro grandes secciones:

1. Gestión: Este paquete de trabajo es el que contiene el análisis, seguimiento y control del proyecto.
2. Desarrollo: Comprende el aprendizaje de las nuevas tecnologías, el desarrollo del proyecto y las pruebas correspondientes.
3. Documentación: Todo lo relacionado con la documentación del proyecto, incluida la creación de algunas lecciones aprendidas para futuros proyectos.
4. Defensa del proyecto: Engloba tanto la realización como la preparación de la misma.

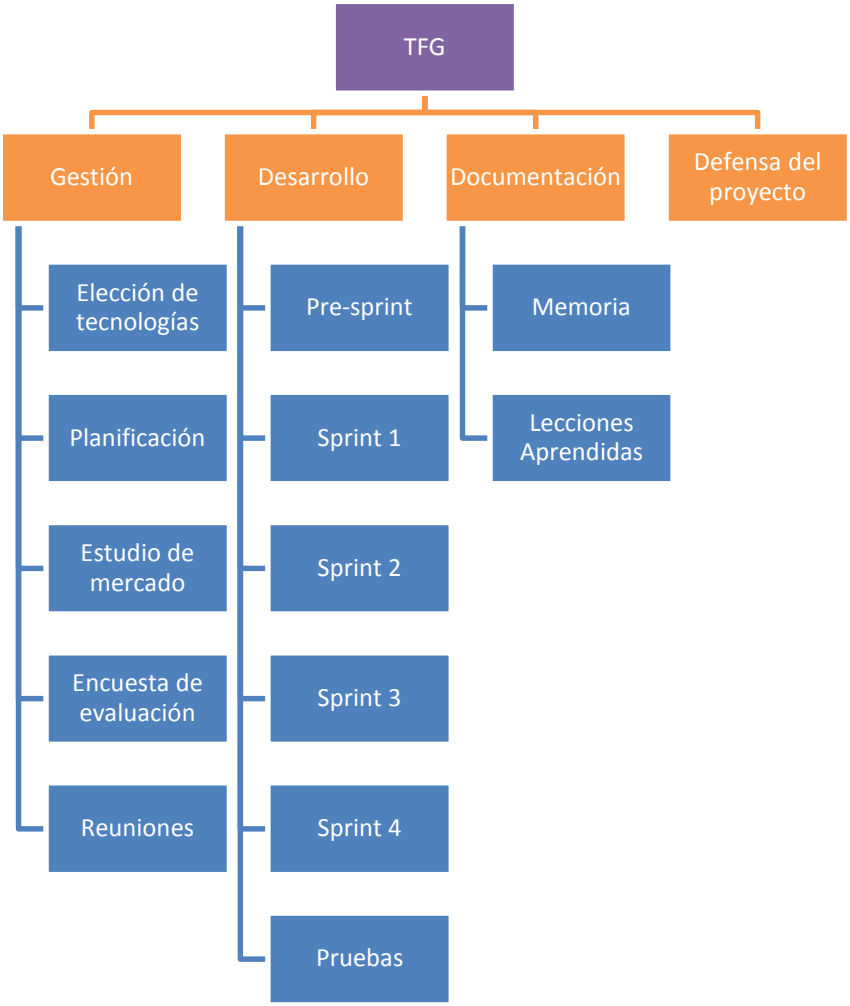


Figura 2.1: Esquema EDT

2.5 GANTT

El diagrama Gantt es especialmente significativo para observar el tiempo de dedicación previsto para cada uno de los paquetes de trabajo impuestos en el EDT. Por ello, para su creación se ha utilizado la herramienta online Tom's Planner [23] dado que es intuitiva y gratuita.

A continuación se muestra el diagrama Gantt de todo el proyecto para obtener una visión global del mismo. Después, se exponen varias imágenes de cada paquete de trabajo.

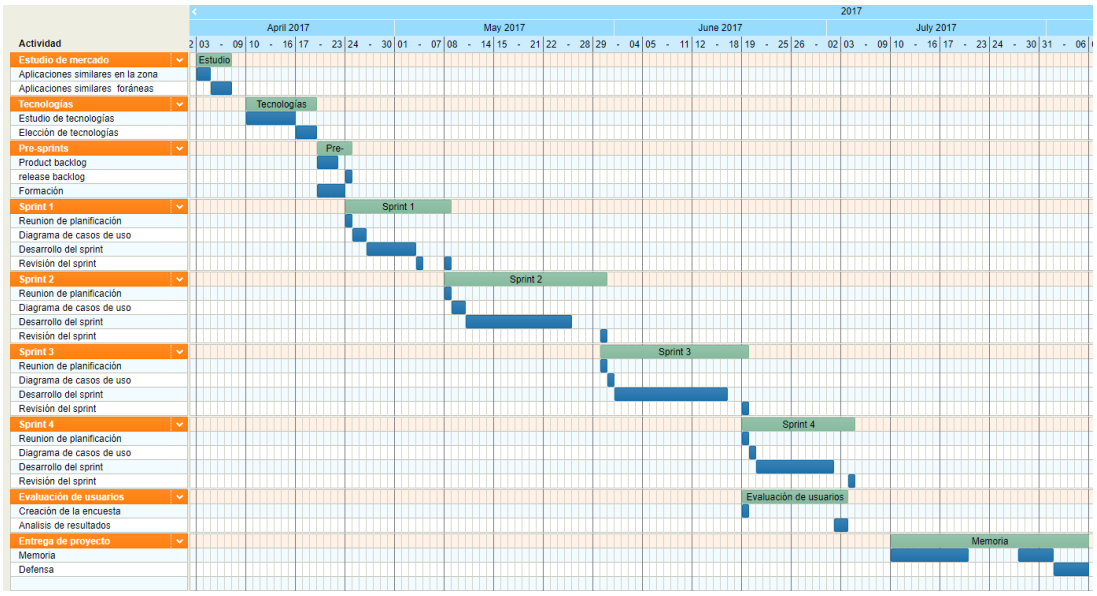


Figura 2.2: Diagrama Gantt completo

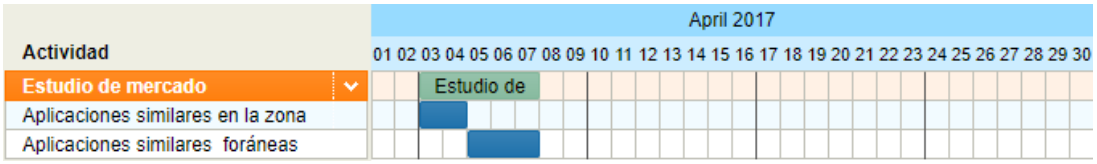


Figura 2.3: Diagrama Gantt - estudio de mercado

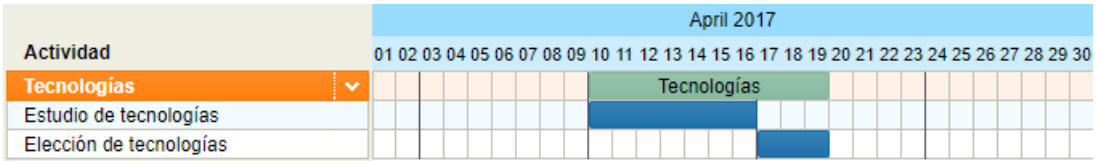


Figura 2.4: Diagrama de Gantt - Tecnologías

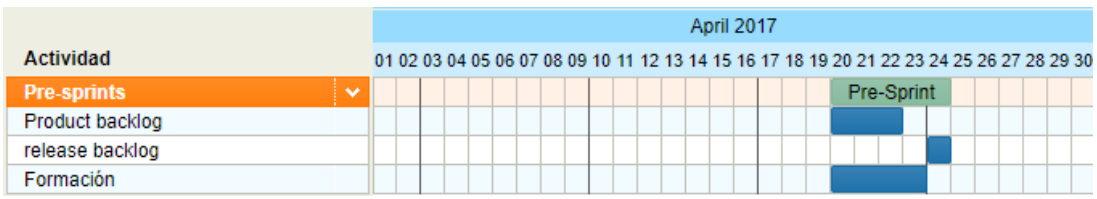


Figura 2.5: Diagrama de Gantt - Pre-Sprint

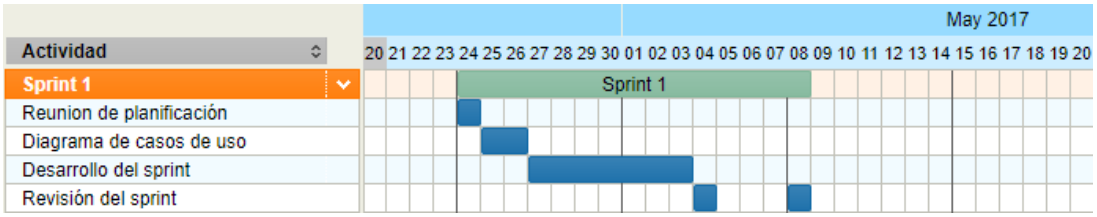


Figura 2.6: Diagrama de Gantt - Sprint 1

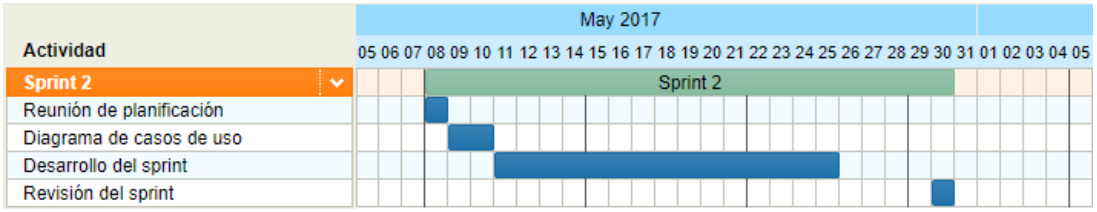


Figura 2.7: Diagrama de Gantt - Sprint 2

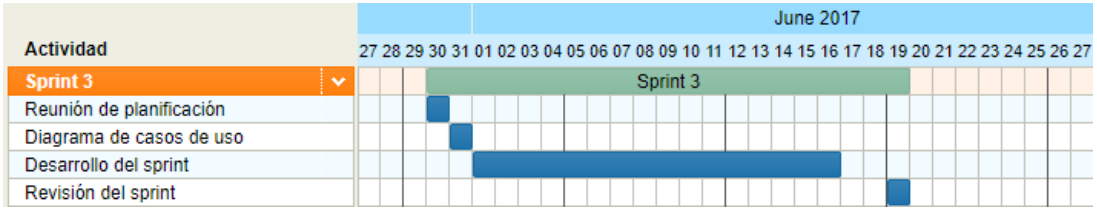


Figura 2.8: Diagrama de Gantt - Sprint 3

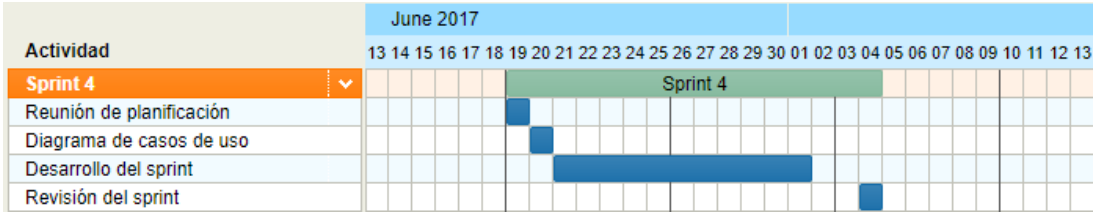


Figura 2.9: Diagrama de Gantt - Sprint 4

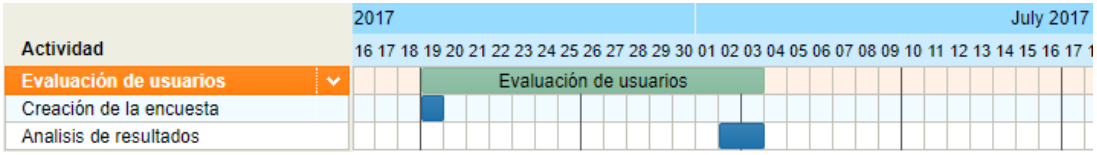


Figura 2.10: Diagrama de Gantt - Evaluación de usuarios

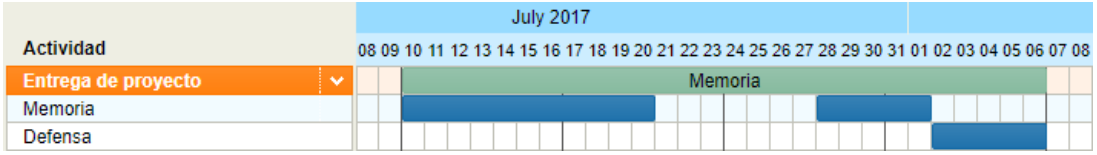


Figura 2.11: Diagrama de Gantt - Entrega de proyecto

2.6 RIESGOS

Existen varios factores de riesgo que han de tenerse en cuenta y que pueden hacer que el desarrollo sufra aplazamientos. Por ello, aparte de identificar dichos factores se establecerá un plan de contingencia para minimizar el riesgo de los mismos. Entre los factores de riesgo identificados están:

1. Problemas personales o de salud pueden hacer que deba pausar el desarrollo del proyecto.
2. Disponibilidad del director y cliente del proyecto a la hora de establecer alguna reunión.
3. Una no optimizada planificación puede hacer que se acumule trabajo o que algún trabajo dependa de otro todavía no realizado.
4. El uso de tecnologías desconocidas o características del producto que no se adecúen a lo pactado con el Product Owner puede llevar a que se produzcan aplazamientos.
5. Pérdida de la información almacenada localmente.

Para cada factor de riesgo identificado anteriormente se han establecido diferentes pautas que formaran el plan de contingencia tal y como se exponen a continuación respectivamente.

1. Este problema paralizará el desarrollo, pero para minimizar el impacto se establecerá unos plazos de entrega con un margen considerable de tiempo que contemple esta posibilidad.
2. En cada reunión se le preguntará al director sobre su disponibilidad para la siguiente reunión y se establecerá dicha reunión con varios días de antelación.
3. En caso de detectar un error o retraso en alguna parte de la planificación del proyecto se intentará identificar cuanto antes y se procederá a planificarlo nuevamente de manera inmediata.
4. En caso de detectar alguna funcionalidad que pudiese generar problemas se procederá a la investigación de su viabilidad antes de su desarrollo.
5. Aunque el proyecto sigue un control de versiones en la plataforma GitHub, los documentos se subirán a la nube, en concreto a Google Drive. De esta manera, se evitarán pérdidas y se tendrá acceso desde cualquier sitio siempre y cuando me identifique como tal en dicha plataforma.

3 REQUISITOS

En este capítulo se detallan los requisitos que componen el desarrollo de la aplicación dBizi-bizi. Se explicarán tanto las historias de usuario desarrolladas en cada Sprint como sus criterios de aceptación. Además, aunque no pertenezca explícitamente a Scrum, también se expondrán los casos de uso con su correspondiente flujo de eventos a fin de dar una mejor visión global del producto y su desarrollo.

3.1 PRODUCT BACKLOG E HISTORIAS DE USUARIO

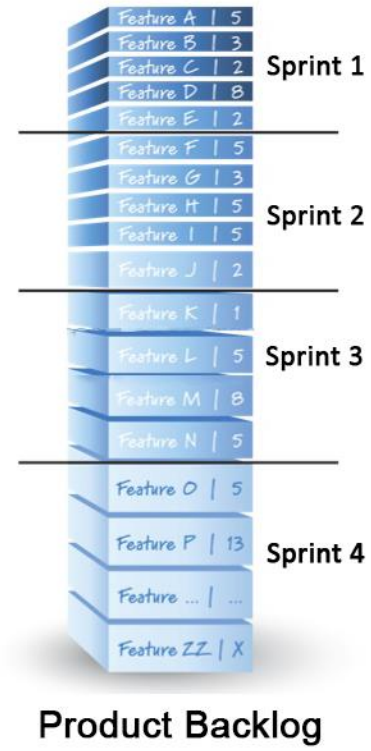


Figura 3.1: Product Backlog

El Product Backlog es el conjunto de historias de usuario [24] que debe cumplir el producto al final de su desarrollo. Es el Product Owner el encargado de priorizar dicho Backlog en función del riesgo, la utilidad y el tiempo de desarrollo de cada funcionalidad. Estas historias de usuario se escogen en cada Sprint en los que vayan a ser desarrolladas.

Una historia de usuario es una descripción de una funcionalidad que debe incorporar la aplicación y cuya implementación aporta valor al cliente. La creación de estas historias se realizará siguiendo una pequeña variación del ejemplo que podemos encontrar en el PMBOK *online* de Scrum disponible en la referencia anterior.

3.2 SPRINTS

Un proyecto realizado con la metodología Scrum se ejecuta en bloques temporales cortos y fijos llamados Sprint. Cada uno de estos Sprints ha de proporcionar un resultado completo y potencialmente entregable al cliente (Product Owner).

Al inicio de cada iteración el cliente presenta al equipo una lista con los requisitos priorizados del proyecto. Después, el Scrum Team elabora una lista de tareas de la iteración necesaria para desarrollar los requisitos que se han comprometido con el cliente.

Aunque lo normal sería hacer una reunión diaria en la que cada integrante del equipo inspeccionase el avance del resto de integrantes, en este caso, no se realizarán por ser una única persona el integrante de dicho equipo.

Por último, al final de cada iteración se realizará una reunión con el cliente en la que se hará una demostración del resultado obtenido a lo largo de dicha iteración. En la misma reunión, se establecerán los requisitos de la siguiente iteración.

3.2.1 Sprint uno

Historias de usuario

A continuación se muestran las historias de usuario que forman parte del primer Sprint:

	Historia de usuario
Número: 1	**Usuario:** Cliente
Nombre historia: Ver las estaciones de bici	
Sprint Asignado: 1	
Descripción: Quiero poder ver las estaciones de bici en un mapa y mi situación geográfica.	
Validación: El cliente podrá visualizar en la aplicación de Android un mapa que marque localización de las estaciones y la suya propia.	

Tabla 3.1: Historia de usuario 1

	Historia de usuario
Número: 2	**Usuario:** Cliente
Nombre historia: Menú en la aplicación	
Sprint Asignado: 1	
Descripción: Quiero disponer de un menú en la aplicación mediante el cual poder acceder a las distintas partes de la aplicación.	
Validación: Se dispondrá de un menú lateral estilo Facebook que pueda abrirse y cerrarse desde cualquier parte de la aplicación y que dará acceso rápido a las partes más importantes de la misma.	

Tabla 3.2 Historia de usuario 2

Sprint Backlog

Tarea	Estado	Horas pendientes	Horas dedicadas
Implementar mapa	Completada	-	5
Implementar menú	Completada	-	3
Mostrar estaciones e información	Completada	-	6

Tabla 3.3 Sprint Backlog 1

Casos de uso

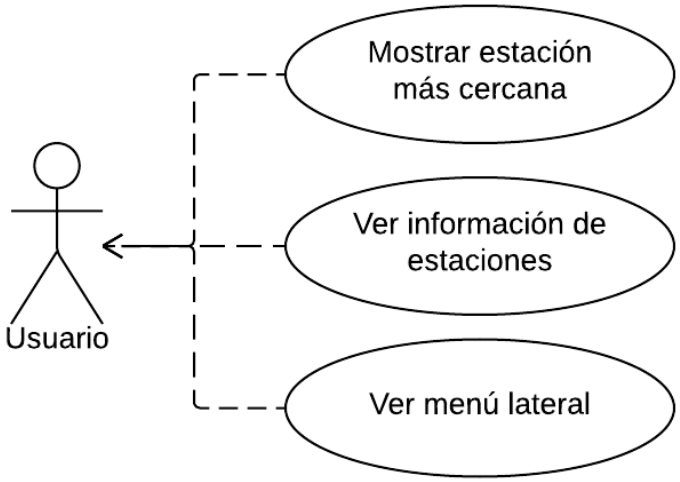


Figura 3.2: Casos de uso Sprint 1

Flujo de eventos

Caso de uso	Flujo de eventos
Ver mapa con estaciones	-La aplicación muestra en su página inicial un mapa con todas las estaciones señaladas.
Ver geolocalización del usuario	-La aplicación muestra un icono para saber la localización exacta del usuario siempre que este tenga el GPS encendido e su dispositivo.
Ver información de estaciones	-El usuario puede desplazarse por el mapa haciendo gestos con el Smartphone hasta localizar la estación deseada. -El usuario hace clic en el *marker*[1] de la estación que desea.

[1] *Marker*: objeto del mapa interactivo proporcionado por la API Google Maps que representa un punto geográfico.

	-La aplicación muestra una ventana con la información de dicha estación.
Mostrar menú lateral	-El usuario pulsa el botón de menú o realiza un deslizamiento desde el borde izquierdo hacia el interior de la pantalla. -La aplicación despliega un menú lateral.

Tabla 3.4: Flujo de eventos Sprint 1

Resultado al final del Sprint 1

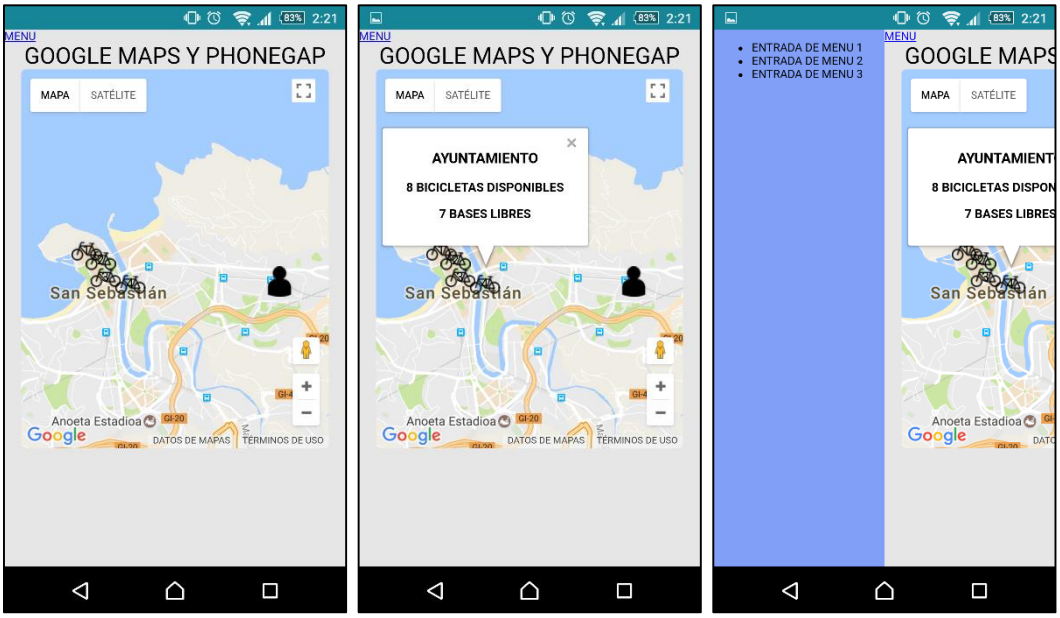


Figura 3.3: Sprint 1

3.2.2 Sprint dos

Historias de usuario

A continuación se muestra las historias de usuario que forman parte del segundo Sprint:

	Historia de usuario
Número: 3	**Usuario:** Desarrollador
Nombre historia: Obtener datos de las estaciones	
Sprint Asignado: 2	
Descripción: Quiero obtener todos los datos relacionados a las estaciones de las bicicletas en un formato normalizado.	

Validación: El servidor enviará los datos de las estaciones de bicicletas en formato JSON. Entre estos datos se incluirá la localización de las estaciones, la cantidad de bicicletas disponibles, la cantidad de bases disponibles para dejar la bicicleta, el nombre de la estación y cualquier otro tipo de información relevante.

Tabla 3.5: Historia de usuario 3

Historia de usuario	
Número: 4	**Usuario:** Cliente
Nombre historia: Saber la estación más cercana	
Sprint Asignado: 2	
Descripción: Quiero saber cuál es la estación de bici más cercana a mi localización actual, saber la distancia a la que se encuentra y saber cuántas bicis y bases tiene disponibles.	
Validación: Se le mostrará al usuario los datos de la estación más cercana y los metros a los que se encuentra dicha estación.	

Tabla 3.6: Historia de usuario 4

Historia de usuario	
Número: 13	**Usuario :** Desarrollador
Nombre historia: Quiero una interfaz de usuario más agradable	
Sprint Asignado: 2	
Descripción: Quiero que la aplicación se vea bien, sea agradable y atraiga a la vista.	
Validación: Se utilizará un *framework CSS*, en concreto Materialize, para dotar a la página de un diseño y unas animaciones más atractivas.	

Tabla 3.7: Historia de usuario 13

Sprint Backlog

Tarea	Estado	Horas pendientes	Horas dedicadas
Implementar *back-end*	Completada	-	15
Mostrar estaciones en tiempo real	Completada	-	5
Mostrar la estación más cercana	Completada	-	6

Implementar Materialize	Completada	-	20

Tabla 3.8: Sprint Backlog 2

Casos de uso

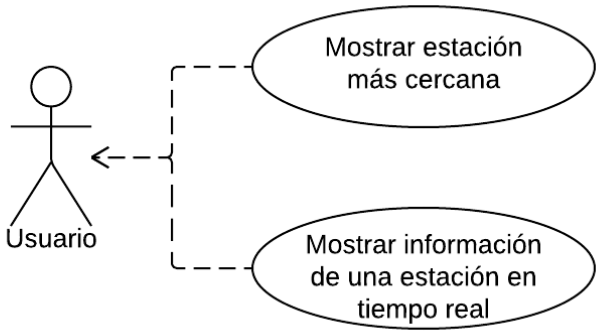


Figura 3.4: Casos de uso Sprint 2

Flujo de eventos

Caso de uso	Flujo de eventos
Mostrar estación más cercana	-El usuario pulsa en el botón de estación más cercana. -El mapa se reposiciona con dicha estación en el centro del mismo. -El sistema simula un clic en el *marker* de la estación más cercana.
Mostrar información de una estación en tiempo real	-El usuario hace clic en la estación que desea. -El sistema muestra una ventana con los datos adquiridos mediante una consulta AJAX.

Tabla 3.9: Flujo de eventos Sprint 2

Resultado al final del Sprint 2

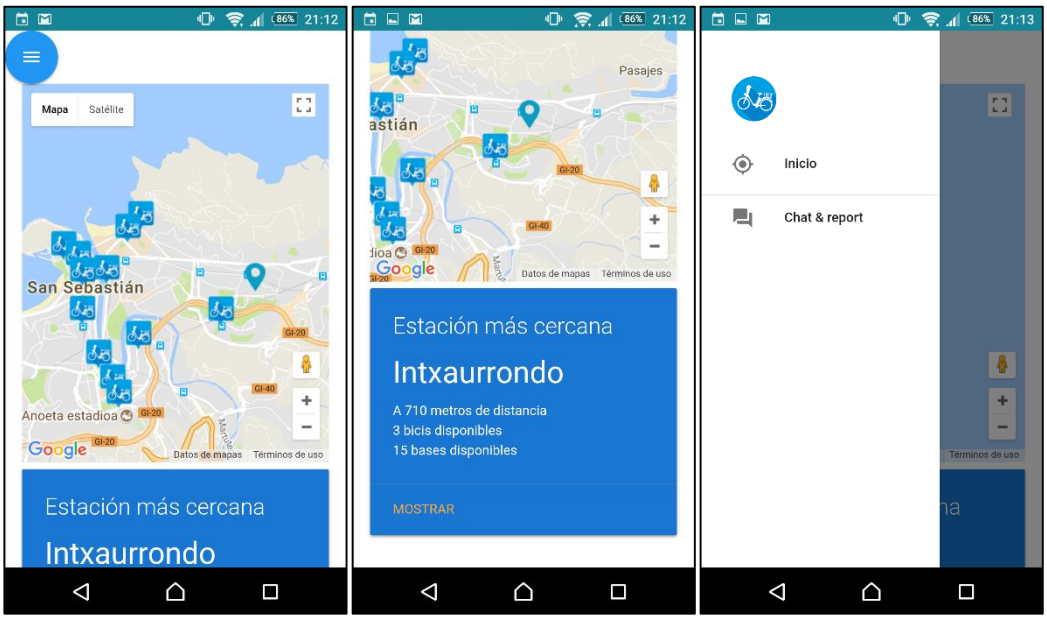


Figura 3.5: Sprint 2

3.2.3 Sprint tres

Historias de usuario

A continuación se muestra las historias de usuario que forman parte del tercer Sprint:

	Historia de usuario
Número: 6	**Usuario:** Cliente
Nombre historia: Chat con otros usuarios	
Sprint Asignado: 3	
Descripción: Quiero disponer de un chat en el que pueda compartir información con otros usuarios del sistema de la aplicación.	
Validación: El cliente tiene acceso a un chat global para los usuarios de la aplicación en el que podrán intercambiar información mediante mensajes de texto.	

Tabla 3.10: Historia de usuario 6

Historia de usuario	
Número: 14	**Usuario:** Desarrollador
Nombre historia: Trasladar el servidor a un host configurable	
Sprint Asignado: 3	
Descripción: Quiero tener un servidor donde pueda establecer tareas programadas con *crontab*, crear *daemons* y poder tener acceso a la configuración PHP de mi servidor.	
Validación: El *back-end* se trasladará a un host de Amazon EC2 el cual ofrece una instancia del SO que se elija para tener acceso y control total del mismo.	

Tabla 3.11: Historia de usuario 14

Historia de usuario	
Número: 15	**Usuario:** Cliente
Nombre historia: Reportar el estado de una bicicleta	
Sprint Asignado: 3	
Descripción: Quiero poder marcar si una bici está en mal estado, en buen estado o cualquier otra observación para saber el estado de la bicicleta.	
Validación: Se podrá hacer uso de un formulario para reportar el estado de una bicicleta en el que se incluirá el número de la misma, su estado y un área de texto para cualquier información relevante sobre la misma.	

Tabla 3.12: Historia de usuario 15

Historia de usuario	
Número: 16	**Usuario:** Cliente
Nombre historia: Ver el estado de una bicicleta	
Sprint Asignado: 3	
Descripción: Quiero poder saber si una bici está en mal estado, en buen estado o cualquier otra observación que otros usuarios hayan hecho sobre la misma.	
Validación: Se podrá acceder a un formulario donde una vez seleccionada la bicicleta por su número aparecerá un listado con los últimos cinco reportes de la misma.	

Tabla 3.13: Historia de usuario 16

Sprint Backlog

Tarea	Estado	Horas pendientes	Horas dedicadas
Trasladar el servidor	Completada	-	20
Chat	Completada	-	25
Reportar estado	Completada	-	3
Ver Estado	Completada	-	4

Tabla 3.14: Sprint Backlog 4

Casos de uso

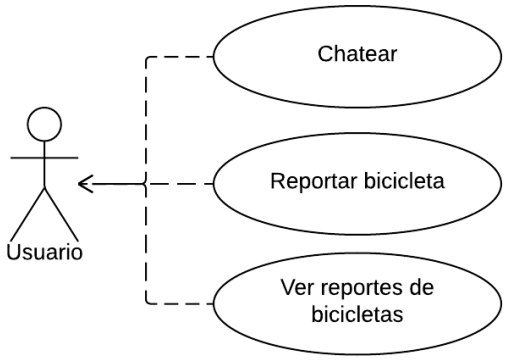


Figura 3.6: Casos de uso Sprint 3

Flujo de eventos

Caso de uso	Flujo de eventos
Chatear	-El sistema muestra un área de texto para introducir el nombre y otra área de texto para introducir el mensaje. -El usuario introduce su nombre y, después, introduce su mensaje. -El sistema guarda el nombre y oculta dicha área de texto. Posteriormente, envía el mensaje al servidor. -El servidor recibe el mensaje y lo reenvía a todos los usuarios conectados.

Reportar bicicleta	-El sistema muestra un formulario a rellenar con tres apartados, un *inputbox*[2] para el número de bicicleta, un *selectbox*[3] para el estado y un área de texto[4] para las observaciones. -El usuario rellena los tres datos y pulsa el botón enviar. -El sistema reenvía al usuario a la ventana de chat donde se muestra que la bici ha sido evaluada.
Ver reportes de bicicletas	-El sistema muestra un *input box* donde se puede introducir el número de la bicicleta. -El usuario introduce el número deseado y pulsa el botón. -El sistema muestra las últimas cinco valoraciones de dicha bicicleta.

Tabla 3.15: Flujo de eventos Sprint 3

Resultado al final del Sprint 3

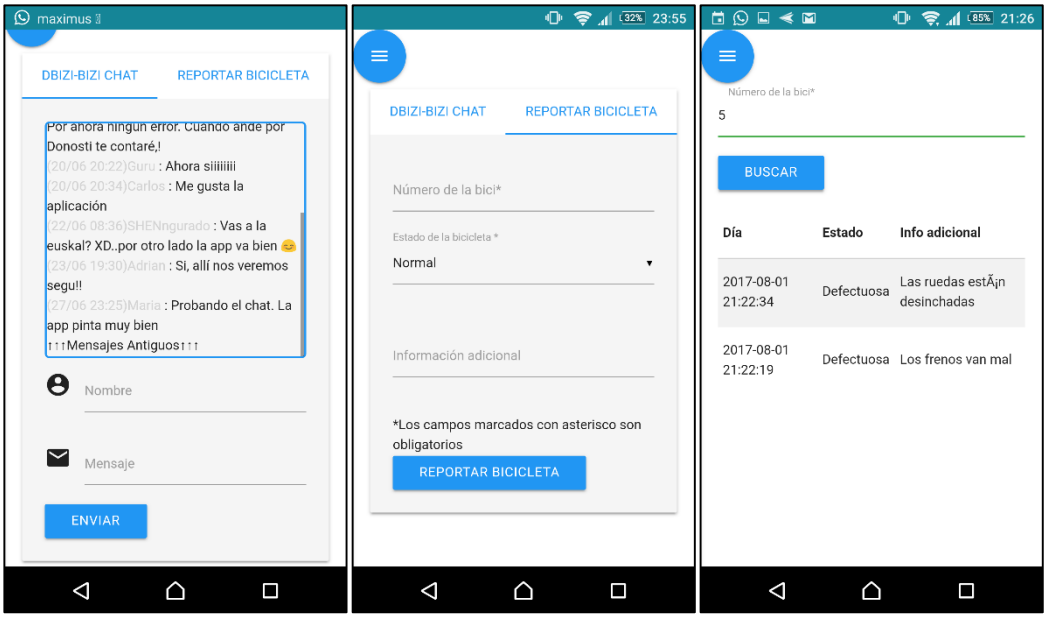


Figura 3.7: Sprint 3

[2] Muestra un indicador en un diálogo en el que el usuario puede introducir texto.
[3] Lista desplegable que permite al usuario seleccionar entre varios atributos preseleccionados.
[4] donde el usuario es capaz de introducir un texto de mayor dimensión que el *Inputbox*.

3.2.4 Sprint cuatro

Historias de usuario

A continuación se muestra las historias de usuario que forman parte del cuarto Sprint:

	Historia de usuario
Número: 11	**Usuario:** Cliente
Nombre historia: Recibir alertas de estaciones libres	
Sprint Asignado: 4	
Descripción: Quiero recibir alertas en el móvil cuando una de las estaciones tenga al menos una bicicleta disponible o una base para colocar la bicicleta.	
Validación: El cliente recibirá una notificación en el móvil cuando alguno de los eventos mencionados ocurra.	

Tabla 3.16: Historia de usuario 11

Sprint Backlog

Tarea	Estado	Horas pendientes	Horas dedicadas
Enviar alertas desde el servidor	Completada	-	4
Gestionar alertas en la aplicación	Completada	-	40

Tabla 3.17: Sprint Backlog 4

Casos de uso

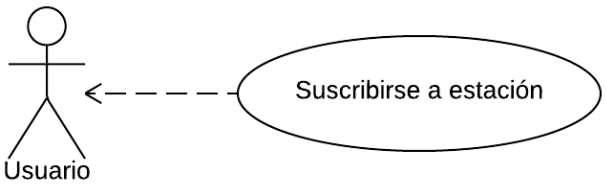


Figura 3.8: Casos de uso Sprint 4

Flujo de eventos

Caso de uso	Flujo de eventos
Suscribirse a estación	-El sistema muestra una lista de información con las estaciones a las que está suscrito ese dispositivo y otra lista en un *select box* con aquellas a las que puede suscribirse. -El usuario selecciona la estación que desea y pulsa el botón para suscribirse. -El sistema muestra una alerta diciendo que se ha suscrito correctamente. <u>Flujo de eventos alternativos</u> -Si el sistema no ha podido realizar la petición se muestra un mensaje con el error correspondiente.
Desuscribirse de todas las estaciones	-El sistema muestra una lista de información con las estaciones a las que está suscrito ese dispositivo junto a una lista con aquellas a las que puede suscribirse, un botón para suscribirse y otro para desuscribirse. -El usuario pulsa el botón de desuscribirse y el sistema le muestra un mensaje de alerta diciendo que se ha desuscrito correctamente.

Tabla 3.18: Flujo de eventos Sprint 4

Resultado final del Sprint 4

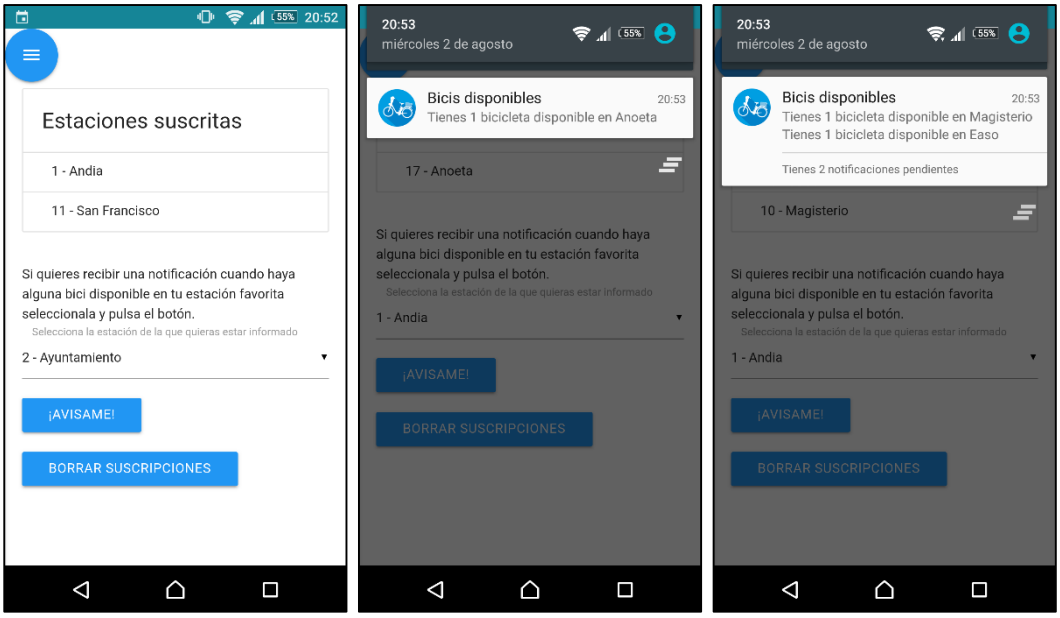


Figura 3.9: Sprint 4

3.3 RESULTADO FINAL

Una vez terminado el cuarto Sprint se obtiene como resultado una aplicación funcional y disponible para presentar al público. La siguiente figura muestra el resultado final de la aplicación.

En el anexo C puede encontrarse un manual de usuario para esta versión de la aplicación.

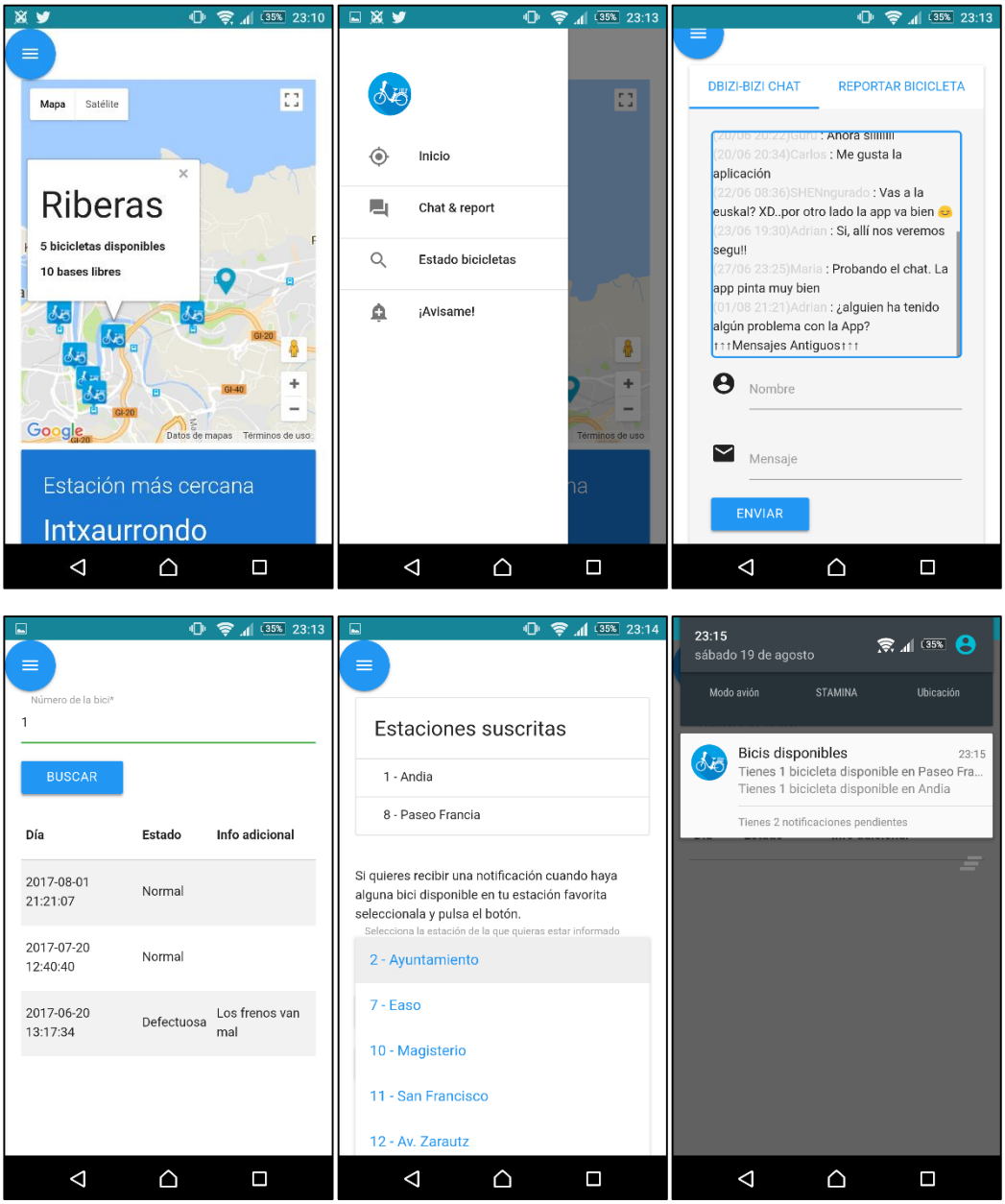


Figura 3.10: Resultado final

En la figura 3.11 se muestran todos los casos de uso implementados en el producto final.

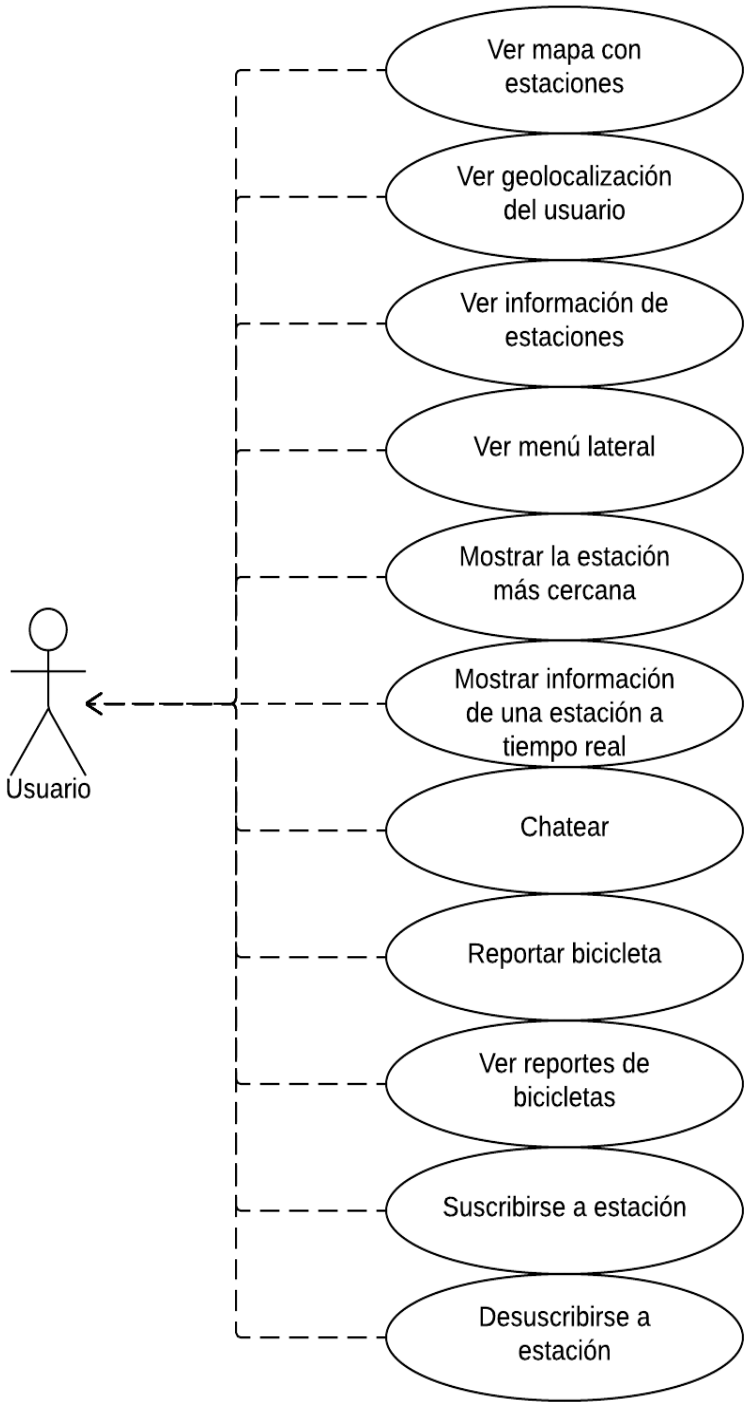


Figura 3.11: Modelo de casos de todos los casos de uso

4 DISEÑO Y ARQUITECTURA

A lo largo de este capítulo se detalla tanto la arquitectura utilizada para la aplicación (MVC) como su implementación en el proyecto. Asimismo, se especifica el modelo de datos utilizado junto a una explicación sobre su elección frente a otras opciones y la funcionalidad de los datos en dicho modelo. Por último se muestra un diagrama de secuencia para mostrar la interacción de los objetos con el sistema.

4.1 ARQUITECTURA

4.1.1 Modelo-Vista-Controlador

Se trata de un patrón de arquitectura de software que separa el modelo de datos, la lógica de negocio y la interfaz de usuario (Vista) en tres capas distintas. Esto facilita tanto la reutilización de código como el desarrollo en paralelo del producto.

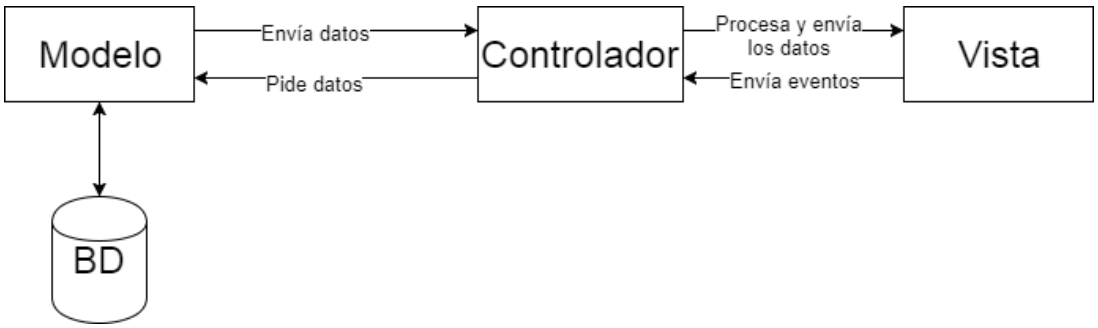
Componentes del MVC:

Modelo: Es el encargado de actualizar, consultar, buscar… los datos, generalmente a través de una base de datos.

Controlador: Es el que recibe las órdenes del usuario y se encarga de solicitar los datos al modelo y comunicárselos a la Vista.

Vistas: Es el componente que se encarga de la representación visual de los datos, la interfaz gráfica.

Este patrón asegura otras ventajas además de las ya mencionadas dado que separar la aplicación en estas tres capas nos permite hacer modificaciones a una sola de forma mucho más eficiente y sin que estos afecten a toda la aplicación.



4.1.2 MVC en la aplicación

Debido a las tecnologías escogidas, en concreto al *framework* PhoneGap, el MVC de nuestro proyecto es similar al que podemos encontrar en una página web.

Además, tenemos las vistas que serán generadas mediante los lenguajes HTML5, JavaScript y CSS, los cuales estarán alojados en los dispositivos de cada usuario. Estos se adquieren mediante la instalación de la aplicación.

Mientras, los controladores estarán desarrollados en PHP bajo el *framework* CodeIgniter, los cuales serán los encargados de atender los eventos solicitados por las vistas e indicar las operaciones a realizar al modelo.

Estos modelos, que también están desarrollados en PHP bajo CodeIgniter, son los encargados de trabajar los datos mediante su consulta, creación o modificación y de entregar dicha información a los controladores.

Los datos de la aplicación se guardan en una base de datos MySQL [25] alojada en su mismo servidor. La información transportada desde la base de datos hasta la vista se realiza mediante el formato estandarizado de JSON [26].

4.2 MODELO DE DATOS

Toda la información que utiliza dBizi-bizi se encuentra almacenada en un servidor con una base de datos administrada con MySQL. Sobre esta base de datos se realizarán las peticiones de actualización, modificación e inserción correspondientes. Se trata de una base de datos relacional [27] en la que existen distintas tablas para alojar la información correspondiente a los distintos apartados de la aplicación, como pueden ser las estaciones, el chat, el estado de las bicicletas…

Todas las tablas de la BBDD funcionan bajo el motor de almacenamiento InnoDB [28] puesto que ofrece una mayor rapidez respecto al motor MyISAM [29] en cuanto a la ejecución de inserción, actualización y borrado de consultas, que serán las que mayormente se generarán en la aplicación.

La base de datos no es muy extensa puesto que no se ha requerido guardar más información para las funcionalidades pactadas con el cliente. Las tablas que forman la base de datos son las siguientes:

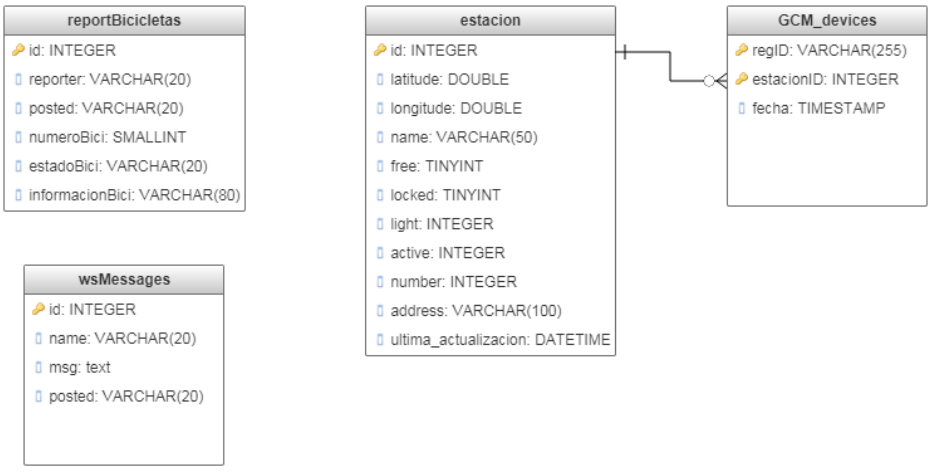


Figura 4.1: Esquema BBDD

La tabla estación guarda toda la información que puede obtenerse a través del portal dbizi.com. Esta información incluye la geolocalización, la dirección, el nombre, el número de bicicletas, las bases disponibles y la fecha y hora de la última vez que se obtuvieron los datos de dicho portal.

La tabla GCM_devices almacena tanto el identificador del dispositivo que quiere recibir las notificaciones *push* [30] como la estación a la que está suscrito y la fecha y hora a la que se suscribió.

La tabla reportBicicletas guarda una entrada por cada reporte que hayan enviado los usuarios sobre las bicicletas. Entre los datos que se guardan se incluyen el nombre la persona que envió el reporte, la fecha, el número de la bici, su estado e información adicional.

La tabla wsMessages almacena mensajes de chat que los usuarios escriben mediante la aplicación junto a la fecha y el emisor.

4.3 DIAGRAMA DE SECUENCIA

Los diagramas de secuencia muestran la interacción de un conjunto de objetos en un sistema a través de tiempo. Estos diagramas detallan la implementación de un sistema en un escenario concreto. A continuación se muestra el ejemplo de un diagrama de secuencia que se da en la aplicación dBizi-bizi.

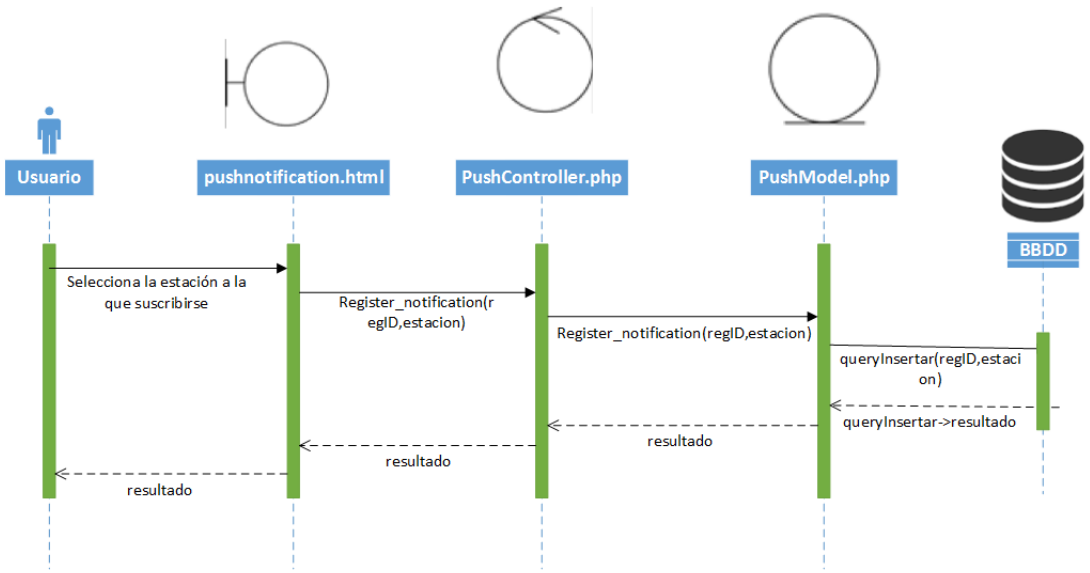


Figura 4.2: Diagrama de secuencia

Este caso corresponde al registro o suscripción de un dispositivo a una estación concreta para la recepción de notificaciones push. Comienza cuando la vista pushnotification.html detecta el evento accionado por el usuario de suscribirse a una estación, lo cual envía una petición al controlador Pushcontroller.php junto al identificador del dispositivo y la estación seleccionada. El controlador deriva la petición al modelo PushModel.php el cual es el encargado de introducir en la BBDD dicha petición y de devolver al controlador el resultado de dicha consulta. Por último, el controlador se encarga, a su vez, de devolver dicha respuesta a la Vista para presentarle el estado de la petición al usuario.

5 IMPLEMENTACIÓN

Como ya se ha explicado en el capítulo anterior, el funcionamiento de la aplicación dBizi-bizi seguirá el patrón MVC. En este capítulo se muestra la implementación de dicho patrón en la aplicación. Está separada en tres capas (Modelo, Vista y Controlador) donde tanto el Controlador como el Modelo estarán alojados en el servidor mientras las Vistas lo estarán en la aplicación móvil. Se mostrarán, además, partes del código donde se ve reflejada la separación de dichas capas y la interacción entre ellas.

5.1 VISTAS

Las vistas generadas mediante HTML5, CSS3 y JavaScript se encuentran en el lado cliente de la aplicación. Son las encargadas de generar una interfaz de usuario.

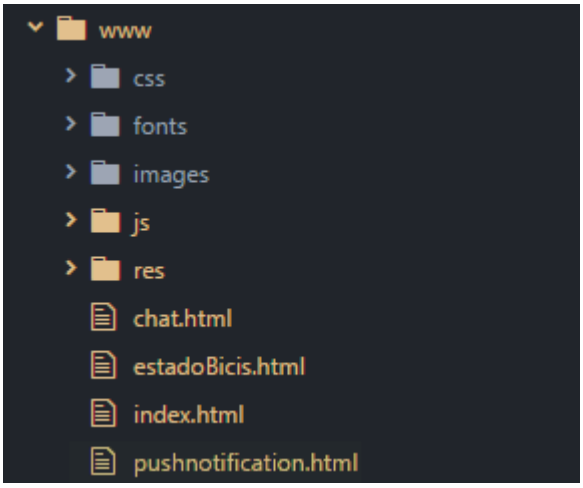


Figura 5.1 : Vistas de dBizi-bizi

El lenguaje HTML es el encargado de establecer el esqueleto de la vista, es decir, cómo está dividido y el texto que contiene. Mientras, las plantillas de estilo CSS son las encargadas de dar el aspecto a dicho esqueleto, añadiéndole distintos tamaños a las letras, colores, sombras y una infinidad de adornos para hacer atractiva e intuitiva nuestra aplicación. Por último, el lenguaje JavaScript nos aporta tanto dinamismo (movimientos, transiciones, alertas…) como conexión con el servidor mediante consultas AJAX (Asynchronous JavaScript And XML). Esta tecnología permite mantener una comunicación asíncrona entre el cliente y el servidor y logra el intercambio de datos entre los mismos sin necesidad de recargar la página entera y sin interferir en lo que se está mostrando en ese momento.

```
$.ajax({
  url: 'http://ec2-35-176-101-5.eu-west-2.compute.amazonaws.com/index.php/PushController/get_registeredStations',
  type: 'post',
  data: {
    'regID': window.sessionStorage.getItem("registrationId")
      },
  dataType: 'json',
  error: function(jqXHR, text_status, strError) {
    alert("No se pudieron obtener las estaciones registradas");
  },
  timeout: 3000,
  success: function(data) {
    var estacionesSelect = $('#estaciones');
    var estacionesSuscritas = $('#estacionesSuscritas');
    $.each(data, function(i, estacion) {
      if (estacion.registrado == "true")
        estacionesSuscritas.append($("<li class='collection-item' ></li>")
          .text(estacion.id + " - " + estacion.name));
      else {
        estacionesSelect.append($("<option></option>")
          .attr("value", estacion.id)
          .text(estacion.id + " - " + estacion.name));
      }
    });


    $('select').material_select();
    $('#preloader').remove();
    $('#send-btn').removeClass("disabled");
    $('#delete-btn').removeClass("disabled");
  }
});
```

Figura 5.2: AJAX en dBizi-bizi

En la figura 5.2 se puede ver cómo se realiza una petición AJAX a la URL de nuestro servidor pasándole la variable “regID” en formato JSON. En caso de que el servidor devuelva correctamente una respuesta se realizará la función incluida en “success”, la cual rellena las listas “estacionesSelect” y “estacionesSuscritas” en función de la respuesta devuelta por el servidor. En caso de no obtener respuesta o una incorrecta por parte del servidor se mostrará la función incluida en “error”.

En la Figura 5.3 se puede observar como dichas listas empiezan vacías y solo se rellenan en función de la respuesta enviada por el servidor. El “preloader” incluido en “estacionesSuscritas” es un componente proporcionado por Materialize para indicar mediante una imagen en movimiento que todavía existe algo que ha de cargarse.

```
<div id="contenidoCuerpo">
  <!-- push notifications -->

  <ul class="collection with-header" id="estacionesSuscritas">
    <li class="collection-header">
      <h5>Estaciones suscritas</h5></li>
      <div id="preloader" class="progress blue lighten-3">
        <div class="indeterminate blue"></div>
      </div>
  </ul>
  <br /> Si quieres recibir una notificación cuando haya alguna bici disponible en tu estación favorita seleccionala y pulsa el botón.
  <form id="formulario" name="formulario"  >
    <div class="input-field col s12">
      <select id="estaciones"></select>
      <label>Selecciona la estación de la que quieras estar informado</label>
    </div>
    <button class="btn disabled blue" type="button" id="send-btn">¡Avisame!</button><br /><br />
    <button class="btn disabled blue" type="button" id="delete-btn">Borrar suscripciones</button>

  </form>
```

Figura 5.3: Vista de notificaciones

Por último, en la figura 5.4 se muestra un pequeño fragmento del código CSS utilizado en nuestra aplicación en el cual damos forma a tanto a los “message_box” como a los “menuIcon” y donde reemplazamos algunos de los estilos proporcionados por Materialize mediante la etiqueta “!important”, la cual hace que se ignore otras reglas que reemplacen a esa misma.

```css
#message_box{
  border-radius: 5px;
  border: 2px solid #2196f3;
  height: 250px;
  overflow:auto;
}
#menuIcon{
  max-width: 100%;
max-height: 100%;
}

ul.dropdown-content.select-dropdown li span {
    color: #2196f3; /* no need for !important :) */
}

.tabs .indicator {
    background-color: #2196f3 !important;
}
.tabs .tab a{
    color:  #2196f3 !important;
}
.tabs .tab a:hover,.tabs .tab a.active {
  background-color:transparent;
  color:  #2196f3 !important;
}
.tabs .tab.disabled a,.tabs .tab.disabled a:hover {
  color:  #2196f3 !important;
}
```

Figura 5.4: CSS dBizi-bizi

5.2 CONTROLADORES Y MODELOS

El servidor es el encargado de alojar el *framework* CodeIgniter con los archivos en PHP y los datos en SQL que forman los Controladores y los Modelos de la arquitectura MVC. En la figura 5.5 se puede observar tanto los Controladores como el Modelo. El chat de la aplicación se encuentra fuera del *framework* CodeIgniter ya que este no soporta el protocolo WebSocket [31] del que se hablará posteriormente.

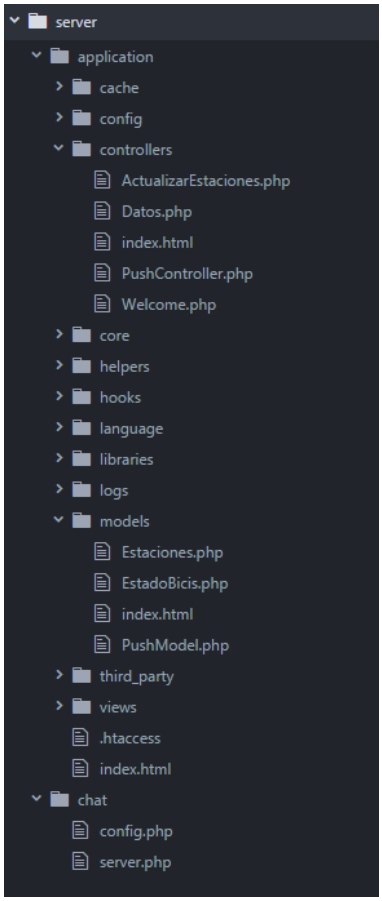


Figura 5.5: Ficheros del servidor

Los Controladores son los encargados de recibir las peticiones desde las vistas y realizar, en caso de ser necesario, la petición de datos al Modelo correspondiente. A continuación, se muestra un Controlador que se encarga de la lógica de negocio y adquiere los datos a través de una petición realizada a un Modelo.

```php
public function send_notifications(){
  //Obtiene las estaciones a las que NO esta suscrito el usuario
  $this->load->model('PushModel');

  $title="Bicis disponibles";

  $result= $this->PushModel->get_FreeRegisteredID();
  foreach($result->result_array() as $row)
  {
    $to=$row['regID'];
    if ($row['free']=1){
      $message="Tienes 1 bicicleta disponible en ".$row['name'];
    }else{
      $message="Tienes ".$row['free']." bicicletas disponibles en ".$row['name'];
    }
    $sendResult = $this->sendPush($to,$title,$message);
    $jsonResult = json_decode($sendResult, true);

    //una vez enviado el mensaje, lo borramos de la BBDD
    if($jsonResult['success'] == 1 ){
      $this->PushModel->delete_notification($row['regID'],$row['id']);
    }

  }
```

Figura 5.6: Controlador dBizi-Bizi

En la figura 5.6 se encuentra la función que envía notificaciones push a aquellos usuarios que así lo deseen. Para ello, primero carga el modelo "PushModel" del cual obtendremos los datos. Después, obtiene los dispositivos suscritos en la variable "result" y, tras ello, envía las notificaciones a dichos usuarios usando la tecnología GCM [32] (Google Cloud Messaging) en la que se profundizará posteriormente.

La figura 5.7 muestra la función del Modelo a la que se ha hecho referencia y en la que se ejecuta una consulta SQL sobre una base de datos.

```php
public function get_registeredStations($regID){

  //Obtiene las estaciones a las que esta suscrito el usuario
  $sql = "select estacion.id, estacion.name from estacion left join GCM_devices on estacion.id = GCM_devices.estacionID where GC

  $query = $this->db->query($sql);
  return $query;

}
```

Figura 5.7: Modelo dBizi-bizi

Como puede observarse en la figura 5.6, los datos que devuelve el servidor están en formato JSON. El siguiente fragmento de código muestra parte de un ejemplo real emitido por el servidor para el estado de las estaciones en dicho formato.

```
[
  {
    "id":"1",
    "latitude":"43.320526",
    "longitude":"-1.982469",
    "name":"Andia",
    "free":"0",
    "locked":"13",
    "light":"1",
    "active":"0",
    "number":"2",
    "address":"Calle Andia",
    "ultima_actualizacion":"2017-08-14 23:51:02"
  },
  {
    "id":"2",
    "latitude":"43.321796",
    "longitude":"-1.985112",
    "name":"Ayuntamiento",
    "free":"0",
    "locked":"0",
    "light":"0",
    "active":"1",
    "number":"1",
    "address":"Injentea 1",
    "ultima_actualizacion":"2017-08-14 23:51:02"
  },
]
```


5.3 API Y OTRAS TECNOLOGÍAS

Durante el desarrollo de la aplicación dBizi-bizi, se ha hecho uso de algunas tecnologías con sus correspondientes API para mejorar la experiencia del usuario y para facilitar el desarrollo del proyecto. En este apartado se nombrarán y explicarán su función dentro de la aplicación.

5.3.1 Jquery

Es una librería de JavaScript diseñada para simplificar los *scripts* que se ejecutan en el lado del cliente. Se trata de la librería de JavaScript más utilizada. Una de las grandes ventajas de esta librería es que facilita la navegación entre los elementos DOM [34] de la página y la creación de consultas AJAX. La Figura 5.2 es un claro ejemplo del uso de JQuery.

5.3.2 WebSocket

Se trata de un protocolo de comunicación *full-duplex*[5] bajo una conexión TCP[6]. Esta tecnología nos permite tener un chat en tiempo real en el que los mensajes se suministran de forma instantánea a todos aquellos usuarios que estén conectados con el servidor. Este protocolo utiliza cabeceras pequeñas que facilitan el tiempo real de transferencia de datos entre el cliente y el servidor evitando un consumo excesivo de los datos del propio usuario. Para mostrar el funcionamiento del protocolo se adjuntan las siguientes imágenes de casos concretos que gestiona el servidor.

```php
//Create TCP/IP sream socket
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
//reuseable port
socket_set_option($socket, SOL_SOCKET, SO_REUSEADDR, 1);
//bind socket to specified host
socket_bind($socket, 0, $port);
//listen to port
socket_listen($socket);
//create & add listning socket to the list
$clients = array($socket);
```

Figura 5.8: Inicialización de socket

En el ejemplo de la figura 5.8 se observa cómo se crea un *socket* en un puerto específico y se pone a la espera de conexiones entrantes desde el lado del cliente mediante el comando "socket_listen".

[5] Transmisión de datos capaz de mantener una comunicación bidireccional.
[6] Protocolo de Control de transmisión.

```php
//check for new socket
if (in_array($socket, $changed)) {
  $socket_new = socket_accept($socket); //accpet new socket
  $clients[] = $socket_new; //add socket to client array

  $header = socket_read($socket_new, 1024); //read data sent by the socket
  perform_handshaking($header, $socket_new, $host, $port); //perform websocket handshake
  socket_getpeername($socket_new, $ip); //get ip address of connected socket

  //Display old messages
  send_old_messages($socket_new);

  //make room for new socket
  $found_socket = array_search($socket, $changed);
  unset($changed[$found_socket]);
}
```

Figura 5.9: Conexión entrante al socket

En la figura 5.9 se puede ver cómo se gestiona una conexión entrante en la que, primero, se acepta dicha conexión y, después, se añade al *array* que tiene guardadas todas las conexiones en tiempo real. Luego, se establece la forma de comunicación entre el servidor y el cliente mediante la función "perform_handshaking". Posteriormente, se le envía a la nueva conexión los últimos mensajes que se hayan enviado al chat. Por último, se marca la conexión como gestionada mediante el "unset".

```javascript
//#### Message received from server?
websocket.onmessage = function(ev) {
  var msg = JSON.parse(ev.data); //PHP sends Json data
  var type = msg.type; //message type
  var umsg = msg.message; //message text
  var uname = msg.name; //user name
  var ucolor = msg.color; //color

  if(type == 'usermsg')
  {
    var posted = msg.posted;
    var postedDay = posted.substring(8,10);
    var postedMonth = posted.substring(5,7);
    var postedHour = posted.substring(11,16);
    var showedDate = postedDay+"/"+postedMonth+" "+postedHour;
    $('#message_box').append("<div><span class=\"user_name\" style=\"color:#"+ucolor+"\">("+showedDate+")"+uname+"</span> : <span cla
  }
  if(type == 'system')
  {
    $('#message_box').append("<div class=\"system_msg\">"+umsg+"</div>");
  }


  var objDiv = document.getElementById("message_box");
  objDiv.scrollTop = objDiv.scrollHeight;
};

websocket.onerror = function(ev){$('#message_box').append("<div class=\"system_error\">Ha ocurrido un error - "+ev.data+"</div>");};
websocket.onclose   = function(ev){$('#message_box').append("<div class=\"system_msg\">Conexión cerrada</div>");};
```

Figura 5.10: Gestión de websocket en el lado del cliente

Finalmente, en la figura 5.10 se muestra cómo se gestiona la comunicación mediante WebSockets en el lado del cliente en la que se obtiene los datos en forma JSON. También, se gestiona el cierre y el posible error que puede llegar a producirse en la comunicación.

5.3.3 Google Maps

Se trata de un servicio gratuito de la compañía Alphabet Inc. (compañía predecesora de Google Inc.) que permite incluir su servicio Maps en cualquier página o dispositivo móvil. También, permite controlar distintas características como la localización inicial, el zoom o la inclusión de marcadores para aportar información adicional.

```
myLatlng = new google.maps.LatLng(lat, long);
var mapOptions = {
  zoom: 13,
  center: myLatlng,
  gestureHandling: 'greedy',
};


map = new google.maps.Map(document.getElementById('map-canvas'), mapOptions);
var markerCercano = new google.maps.Marker();
var infowindow = new google.maps.InfoWindow();

//muestra donde esta el usuario en el mapa
var markerUser = new google.maps.Marker({
  position: myLatlng,
  map: map,
  icon :'images/location-icon-blue.png'
});
```

Figura 5.11: Implementación de Google Maps

En la figura 5.11 podemos observar tanto la inicialización del mapa como de un marcador en unas coordenadas determinadas.

5.3.4 *Web scraping*

Es una técnica que se utiliza para extraer datos de otras páginas web. En este caso, se utilizará para recoger los datos en tiempo real de la página oficial de dBizi y actualizar la base de datos de este proyecto.

```
//OBTIENE LOS DATOS DE LA WEB DBIZI//
private function getDatosFromdBizi(){

  $html = file_get_contents('https://www.dbizi.com/map/'); //Convierte la información de la URL en cadena

  $posicion_datos_inicio_de_estaciones = strpos($html, '{"stations":');
  $posicion_datos_fin_de_estaciones = strpos($html, "}]}');")+3;
  $length = $posicion_datos_fin_de_estaciones - $posicion_datos_inicio_de_estaciones;
  $datos = substr ( $html , $posicion_datos_inicio_de_estaciones , $length  );
  $datos_json = json_decode($datos,true);
  return $datos_json;


}
```

Figura 5.12: Web scraping

5.3.5 Google Cloud Messaging

Es otro servicio gratuito de Google que permite enviar información en forma de notificación desde nuestro servidor a los dispositivos de los usuarios. Este servicio maneja todo lo referente a la gestión y emisión de los mensajes a las aplicaciones cliente. Para su utilización en nuestro servidor, se debe activar el módulo cURL [34] de PHP.

```
$fields = array(
  'registration_ids' => $registrationIds,
  'data' => $msg
);

$headers = array(
  'Authorization: key=' . API_ACCESS_KEY,
  'Content-Type: application/json'
);

$ch = curl_init();
curl_setopt( $ch,CURLOPT_URL, 'https://android.googleapis.com/gcm/send' );
curl_setopt( $ch,CURLOPT_POST, true );
curl_setopt( $ch,CURLOPT_HTTPHEADER, $headers );
curl_setopt( $ch,CURLOPT_RETURNTRANSFER, true );
curl_setopt( $ch,CURLOPT_SSL_VERIFYPEER, false );
curl_setopt( $ch,CURLOPT_POSTFIELDS, json_encode( $fields ) );
$result = curl_exec($ch );
curl_close( $ch );
return $result;
}
```

Figura 5.13: Implementación de GCM

En la figura 5.13 se observa cómo se crean dos "*arrays*", uno con la información del mensaje y sus destinatarios y otro con la clave que te proporciona Google para utilizar su servicio. Después, se llama al servicio GCM mediante el módulo cURL de PHP.

5.3.6 Crontab

No se trata de una tecnología, sino de una utilidad que nos proporcionan los sistemas UNIX para la administración regular de procesos en segundo plano. Este administrador se utiliza para realizar dos funciones cada minuto. Una actualiza los datos de nuestra base de datos mediante la técnica *Web scraping* mencionada anteriormente y la otra comprueba y envía las notificaciones push mediante el sistema GCM.

5.3.7 Google Play

Google Play o Play Store es la tienda de aplicaciones para Android creada por Google. Esta plataforma permite administrar y distribuir cualquier aplicación a todo aquel que disponga de un dispositivo Android. Es la tienda de aplicaciones más utilizada de Android dado que es la que viene instalada por defecto en todas las distribuciones del sistema operativo de Google. Esta plataforma se utilizará para que la gente sea capaz de adquirir la aplicación de una forma oficial, segura y sencilla.

6 PRUEBAS

A lo largo de este capítulo se muestran las pruebas realizadas en la aplicación para comprobar el funcionamiento esperado de las funcionalidades implementadas.

Las pruebas realizadas serán de tipo caja negra. Así, cada funcionalidad es estudiada desde el punto de vista de las entradas y salidas que recibe o desde las respuestas que genera, las cuales son sin tener en cuenta el código de la aplicación.

Las pruebas se han realizado sólo sobre dispositivos reales de Android, puesto que para compilar la aplicación de iOS se necesita un certificado no gratuito del que se dispone. Aun así, gracias a una aplicación denominada PhoneGap Developer APP hemos conseguido emular el comportamiento de la aplicación en modo desarrollo.

6.1 PRUEBAS EN LAS FUNCIONALIDADES

6.1.1 Pruebas en el mapa

Esta funcionalidad permite al usuario ver su localización, el de las estaciones y la disponibilidad de las mismas.

Prueba 1: El usuario intenta entrar a la aplicación sin tener ningún acceso a Internet.

Entrada: Inicio de la aplicación sin acceso a Internet.

Salida esperada: El mapa no podrá cargarse y se avisará al usuario del error.

Salida real: El mapa no puede cargarse y se le avisa al usuario.

Prueba 2: El usuario inicia la aplicación sin activar el GPS o en una zona cuya cobertura es demasiado baja para obtener una señal.

Entrada: Inicio de la aplicación sin señal GPS.

Salida esperada: No podrá obtenerse la geolocalización del usuario y se le avisará para que active el GPS y reinicie la aplicación.

Salida real: No se puede obtener la geolocalización y se le avisa al usuario con un mensaje.

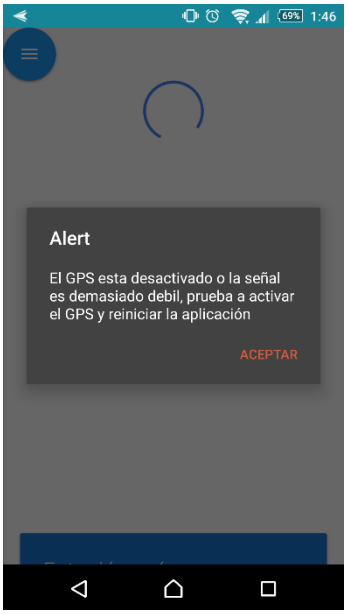


Figura 6.1: Pruebas en el mapa

6.1.2 Pruebas en el chat

Esta funcionalidad permite a los usuarios interactuar entre ellos mediante una sala de chat común para todos.

Prueba 1: El usuario intenta conectarse al chat sin tener ningún acceso a Internet.

Entrada: Inicio del chat sin acceso a Internet.

Salida esperada: El chat no puede establecer la conexión TCP con el *socket* del servidor y emite un error cerrando el intento de conexión y no habilitando los botones para enviar mensajes.

Salida real: El chat no puede establecer la conexión TCP con el *socket* del servidor y emite un error cerrando el intento de conexión y no habilitando los botones para enviar mensajes.

Prueba 2: El usuario intenta enviar un mensaje al chat sin introducir su nombre.

Entrada: La casilla del nombre se encuentra vacía.

Salida esperada: Se le avisa al usuario que ha de introducir el nombre mediante un mensaje y se establece el foco de la aplicación en el campo a rellenar.

Salida real: Se le avisa al usuario que ha de introducir el nombre mediante un mensaje y se establece el foco de la aplicación en el campo a rellenar.

Prueba 3: El usuario intenta enviar un mensaje vacío al chat pero con el campo del nombre rellenado.

Entrada: La casilla mensaje se encuentra vacía.

Salida esperada: Se le avisa al usuario que ha de escribir algo en el mensaje y se establece el foco de la aplicación en el campo a rellenar.

Salida real: Se le avisa al usuario que ha de escribir algo en el mensaje y se establece el foco de la aplicación en el campo a rellenar.

Prueba 4: Para regular el formato del chat se han establecido unos consejos de límites de caracteres que en ningún momento son restrictivos.

Entrada: El usuario introduce un nombre de longitud mayor a quince caracteres.

Salida esperada: Se le avisa al usuario de que la longitud es mayor que la aconsejada, pero se le deja realizar la operación.

Salida real: Se le avisa al usuario de que la longitud es mayor que la aconsejada, pero se le deja realizar la operación.

Prueba 5: Para regular el formato del chat se han establecido unos consejos de límites de caracteres que en ningún momento son restrictivos.

Entrada: El usuario introduce un mensaje de longitud mayor a ochenta caracteres.

Salida esperada: Se le avisa al usuario que la longitud es mayor que la aconsejada, pero se le deja realizar la operación.

Salida real: Se le avisa al usuario que la longitud es mayor que la aconsejada, pero se le deja realizar la operación.

Prueba 6: El usuario intenta enviar un mensaje al chat con ambos campos correctamente rellenados, lo cual debe enviar el mensaje al servidor.

Entrada: El usuario rellena los datos correctamente.

Salida esperada: Se envía el mensaje al servidor, el cual devuelve el mensaje a todos los dispositivos conectados.

Salida real: Se envía el mensaje al servidor, el cual devuelve el mensaje a todos los dispositivos conectados.

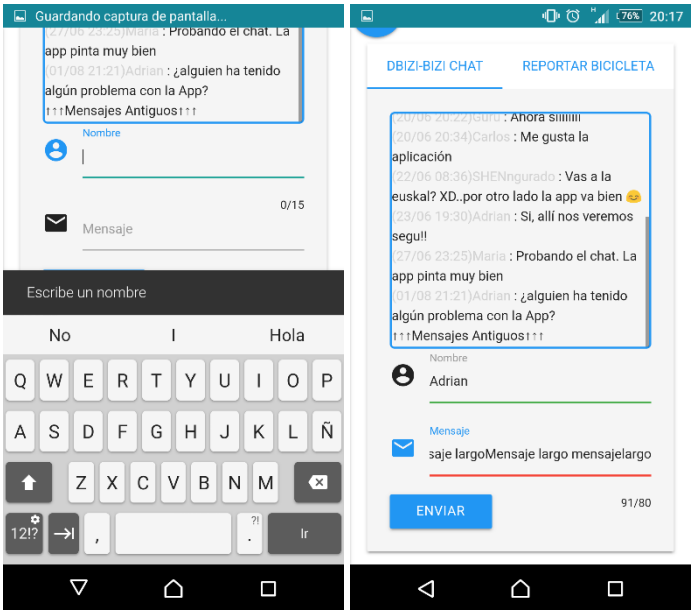


Figura 6.2: Pruebas en el chat

6.1.3 Pruebas en el informe de bicicletas

Esta funcionalidad permite a los usuarios informar sobre el estado de las bicicletas.

Prueba 1: El usuario intenta enviar un informe sin tener ningún acceso a Internet.

Entrada: Intento de enviar un informe sin tener acceso a Internet.

Salida esperada: Si no se detecta conexión a Internet, la aplicación no habilita el botón para enviar los informes.

Salida real: La aplicación no habilita el botón para enviar los informes.

Prueba 2: El usuario intenta enviar un informe sin haberse identificado previamente.

Entrada: La casilla del nombre se encuentra vacía.

Salida esperada: La aplicación muestra un mensaje de error y establece el foco de la aplicación en el campo del nombre.

Salida real: La aplicación no habilita el botón para enviar los informes.

Prueba 3: El usuario envía un informe habiendo rellenado todos los campos salvo el número de la bici.

Entrada: El campo del número de la bici está vacío y el resto correctamente rellenados.

Salida esperada: La aplicación muestra un mensaje para que rellene los campos obligatorios y no envía el informe.

Salida real: La aplicación muestra un mensaje para que rellene los campos obligatorios y no envía el informe.

Prueba 4: El usuario envía un informe habiendo rellenado todos los campos salvo el de información adicional.

Entrada: El campo de información adicional se encuentra vacío y el resto correctamente rellenados.

Salida esperada: La aplicación envía el informe puesto que esa información no es obligatoria y se muestra un mensaje de dicho informe en la ventana del chat.

Salida real: La aplicación envía el informe puesto que esa información no es obligatoria y se muestra un mensaje de dicho informe en la ventana del chat.

Prueba 5: El usuario envía un informe habiendo rellenado correctamente todos los campos.

Entrada: Todos los campos están correctamente rellenados.

Salida esperada: La aplicación envía el informe y se muestra un mensaje de dicho informe en la ventana del chat.

Salida real: La aplicación envía el informe y se muestra un mensaje de dicho informe en la ventana del chat.

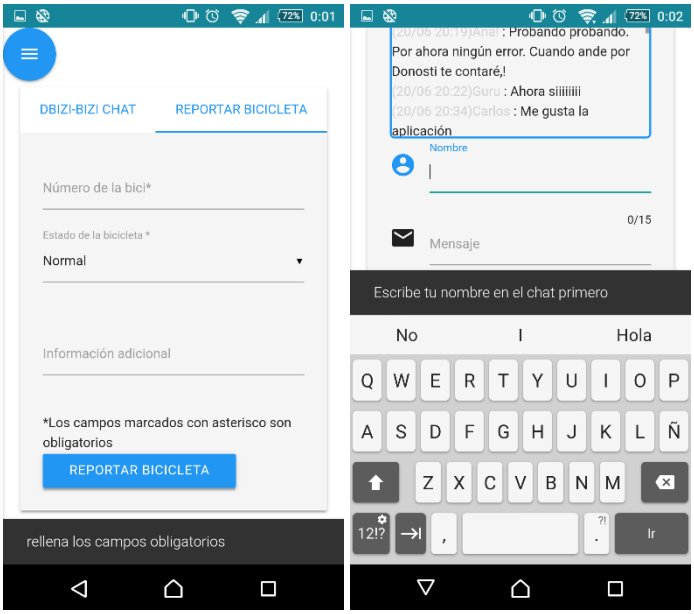


Figura 6.3: Pruebas en los informes

6.1.4 Pruebas en la visualización de los informes de bicicletas

Esta funcionalidad permite a los usuarios visualizar los informes que se hayan emitido sobre el estado de las bicicletas.

Prueba 1: El usuario intenta visualizar un informe sin tener ningún acceso a Internet.

Entrada: Intento de visualizar informe sin tener acceso a Internet.

Salida esperada: Si no se detecta conexión a Internet la aplicación no habilita el botón para visualizar los informes.

Salida real: La aplicación no habilita el botón para visualizar los informes.

Prueba 2: El usuario intenta visualizar un informe de una bicicleta inexistente.

Entrada: Número de una bicicleta inexistente.

Salida esperada: La aplicación muestra la tabla de informes vacía.

Salida real: La aplicación muestra la tabla de informes vacía.

Prueba 3: El usuario intenta visualizar un informe de una bicicleta existente y con varios informes.

Entrada: Número de una bicicleta existente.

Salida esperada: La aplicación muestra la tabla de informes con los últimos cinco informes de dicha bicicleta.

Salida real: La aplicación muestra la tabla de informes con los últimos cinco informes de dicha bicicleta.

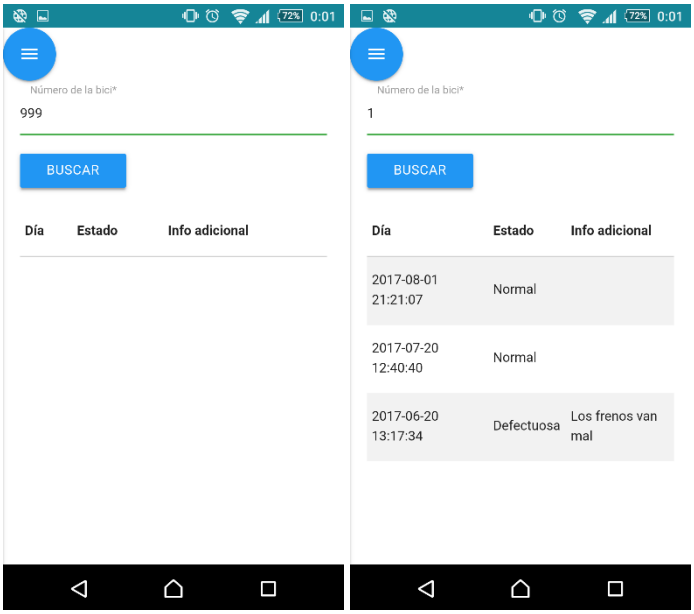


Figura 6.4: Pruebas en los informes

6.1.5 Pruebas en la suscripción a estaciones

Esta funcionalidad permite a los usuarios suscribirse y recibir notificaciones de aquellas estaciones que seleccionen.

Prueba 1: El usuario intenta visualizar sus suscripciones sin tener ningún acceso a Internet.

Entrada: Intento de visualización de suscripciones sin acceso a Internet.

Salida esperada: Mensaje de error avisando al usuario de que no se pudieron obtener dichas estaciones.

Salida real: Mensaje de error avisando al usuario de que no se pudieron obtener dichas estaciones.

Prueba 2: El usuario se suscribe a una de las estaciones.

Entrada: Suscripción a una de las estaciones.

Salida esperada: Se muestra un mensaje de confirmación al usuario y se elimina de la lista de posibles suscripciones la estación a la que se ha suscrito.

Salida real: El usuario recibe el mensaje de confirmación y se elimina de la lista de posibles suscripciones la estación a la que se ha suscrito.

Prueba 3: El usuario se desuscribe de todas las estaciones.

Entrada: Desuscripción de todas las estaciones.

Salida esperada: Se muestra un mensaje de confirmación al usuario, se le elimina de la lista de suscripciones y se renueva la lista de suscripciones posibles.

Salida real: Se muestra un mensaje de confirmación al usuario, se le elimina de la lista de suscripciones y se renueva la lista de suscripciones posibles.

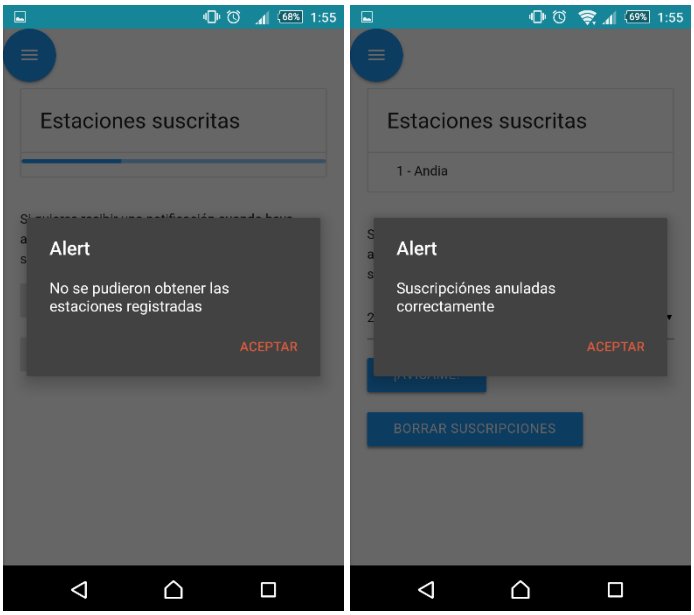


Figura 6.5: Pruebas en las suscripciones

6.2 PRUEBAS EN DISTINTAS PLATAFORMAS

Una de las principales ventajas de usar PhoneGap es la posibilidad de trasladar la aplicación a diferentes plataformas con un esfuerzo mínimo. Por ello, una de las pruebas a realizar es la ejecución de la aplicación en otros sistemas operativos como iOS y Windows Phone puesto que Android ha sido la plataforma utilizada en el resto de pruebas y su funcionamiento ha quedado demostrado.

6.2.1 Windows Phone

Debido a que me ha resultado imposible encontrar un dispositivo real de Windows Phone y a la dificultad que ha supuesto ejecutar un emulador del mismo esta prueba ha tenido que ser descartada. En cualquier caso se puede observar que la aplicación es multiplataforma en el siguiente apartado.

6.2.2 iOS

Para la compilación de aplicaciones en iOS se requiere de un certificado denominado Apple Developer Certificate [35], el cual es de pago y del que se dispone. Por lo tanto, gracias a una

aplicación denominada PhoneGap Developer App se puede lanzar una instancia de la aplicación en cualquier dispositivo iOS sin necesidad de instalación ni del certificado. La aplicación funciona correctamente aunque los tiempos de carga de algunas páginas son mayores que su versión en Android.

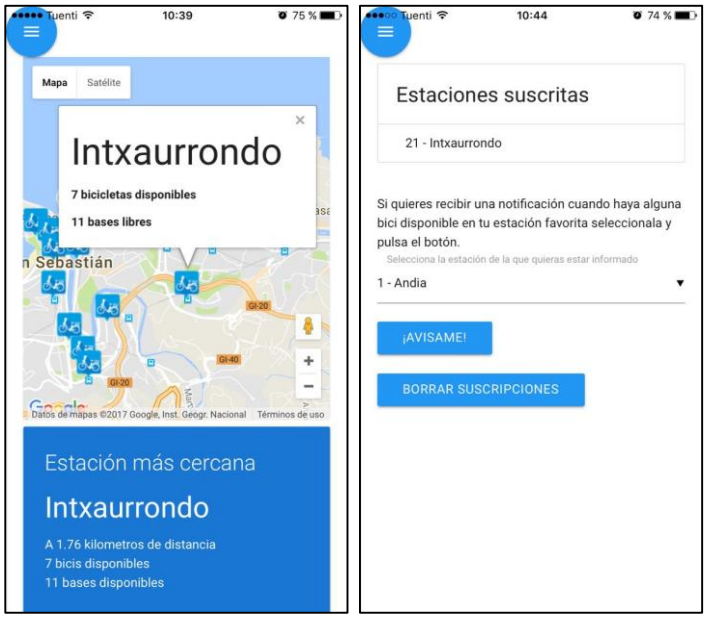


Figura 6.6: dBizi-bizi en iOS

6.3 FORMULARIO DE VALORACIÓN

Durante el desarrollo del proyecto, tal y como se acordó con el cliente, al alcanzarse un prototipo los suficientemente avanzado se procedió a distribuir la aplicación y a realizar una encuesta abierta. Para su distribución se subió la aplicación al Play Store de Google. Así pues, los usuarios tuvieron acceso a la versión final del tercer sprint para que pudiesen hacer una valoración sobre su experiencia y el comportamiento de la aplicación.

Para la obtención del *feedback* de los usuarios se les ofreció la posibilidad de rellenar una encuesta a través de Google Forms [36] la cual rellenaron un total de trece personas. Con esto se pretende conseguir la detección de errores, la mejora de funcionalidades e ideas nuevas para la aplicación.

6.3.1 Encuesta dBizi-bizi

A continuación se muestran alguna de las preguntas y respuestas que se obtuvieron mediante la realización de la encuesta. Sin embargo, en el anexo B puede obtenerse la versión completa de dicha encuesta con sus respectivas respuestas.

6.3.1.1 Preguntas

¿Ves correctamente los iconos de geolocalización de usuario y las estaciones dBizi en el mapa? *

- Sí
- No
- Otra...

¿Añadirías, modificarías o quitarías información en los reportes enviados sobre las bicicletas? *

Texto de respuesta larga

¿Te ha resultado sencilla la navegación? *

- Sí
- No
- Otra...

6.3.1.2 Respuestas

¿Ves correctamente los iconos de geolocalización de usuario y las estaciones dBizi en el mapa?

13 respuestas

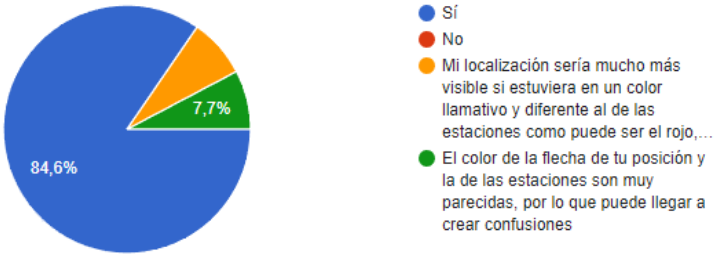


Figura 6.7: Formulario de valoración (versión reducida)

La figura 6.7 muestra tres preguntas con sus correspondientes respuestas pertenecientes a la encuesta que los usuarios pudieron rellenar de forma anónima.

La encuesta fue rellenada por una amplia variedad de personas con distintos niveles de conocimiento sobre el uso de aplicaciones móviles e incluso por personas que han probado aplicaciones similares en otras ciudades como Madrid. De esta manera, se pretende conseguir una respuesta más heterogénea y objetiva. Como puede observarse en las respuestas mostradas en la figura 6.7, la aplicación ha tenido un resultado satisfactorio con un alto nivel de aceptación y una puntuación muy alta.

Las respuestas a la encuesta mostraron ideas de mejora interesantes para la aplicación pero que no pudieron llevarse a cabo en el Sprint cuatro por escasez de tiempo para desarrollar la funcionalidad crucial de dicho sprint.

7 SEGUIMIENTO DEL PROYECTO

En este capítulo se explica cómo se ha realizado el seguimiento y control del proyecto.

Por un lado, se explica cómo ha avanzado el alcance del proyecto y, por otro lado, se exponen los cambios de tiempo sufridos respecto a la planificación inicial. Esto último se recoge en una tabla donde puede verse las horas iniciales estimadas, las reales invertidas y la diferencia entre las mismas.

7.1 GESTIÓN DEL ALCANCE

Debido a la metodología ágil utilizada, Scrum, el alcance inicial de la aplicación ha sufrido algunas variaciones. Esto se debe a que al final de cada Sprint e inicio del siguiente se acordaba el alcance del mismo y, en algunos casos, se estudiaba la viabilidad de algunas funcionalidades y la implementación de nuevas tecnologías.

Dentro del alcance se han añadido las notificaciones *push* a través de la plataforma GCM, las cuales permiten notificar a cualquier dispositivo suscrito sobre la posibilidad de adquirir una bicicleta en una o varias estaciones incluso con la aplicación cerrada o en segundo plano. Por el contrario, se ha eliminado del alcance la inclusión de la plataforma Windows Phone debido a la imposibilidad de realizar prueba alguna de la aplicación en dicho sistema operativo. Por último, cabe destacar que también se ha eliminado la realización de las lecciones aprendidas que se describió en el EDT del proyecto.

7.2 GESTIÓN DEL TIEMPO

La gestión del tiempo ha sido una de las cosas que más alteraciones ha sufrido respecto a la planificación inicial. Esto se debe, principalmente, a que la inclusión de nuevas tecnologías conlleva un tiempo de aprendizaje que muchas veces resulta difícil de cuantificar.

A continuación, se presenta una tabla con las horas estimadas frente a las invertidas según los paquetes de trabajo del esquema EDT mostrado en la figura 2.1.

Paquetes de trabajo			Horas estimadas	Horas invertidas	Diferencia
Gestión	Elección de tecnologías		20	30	+10
	Planificación		10	7	-3
	Estudio de mercado		15	8	-7
	Encuesta de evaluación		5	5	0
Desarrollo	Reuniones		5	4	-1
	Pre-Sprints		30	20	-10
	Sprint 1	Diseño	5	10	+5
		Implementación	10	11	+1
		Pruebas unitarias	1	3	+2
		Total	16	24	+8
	Sprint 2	Diseño	10	20	+10
		Implementación	30	46	+16
		Pruebas unitarias	5	5	0
		Total	45	71	+26
	Sprint 3	Diseño	10	10	0
		Implementación	50	52	+2
		Pruebas unitarias	5	8	+3
		Total	65	70	+5
	Sprint 4	Diseño	5	10	+5
		Implementación	20	44	+22
		Pruebas unitarias	3	8	+5
		Total	28	62	+34
	Pruebas integrales		10	20	+10
Documentación	Memoria		40	45	+5
Defensa del proyecto			10	10	0
Horas totales			299	376	+77

Tabla 7.1: Gestión del tiempo

Por un lado, cabe destacar el aumento de tiempo dedicado al Sprint dos, el cual se debió al tiempo de configuración del servidor en Amazon. Al ser la primera vez que se utilizaba dicha plataforma (Amazon Cloud Computing), se esperaba una implementación sencilla y rápida. En cambio, la configuración de dichos servidores fue más costosa de lo esperado. No obstante, ello conllevó a establecer a un nivel de seguridad mayor en el servidor. Por otro lado, el aumento del tiempo dedicado en el Sprint cuatro se debió a la utilización de un *pluggin* cuya versión dificultó la implementación del mismo. Sin embargo, ninguno de estos dos aumentos de tiempo retrasó los plazos de entrega de los Sprints. Por último, cabe señalar la pausa tomada en el momento de desarrollo de la documentación entre los días 21 de julio y 10 de agosto. Para reflejarlo se ha reestructurado el diagrama de Gantt quedando de la siguiente manera:

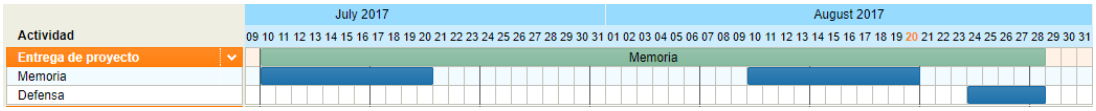


Figura 7.1: Diagrama de Gantt reestructurado - Entrega de proyecto

8 CONCLUSIONES Y LÍNEAS FUTURAS

En este último capítulo se recogen las conclusiones obtenidas a lo largo del desarrollo del proyecto a la vez que se realiza una valoración personal sobre las tecnologías utilizadas y su implantación en la aplicación. Asimismo, se tratan las líneas futuras de dBizi-bizi puesto que los proyectos de software necesitan de un mantenimiento y actualización constante para competir en el mercado.

8.1 CONCLUSIONES DEL PROYECTO

Aunque la metodología escogida no era la más adecuada teniendo en cuenta las limitaciones de equipo y cliente ha resultado ser tremendamente instructiva a la par que eficaz.

El hecho de usar una metodología ágil, establecer los criterios de cada iteración al principio de la misma y conseguir un resultado al final del Sprint ha provocado motivación y ha impulsado a continuar desarrollando el producto ya que los resultados son visibles y tangibles. El poder establecer periodos cortos para los desarrollos del Sprint facilita un progreso continuo y propician la detección de dificultades durante la implementación de la aplicación.

No obstante, la utilización de tantas tecnologías nuevas y desconocidas ha supuesto un hándicap que, sin embargo, se ha logrado superar. Se ha conseguido un producto con muchas tecnologías distintas pero con sinergia entre ellas.

PhoneGap ha resultado ser una tecnología de fácil acceso pero de difícil profundización debido a la gran cantidad de versiones y diferencias entre las mismas. Todo ello junto a los *pluggins* que la soportan hacen que esta herramienta presente tanto ventajas como desventajas como puede ser el caso de incompatibilidad de *pluggins* no actualizados.

Los objetivos prioritarios establecidos con el cliente sobre la aplicación han sido cumplidos y, tras los buenos resultados obtenidos en la encuesta de valoración, se puede concluir que se ha desarrollado el proyecto de forma exitosa.

8.2 CONCLUSIONES PERSONALES

La propia coyuntura del proyecto me ha impulsado a usar tecnologías que me eran completamente desconocidas y que no se dan en la carrera. Sin embargo, esto resulta positivo tanto para la ampliación de mis conocimientos como para mi preparación hacia el mundo laboral.

Asimismo, este proyecto me ha permitido acotar de forma más precisa mis límites en áreas nuevas, tal y como he podido observar en los aumentos de tiempo de desarrollo de algunas funcionalidades.

Por último, el desarrollo de un proyecto de esta índole me ha aportado experiencia y me ha proporcionado una sólida base para continuar mi progreso en este campo.

8.3 MEJORAS Y LÍNEAS FUTURAS

Aunque el proyecto haya llegado a su final dentro del marco académico, dado que las funcionalidades acordadas están implementadas y el tiempo del Trabajo de Fin de Grado es finito, comienza otra etapa en el ciclo de vida del proyecto puesto que la intención es continuarlo.

dBizi-bizi es una aplicación en funcionamiento con utilidad real y, como tal, continuará creciendo tanto respecto a su funcionalidad como en número de usuarios. Para continuar con ese crecimiento existen actualmente ideas que se irán realizando en un futuro.

Entre las mejoras que se contemplan a corto plazo están:

- Mejorar el tiempo de carga y la visualización del mapa mediante la inclusión de algún pluggin que cargue el mapa desde una ruta local y no desde Internet como lo hace Google Maps.
- Añadir un cronómetro que notifique al usuario mediante un aviso acústico de que el tiempo de uso gratuito de la bicicleta (quince minutos) está llegando a su fin.
- Añadir un aviso cuando un usuario intente utilizar la aplicación dBizi-bizi fuera del horario de funcionamiento del servicio. Esto es especialmente complicado porque en los días de fiesta y otras fechas señaladas se altera el horario normal del servicio.
- Inclusión de un sistema enlazado a Twitter y otras redes sociales que muestre las noticias más relevantes del propio sistema dBizi.

Entre las mejoras a largo plazo están:

- Optimización de la aplicación en otras plataformas como iOS y Windows Phone, las cuales por falta de recursos no han podido ser completamente optimizadas.
- Inclusión de anuncios para financiar los gastos generados por el mantenimiento del servidor en la plataforma de Amazon Cloud Computing que solo es gratuita durante el primer año de funcionamiento.

9 BIBLIOGRAFÍA

[1] Google Play Store. dBizi. URL:

https://play.google.com/store/apps/details?id=net.sareweb.android.dBizi

[Internet; visitado 4-04-2017]

[2] Google Play Store. biciMad. URL:

https://play.google.com/store/apps/details?id=com.bonopark.bicimad&hl=en

[Internet; visitado 5-03-2017]

[3] Google Play Store. Cicleta. URL:

https://play.google.com/store/apps/details?id=com.durbon.cicleta

[Internet; visitado 6-04-2017]

[4] Google Play Store. Madbike. URL:

https://play.google.com/store/apps/details?id=org.drunkcode.madbike

[Internet; visitado 7-04-2017]

[5] Jquery mobile. https://jquerymobile.com/ [Internet; visitado 10-04-2017]

[6] Jquery Foundation. https://js.foundation/ [Internet; visitado 12-04-2017]

[7] AJAX. https://en.wikipedia.org/wiki/Ajax_(programming) [Internet; visitado 12-04-2017]

[8] Software widget. https://en.wikipedia.org/wiki/Software_widget [Internet; visitado 12-04-2017]

[9] Ionic *Framework*. https://ionicframework.com/ [Internet; visitado 12-04-2017]

[10] Cordova *Framework*. https://cordova.apache.org/ [Internet; visitado 13-04-2017]

[11] MVC. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller [Internet; visitado 21-04-2017]

[12] Ionic creator. https://creator.ionic.io [Internet; visitado 12-04-2017]

[13] Nitobi. https://github.com/nitobi [Internet; visitado 13-04-2017]

[14] Adobe Systems. http://www.adobe.com/ [Internet; visitado 13-04-2017]

[15] PhoneGap. https://phonegap.com/ [Internet; visitado 13-04-2017]

[16] PHP. https://en.wikipedia.org/wiki/PHP [Internet; visitado 14-04-2017]

[17] Licencia MIT. https://en.wikipedia.org/wiki/MIT_License [Internet; visitado 14-04-2017]

[18] Licencia New BSD. https://en.wikipedia.org/wiki/BSD_licenses [Internet; visitado 15-04-2017]

[19] PhoneGap build. https://build.phonegap.com [Internet; visitado 13-04-2017]

[20] Scrum. https://www.scrum.org/resources/what-is-scrum [Internet; visitado 18-07-2017]

[20] Sprint. https://www.scrum.org/resources/what-is-a-sprint-in-scrum [Internet; visitado 21-05-2017]

[22] EDT / WBS. http://www.workbreakdownstructure.com/ [Internet; visitado 20-04-2017]

[23] Tomsplanner. https://www.tomsplanner.com [Internet; visitado 15-08-2017]

[24] Historia de usuario.

http://www.scrummanager.net/bok/index.php/Historia_de_usuario

[Internet; visitado 18-07-2017]

[25] MySQL. https://www.mysql.com/ [Internet; visitado 22-04-2017]

[26] JSON. http://www.json.org/ [Internet; visitado 21-04-2017]

[27] Base de datos relacional.

https://es.wikipedia.org/wiki/Base_de_datos_relacional [Internet; visitado 22-04-2017]

[28] InnoDB. https://dev.mysql.com/doc/refman/5.7/en/innodb-storage-engine.html [Internet; visitado 19-05-2017]

[29] myISAM. https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html [Internet; visitado 19-05-2017]

[30] Notificaciones push. https://es.wikipedia.org/wiki/Notificaci%C3%B3n_push [Internet; visitado 13-06-2017]

[31] WebSocket. https://en.wikipedia.org/wiki/WebSocket [Internet; visitado 06-06-2017]

[32] GCM. https://developers.google.com/cloud-messaging/ [Internet; visitado 29-06-2017]

[33] DOM. https://en.wikipedia.org/wiki/Document_Object_Model [Internet; visitado 14-04-2017]

[34] cURL. https://en.wikipedia.org/wiki/CURL [Internet; visitado 05-06-2017]

[35] Apple Developer Certificate. https://developer.apple.com/support/certificates/ [Internet; visitado 06-07-2017]

[36] Google forms. https://www.google.com/forms/about/ [Internet; visitado 19-06-2017]

10 ANEXOS

10.1 A: ACTAS DE REUNIÓN

ACTA REUNIÓN DE TRABAJO

Acta Nº 1 Fecha: 5-4-2017

Asistentes:

-Adrián González

-Alfredo Goñi

Lugar: Despacho del director de proyecto Alfredo Goñi.

Orden del día:

- Discutir sobre el modelo de desarrollo que se llevará a cabo durante el proyecto.
- Discutir los *frameworks* y lenguajes de programación que se utilizarán durante el desarrollo.

Acuerdos y decisiones tomadas:

1. Se seguirá el modelo de desarrollo ágil Scrum en el proyecto aunque no se disponga de un grupo de trabajo. El director de proyecto hará también de cliente.
2. Se ha decidido dedicar un tiempo a la documentación y estudio de los diferentes lenguajes y *frameworks* que pueden ser utilizados en el proyecto.

ACTA REUNIÓN DE TRABAJO

Acta Nº 2 Fecha: 24-4-2017

Asistentes:

-Adrián González

-Alfredo Goñi

Lugar: Despacho del director de proyecto Alfredo Goñi.

Orden del día:

- Exponer las tecnologías y *frameworks* decididos para utilizarse en el TFG (PhoneGap y CodeIgniter).
- Escoger las historias de usuario que entrarán dentro del Release Backlog.
- Escoger las historias de usuario que se desarrollarán durante el Sprint uno.

Acuerdos y decisiones tomadas:

1. La tecnología utilizada para el desarrollo de la aplicación cliente será PhoneGap y la del servidor CodeIgniter.
2. Quedán establecidas las historias de usuario para el Sprint uno:
    a. Número 1 – Ver las estaciones de las bici.
    b. Número 2 – Menú en la aplicación.

ACTA REUNIÓN DE TRABAJO

Acta Nº 3 Fecha: 8-5/2017

Asistentes:

-Adrián González

-Alfredo Goñi

Lugar: Despacho del director de proyecto Alfredo Goñi.

Orden del día:

-Mostrar el progreso del desarrollo y finalizar el Sprint uno.

-Escoger las historias de usuario que se desarrollarán durante el Sprint dos.

-Exponer la idea de la utilización de un *framework* CSS y JavaScript que mejore la calidad visual de la aplicación.

Decisiones y acuerdos tomados:

1. Quedan establecidas las historias de usuario para el Sprint dos:
    a. Número 3 – Obtener datos de las estaciones.
    b. Número 4 – Saber la estación más cercana.
2. Se utilizará el *framework* Materialize para mejorar la calidad visual del producto. http://materializecss.com/

ACTA REUNIÓN DE TRABAJO

Acta Nº 5 Fecha: 30-5-2017

Asistentes:

-Adrián González

-Alfredo Goñi

Lugar: Despacho del director de proyecto Alfredo Goñi.

Orden del día:

-Mostrar la evolución del proyecto, en concreto la fase final del Sprint dos.

-Establecer las historias de usuario que se realizarán en el Sprint tres.

Decisiones y acuerdos tomados:

1. Establecidos las historias de usuario para el Sprint tres:
    a. Número 6 – Chat con otros usuarios.
    b. Número 15 – Reportar el estado de una bicicleta.
    c. Número 16 – Ver el estado de una bicicleta.
2. Realizar un estudio para la siguiente reunión a fin de ver la viabilidad de establecer notificaciones push en el Sprint cuatro.
3. Se ha establecido que entre el Sprint tres y cuatro se realizará una encuesta que será enviada a personas con distintos perfiles, tanto usuarios como no usuarios de dBizi, para valorar su opinión. Estas encuestas se realizarán entre el 19 y el 26 de junio.

ACTA REUNIÓN DE TRABAJO

Acta Nº 6 Fecha: 19-6-2017

Asistentes:

-Adrián González

-Alfredo Goñi

Lugar: Despacho del director de proyecto Alfredo Goñi.

Orden del día:

-Mostrar la evolución del proyecto, en concreto la fase final del Sprint tres.

-Establecer las historias de usuario que se realizarán en el Sprint cuatro.

Decisiones y acuerdos tomados:

1. Quedan establecidas las historias de usuario para el Sprint cuatro:
    a. Número 11 – Recibir alertas de las estaciones libres.

10.2 B: ENCUESTA COMPLETA

10.2.1 Preguntas

Encuesta dBizi-bizi

Me ayudaría mucho saber tu opinión, gracias.

¿Ves correctamente los iconos de geolocalización de usuario y las estaciones dBizi en el mapa? *

- Sí
- No
- Otra...

¿Te parece adecuada la forma en la que se muestra la información de las estaciones? *

- Sí
- No
- Otra...

¿Ves correctamente el chat?¿Cambiarías o añadirias algo? *

Texto de respuesta larga

¿Mostrarías la hora de los mensajes enviados en el chat? *

- Sí
- No
- Otra...

¿Y la fecha de los mensajes? *

- ◯ SI
- ◯ No

¿Añadirías, modificarías o quitarías información en los reportes enviados sobre las bicicletas? *

Texto de respuesta larga

¿Te ha resultado sencilla la navegación? *

- ◯ Sí
- ◯ No
- ◯ Otra...

¿Cambiarías algo de las funciones existentes actualmente? *

Texto de respuesta larga

¿Qué funciones adicionales te gustaría ver en la aplicación? *

Texto de respuesta larga

Si quieres comentar cualquier otra cosa este es el lugar: *

Texto de respuesta larga

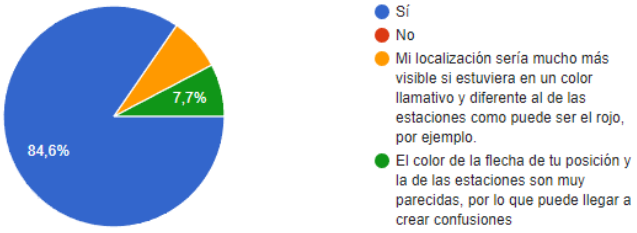
En general, ¿te ha gustado la aplicación? Puntúala del 1 al 5 siendo 1 la menor nota y 5 la mayor. *

1	2	3	4	5
◯	◯	◯	◯	◯

10.2.2 Respuestas

¿Ves correctamente los iconos de geolocalización de usuario y las estaciones dBizi en el mapa?

13 respuestas



¿Te parece adecuada la forma en la que se muestra la información de las estaciones?

13 respuestas

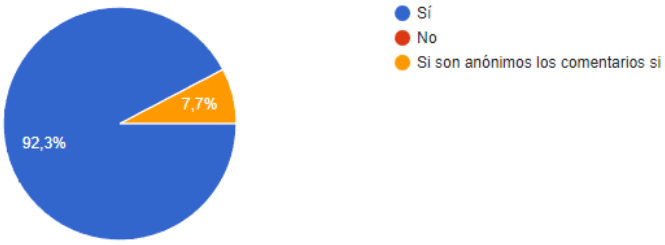
¿Ves correctamente el chat?¿Cambiarías o añadirias algo?

11 respuestas

No.
Lo veo correctamente, lo iría adaptando a las necesidades si se une mucha gente
Si me parece una gran idea
Veo correctamente.
Sí. No, me parece correcto
Todo bien
No
Lo pondría con un icono mas claro. No lo encontraba
No me parece muy bueno
No, lo veo todo correctamente
Cambiaría la forma de mostrar día, hora, nombre y mensaje, cuesta un poco de leer los mensajes

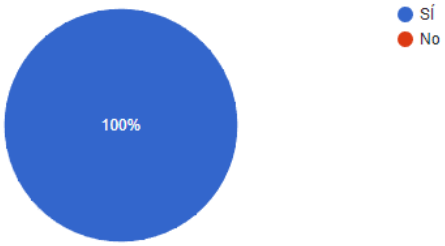
¿Mostrarías la hora de los mensajes enviados en el chat?

13 respuestas



¿Y la fecha de los mensajes?

12 respuestas



¿Añadirías, modificarías o quitarías información en los reportes enviados sobre las bicicletas?

11 respuestas

No (6)
No.
Quitaria
No. El icono masclaro. Igual.algo de identificación, número de carnet de alquiler?eso igual.sale y no me he fijado
No,es correcto
Eso no he probado

¿Te ha resultado sencilla la navegación?

13 respuestas

¿Cambiarías algo de las funciones existentes actualmente?

10 respuestas

No (6)
En la pantalla de inicio sería útil poder ver todas las posibilidades desde el inicio sin necesidad de tener que desplazar la pantalla hacia abajo para ver algunas puesto que pueden pasar desapercibidas. Ejemplo de ello es la función de "mostrar" la estación más cercana.
Que se pudiera desactivar la localizacion
No
Sencilla y práctica, me gusta

¿Qué funciones adicionales te gustaría ver en la aplicación?

9 respuestas

El tiempo restante o límite, por ejemplo.
No se me ocurren
Previsión de disponibilidad de bicicletas en la próxima hora en relación al histórico de esa parada. Posibilidad de saber horas puntas basadas en el historial.
Nada que añadir
La información está clara. No cambiaría nada.
Poder poner que te acercas a coger una bici a una estación y que te vise si se coge y no queda ninguna cuando voy de camino
Ninguna
Lo veo bien asi
Por aportar... Además de la estación más cercana daría la posibilidad de buscar dónde quedan bicis lo más cerca de mi posición.

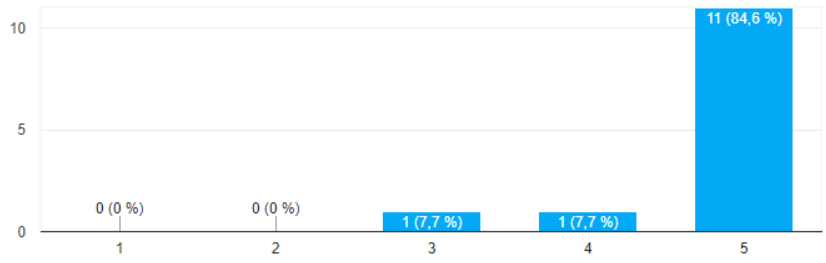
Si quieres comentar cualquier otra cosa este es el lugar:

7 respuestas

La posibilidad de poder iniciar sesión podría ayudar a llevar un seguimiento de faltas, tiempo o km recorridos, entre otras cosas.
Todo bien
No
Eres un crack
Me parece una idea genial!muy bien para los usuarios!
No tengo nada para comentar
Es muy buena idea

En general, ¿te ha gustado la aplicación? Puntúala del 1 al 5 siendo 1 la menor nota y 5 la mayor.

13 respuestas



10.3 C: MANUAL DE USUARIO

El siguiente manual pretende facilitar la experiencia de cualquier usuario que desee utilizar la aplicación. En este se indican las funcionalidades, su posible uso y varios datos relevantes sobres las mismas.

10.3.1 Adquisición de la aplicación

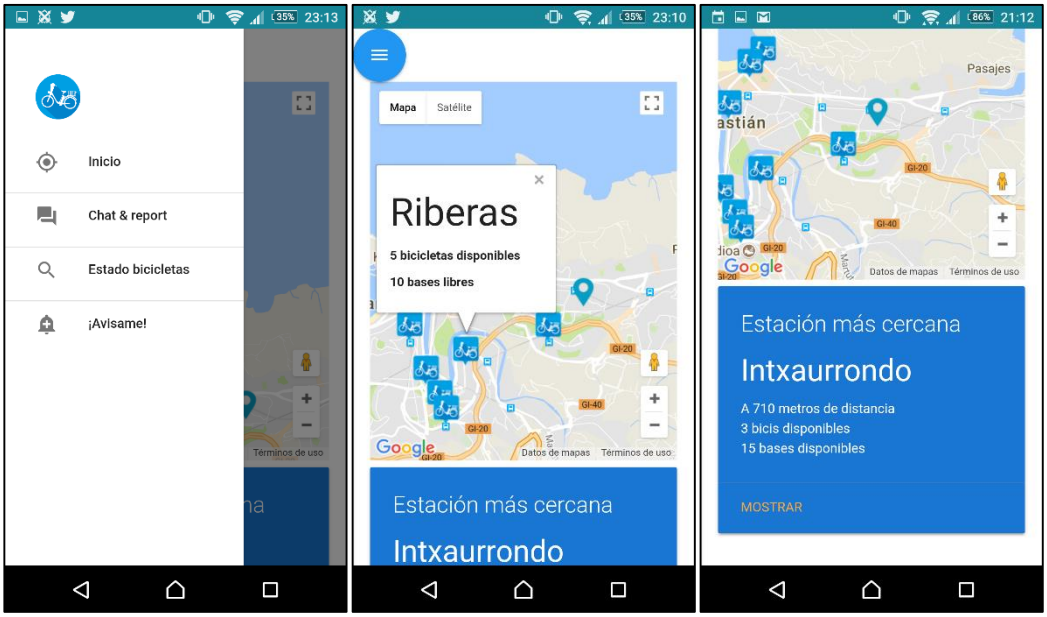
Esta aplicación puede adquirirse para Android a través de la tienda oficial de aplicaciones, Google Play, o mediante el siguiente código QR.

10.3.2 Visualizar Mapa

Esta funcionalidad permite al usuario orientarse a través de la ciudad ya que muestra un mapa de Donostia con las estaciones de dBizi marcadas y la localización geográfica del propio usuario.

El usuario puede hacer uso de los gestos típicos de estos mapas (pinchar, alejar, arrastrar y clicar) para moverse por el mapa y orientarse mejor en caso de que fuese necesario.

Asimismo, tiene la opción de mostrar la estación dBizi más cercana a su ubicación haciendo clic en el botón “mostrar estación más cercana” situado en la parte inferior de la pantalla principal.

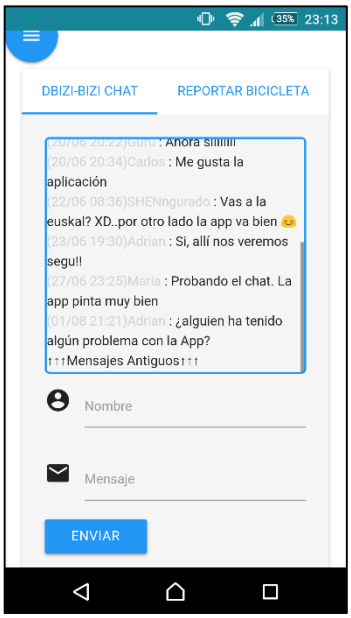


10.3.3 Chatear

El chat permite a los usuarios mantenerse informados sobre los hechos más recientes del sistema. Puede avisarse sobre estaciones que están en mantenimiento, experiencias con las bicicletas o el servicio o preguntar cualquier tipo de duda que a un usuario pueda surgirle y otro sepa ayudarle.

Para conectarse al chat hay que acceder al menú lateral y escoger la segunda opción “chat & report”. Se cargará una pantalla que se conectarán directamente al chat para ver los mensajes más recientes.

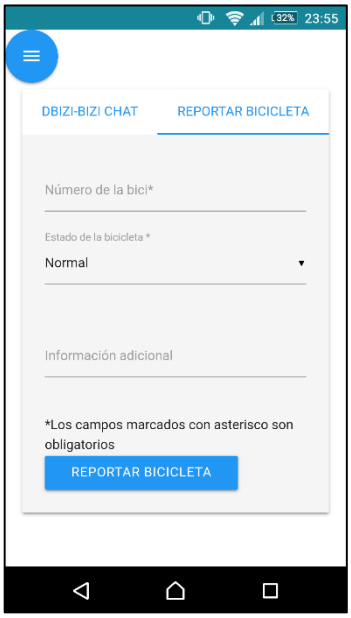
En caso de querer enviar un mensaje, tendremos que rellenar la casilla del nombre seguido de nuestro mensaje y pulsar enviar. A partir de entonces la casilla de nombre se ocultará y no será necesario volver a rellenarla hasta que nos salgamos del chat.



10.3.4 Reportar bicicleta

Esta funcionalidad permite a cualquiera valorar su experiencia o grado de satisfacción con una bicicleta en concreto y dejar constancia de ello para cualquiera que desee visualizar las valoraciones concretas de una bicicleta.

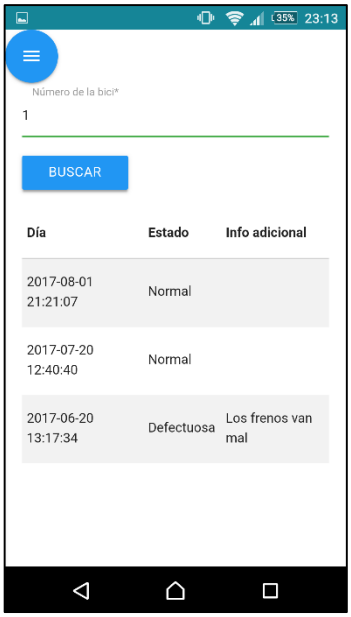
Para reportar una bicicleta se selecciona la segunda opción del menú lateral “chat & report” y, después, la pestaña “reportar bicicleta” situada en la parte superior de la pantalla. Tras rellenar los campos obligatorios podremos enviar dicho reporte[7] y se mostrará un aviso del reporte enviado en la ventana del chat para que los usuarios conectados puedan estar al tanto de las noticias más recientes.



[7] En caso de que no se haya identificado antes, la aplicación pedirá que lo haga.

10.3.5 Consultar estado de bicicletas

Mediante esta funcionalidad un usuario puede consultar el historial del estado de cualquier bicicleta. Para ello, se selecciona la tercera opción del menú lateral denominada “Estado bicicletas”, se rellena el número de la bici que se desea consultar y se presiona el botón “buscar”. En caso de que hubiera reportes sobre dicha bicicleta solo aparecerán los cinco más recientes.



10.3.6 Suscribirse a estaciones

En caso de querer recibir una notificación en el móvil cuando una de las estaciones escogidas tenga, al menos, una bicicleta libre para poder usarse se recurrirá a la cuarta opción del menú denominada “¡Avísame!”.

En la página se cargará una lista de las estaciones a las que el dispositivo está suscrito en caso de que estuviera suscrito a alguna. Sino, aparecerá una lista con las estaciones a las que puede suscribirse. Se selecciona la estación a la que se quiere suscribir y se presiona el botón “¡Avísame!”. Por el contrario, si uno no se quiere recibir más notificaciones se pulsará el botón “Borrar suscripciones” y se dejará de estar suscrito a cualquier estación a la que se estuviese suscrito.