Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

El servicio de alquiler de patinetes eléctricos: PatiVal

Trabajo fin de grado

Grado en Ingeniería Informática

Autor: Imanol Carbonell Márquez

Tutora: Manuela Albert Albiol

Cotutora: María Victoria Torres Bosch

2018-2019

Resum

El servei de lloguer de patinets elèctrics: PatiVal és un projecte d'aplicació mòbil híbrida construïda amb el framework IONIC pensada per a grans ciutats on la xarxa d'usuaris puguen reservar i consultar informació dels patinets. Per això, i sempre amb informació en temps real, permetrà oferir i obtindre informació personalitzada per a cada usuari. Per tant es tracta d'una aplicació completa per a un projecte real.

Paraules clau: lloguer, patinet, elèctric, mòbil, app, mobilitat, urbana, Full-Stack, frontend, backend, FireBase, Ionic, web.

Resumen

El servicio de alquiler de patinetes eléctricos: PatiVal es un proyecto de aplicación móvil híbrida construida con el framework IONIC pensada para grandes ciudades donde la red de usuarios pueda reservar y consultar información de los patinetes. Por esto, y siempre con información en tiempo real, permitirá ofrecer y obtener información personalizada para cada usuario. Por lo tanto, se trata de una aplicación completa para un proyecto real.

Palabras clave: alquiler, patinete, electrico, móvil, app, mobilidad, urbana, Full-Stack, frontend, backend, FireBase, Ionic, web.

Abstract

The electric scooter rental service: PatiVal is a hybrid mobile application project built with the IONIC framework designed for large cities where the network of users can reserve and consult information on the scooters. For this reason, and always with real time information, it will allow to offer and obtain personalized information for each user. Therefore, it is a complete application for a real project.

Keywords: rental, scooter, electric, phone, app, mobility, urban, Full-Stack, frontend, backend, FireBase, Ionic, web.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1
Introducción

En los tiempos que corren, la necesidad de transporte y de moverse rápidamente de un lugar a otro es cada vez más y más pedido por la gente. Sobre todo, en las grandes ciudades donde cualquier trayecto puede hacerse muy largo o puedes tardar mucho de tiempo yendo con un vehículo tradicional. Entonces es por eso que los nuevos medios de transporte portables y ligeros cada vez son más utilizados por todo el mundo por su facilidad de carga, comodidad y su facilidad de aparcamiento en destinos.

Por eso, este proyecto va destinado a todas las personas que quieran emplear patinetes eléctricos en su día a día para moverse a cualquier momento del día. El usuario mediante una interfaz gráfica amigable podrá sin esfuerzo hacer y realizar

Dicho proyecto, consiste en una parte frontend desarrollada con el framework Ionic y Angular y una parte backend hecho mediante el servicio de bases de datos en tiempo real Firebase, que conectará la base de datos remota con la parte frontend. Ambas partes de la aplicación desarrolladas con tecnologías que puedan hacer de esta una aplicación híbrida, es decir, que funcione tanto para el sistema operativo iOS como para Android.

1.1 Motivación

El motivo principal por el cual decidí proponer y realizar este TFG fue poder trabajar en un proyecto que estuviera conectado con la realidad y las necesidades de la gente, sobre todo joven, que se mueve mucho por los pueblos y ciudades y vuelan un medio de transporte fácil, rápido y respetuoso con el medio ambiente. Además, la temática me resulta muy interesante porque soy usuario de este tipo de transporte como del transporte público y que nos ayuda a desplazarnos de un lugar a otro sin muchas complicaciones y a un precio accesible por todo el mundo, puesto que los patinetes eléctricos son todavía caros por la mayor parte de la gente corriente trabajadora.

Desde un punto de vista tecnológico también me interesaba un proyecto completo donde hubiera su parte de frontend como de backend aunque con Firebase esta parte se simplifica mucho, incluso a ser casi inexistente, además de ser una tecnología muy empleada actualmente. Al mismo tiempo también que usara Ionic, ya que me ha permitido aprender más sobre el desarrollo híbrido de aplicaciones y que utilizara un framework de Javascript como Angular, porque ya tenía ciertos conocimientos sobre él, y utilizar un framework como Ionic me ha permitido un aprendizaje basta más rápido.

1.2 Objetivos

El objetivo fundamental de esta aplicación es poder ofrecer información en tiempo real de manera muy estructurada al usuario sobre los patinetes disponibles a cada estación además de la posibilidad de reservar y poder gestionar sus datos personales. Esta aplicación ha sido resultado de una inquietud y necesidad mía como usuario de este tipo de mediados de transporte y como persona que vive en una gran ciudad como Barcelona y por esto se pretende que el diseño de la interfaz sea fácil y sencillo para que cualquier persona pueda de manera intuitiva utilizarla sin tener experiencia previa en otro tipo de aplicación de parecido concepto y funcionamiento.

1.3 Estructura de la memoria

- Introducción: se hace una primera explicación a grandes rasgos de la aplicación, se exponen los principales motivos y el que se espera conseguir con la realización del proyecto para obtener una imagen general que nos ayudará a entender los siguientes apartados.

- Estado del Arte: en este capítulo se hace un estudio de los productos parecidos en el mercado enfocados en el servicio de alquiler de bicicletas a dos ciudades grandes como Valencia y Barcelona y sus características principales.

- Análisis de los requisitos: se expondrá las características que tendrá la aplicación, mediante el análisis de requisitos del estándar IEEE 830 además de los casos de uso de la aplicación y finalmente una muestra de los mockups o borradores realizados para el proyecto.

- Diseño: este apartado está enfocado a explicar cómo se ha implementado la solución final y las tecnologías utilizadas en la aplicación, su estructura y las consideraciones a tener en cuenta.

- Ejecución de los casos de uso: se mostrarán ejemplos de las diferentes secuencias de uso del usuario de la aplicación.

- Conclusiones: se comenta diferentes valoraciones sobre la realización del proyecto, además de exponer los problemas surgidos a lo largo del desarrollo y se propone cambios futuros.

Capítulo 2
Estado de Arte

En el mercado de aplicaciones móviles existente tanto en Google Play como en el Apple Store hay muchas aplicaciones relacionadas sobre todo con el alquiler de coches o bicicletas, algunas de ellas incluso no oficiales, que nos ofrecen un servicio de alquiler completo o simplemente nos ofrecen información sobre la disponibilidad o estado de los transportes a alquilar por la dicha aplicación. Todas ellas con una interfaz de usuario a veces más claras y a veces más complicadas. A continuación, analizaremos dos aplicaciones que he usado en mi día a día de parecido estilo pero diferente temática. Me han aportado mucha información sobre cómo hacer el diseño de la interfaz gráfica y que han sido un referente y una inspiración para realizar este proyecto: Bicing y Valencia Valenbisi, una es la aplicación oficial de reserva e información de bicicletas públicas de la ciudad de Barcelona y la otra una aplicación no oficial pero que bebe de los datos oficiales del servicio ValenBisi.

2.1 Bicing

Bicing es el servicio de bicicletas públicas de la ciudad de Barcelona inaugurado a inicios del 2007 que para poder utilizar el servicio hay que darse de alta por Internet o bien presencialmente en la oficina de Bicing, y es necesario presentar una tarjeta de crédito junto con el documento de identidad, aun así, los jóvenes entre 16 y 18 años necesitan una autorización firmada por su padre/madre o tutor legal que tendrán que entregar en la oficina Bicing.

Las bicicletas se pueden encontrar en cualquier las estaciones de la ciudad, situadas a puntos próximos de estaciones de metro, tren, ferrocarriles de la Generalitat de Cataluña y aparcamientos públicos donde más o menos la distancia entre las estaciones es de unos 400 metros.

La App del servicio Bicing tiene las siguientes funcionalidades principales:

- Consultar las estaciones en el mapa y mostrar la disponibilidad de bicicletas y anclajes en cada una de ellas en tiempo real.
- Coger una bicicleta, comprobar que la has anclado correctamente o pedir 10 minutos extras si la estación está llena.
- Planificar tus recorridos.

Una vez bajada de Google Play o del Apple Store solo hay que seguir los siguientes pasos para utilizarla.

Figura 2.1: Funcionamiento de Bicing

A continuación, se mostrará el contenido de la aplicación donde podremos ver las diferentes funcionalidades que ofrece al usuario:

- Inicio:

  Después de una pantalla inicial de carga (Figura 2.2), donde podremos ver la publicidad del ayuntamiento de Barcelona y el logotipo de Bicing se muestra el mapa de Barcelona donde están las estaciones de bicicletas que podremos encontrar dispersiones por toda la ciudad (Figura 2.3).

Figura 2.2: Pantalla de inicio

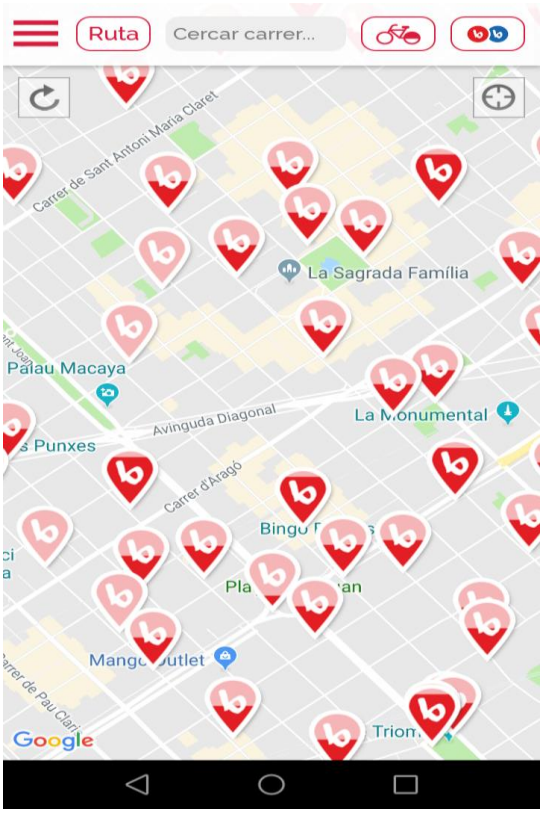

Figura 2.3: Pantalla principal

- Mapa:

  Haciendo clic a cualquier icono con el logotipo de Bicing (Figura 2.4) podremos seleccionar una estación de bicicletas donde podremos ver la cantidad de bicicletas libres, tanto las mecánicas como las eléctricas, además de la cantidad de aparcamientos disponibles en la estación elegida.

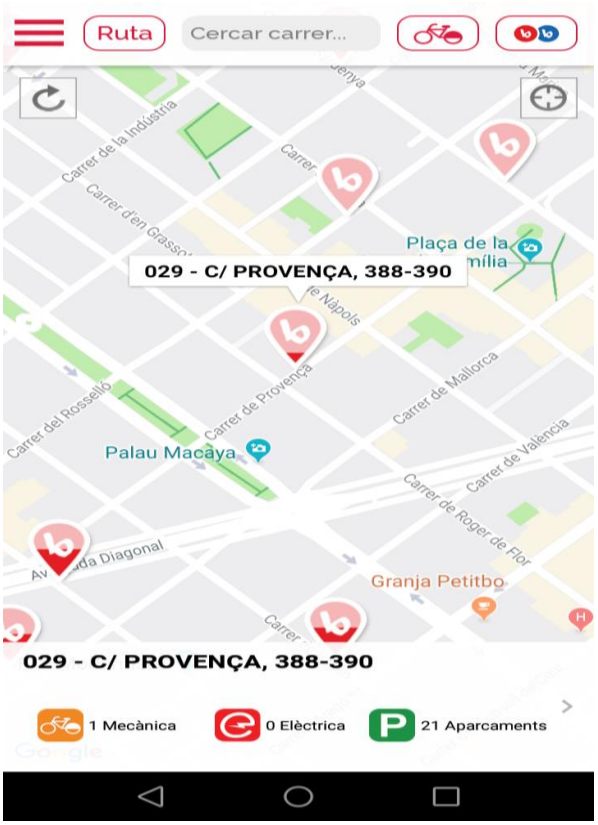

Figura 2.4: Mapa de estaciones

- Menú:

Si pulsamos el icono de la parte izquierda superior de la aplicación se nos abrirá un menú lateral (Figura 2.5) donde podremos encontrar cinco opciones posibles que nos darán las diferentes funcionalidades disponibles en la aplicación.

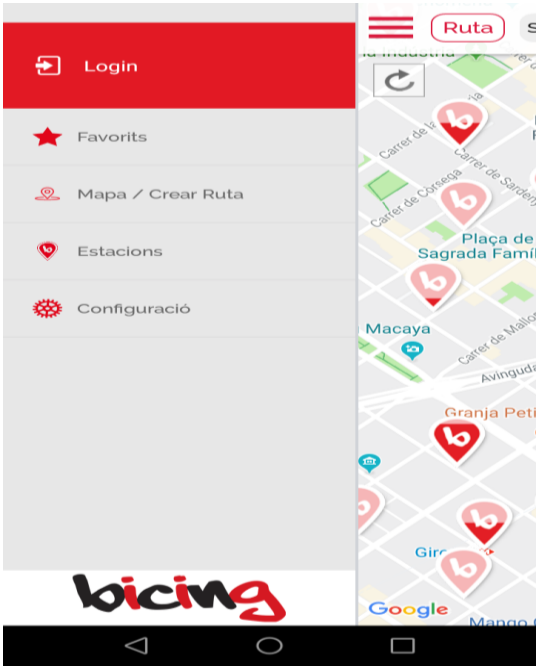

Figura 2.5: Menu

- Login:

En este apartado de la aplicación introduciremos nuestro correo electrónico y contraseña que previamente hemos grabado en el web del servicio Bicing (Figura 2.6).

Figura 2.6: Login

- Estaciones favoritas:

  Aquí podremos guardar nuestras estaciones más usadas o frecuentadas, entonces podremos ver de un vistazo y fácilmente toda la información relacionada si pulsamos a cualquier de ellas (Figura 2.7).

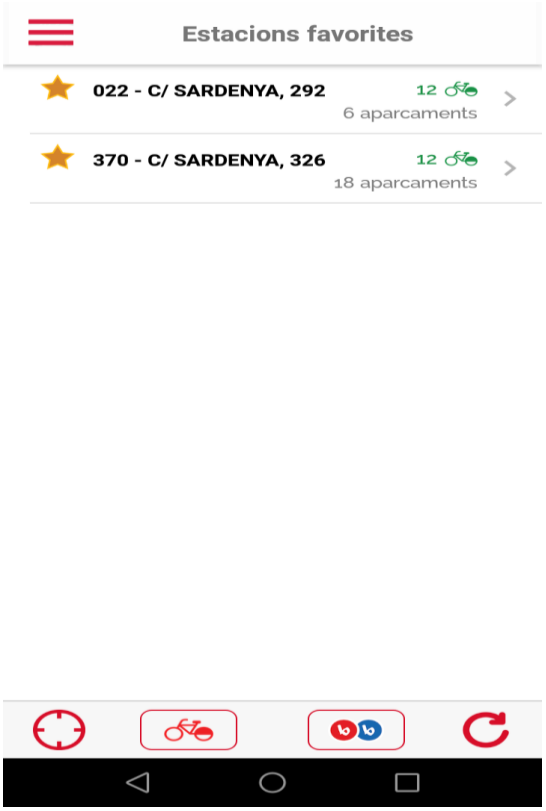

Figura 2.7: Estaciones favoritas

- Mapa / crear ruta:

  En esta pantalla podremos volver a ver el mapa de las estaciones y al lado derecho del icono del menú tenemos que pulsar al botón de “Ruta” para crear una ruta (Figura 2.8).

  Entonces tendremos que pulsar sobre “Nova ruta” para después seleccionar al mapa dos localizaciones diferentes para que se nos marque la ruta más rápida posible para llegar. Además, si pulsamos al icono del lado derecho podremos consultar los pasos exactos por tendremos que ir hasta el destino elegido.

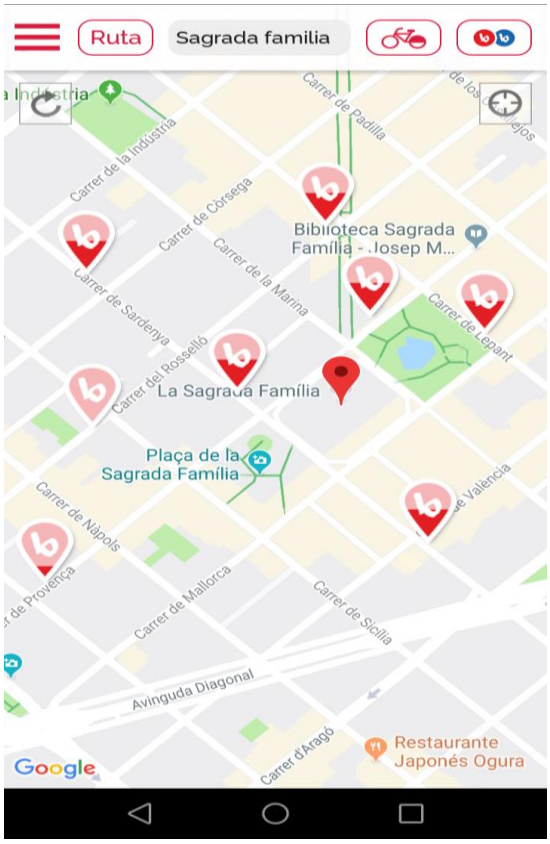

Figura 2.8: Mapa / crear ruta

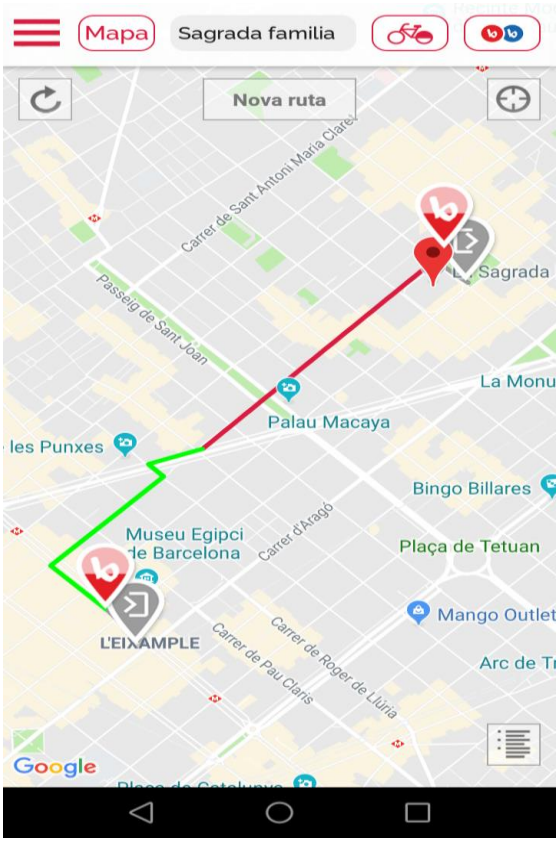

Figura 2.8: Mapa / crear ruta

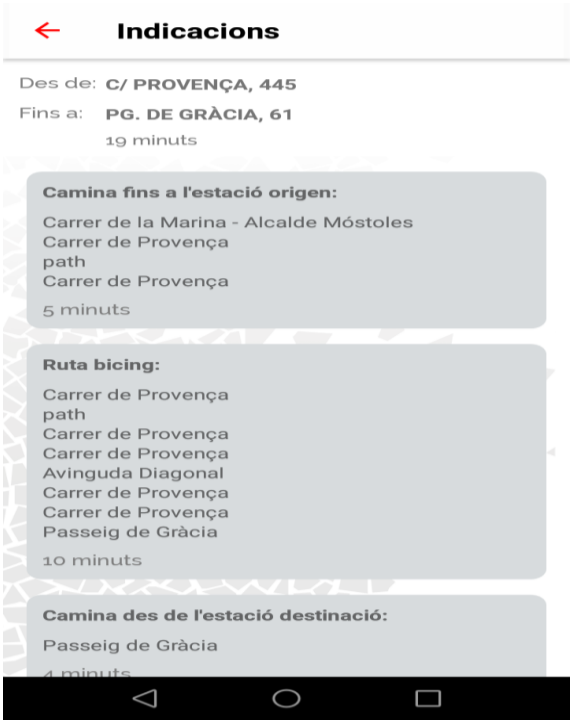

Figura 2.8: Mapa / crear ruta

- Estaciones:

  Si pulsamos al icono del menú “Estacions” se mostrará el listado de estaciones que hay en la ciudad ordenados por su identificador donde también podremos consultar la información correspondiente a cada una de estas (Figura 2.9).

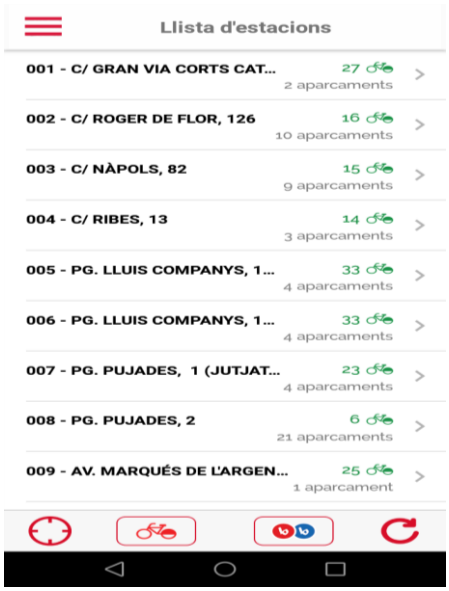

Figura 2.9: Estaciones

- Configuración:

  Finalmente tenemos la opción de poder configurar unos pequeños apartados (Figura 2.10) relacionados con el idioma, catalán, castellano o por ejemplo el tipo de ruta además de la máxima distancia andando para llegar al lugar final de la ruta.

Figura 2.10: Configuración

Cómo hemos podido observar su interfaz en muchos apartados es clara y directa sin muchas complicaciones tampoco de configuración haciendo de la aplicación muy accesible para cualquier persona que no utiliza a menudo este tipo de transporte, aunque a veces pueda mostrar información redundante a mi parecer. Se respeta una coherencia con los colores y no son nada agresivos a los ojos del usuario además de que respeta los cañones establecidos en cuanto a diseño de aplicaciones móvil donde podemos encontrar por ejemplo el menú lateral en el izquierdo. Ha sido mi principal referencia para hacer este proyecto.

2.2 Valencia Valenbisi

Valencia Valenbisi es una aplicación no oficial pero que permite ver el estado de las estaciones de bicicletas del servicio Valenbisi. Mediante el mapa, puedes encontrar las estaciones más próximas en la ubicación donde estás además del número de bicicletas y el número de plazas disponibles en la estación seleccionada. Esta, a diferencia de la aplicación de Bicing solo permite la consulta de información y no la reserva de bicicletas.

- Inicio:

A continuación de cargar la aplicación se nos mostrará el mapa de la ciudad (Figura 2.11) donde saldrán las estaciones de bicicletas repartidas por todo Valencia. Además, también tenemos otras dos pestañas: Favoritos y Tweets.

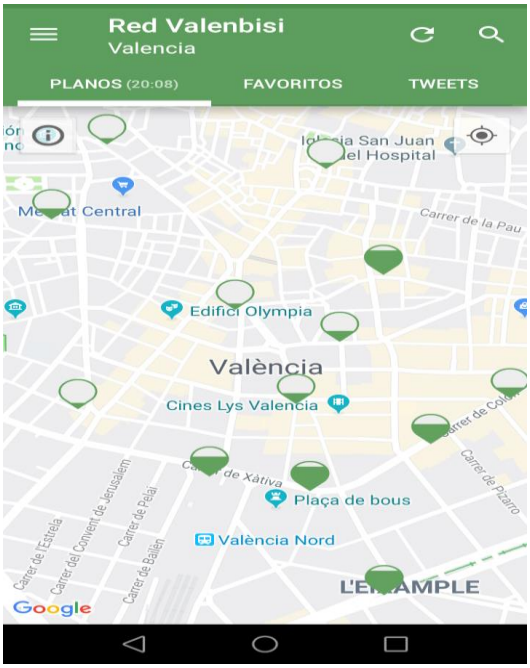

Figura 2.11: Pantalla principal y mapa

En pulsar sobre una estación cualquiera nos aparece la información detallada de la parada (Figura 2.12), donde podemos ver por ejemplo el número de plazas libres o el número de bicicletas disponibles.

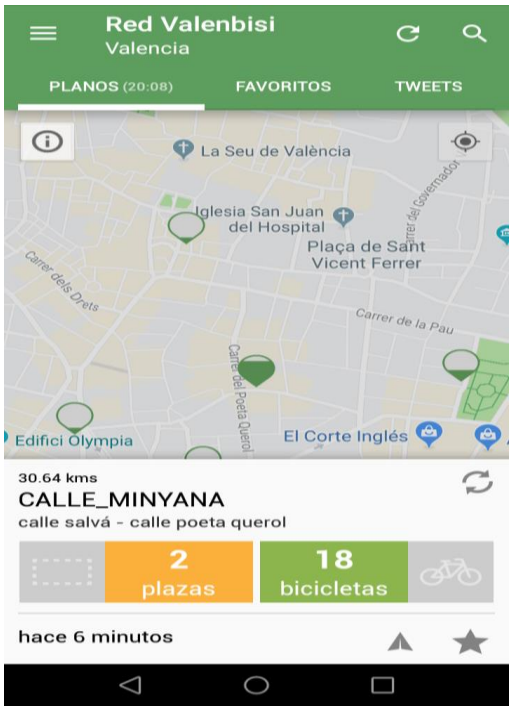

Figura 2.12: Planos

- Favoritos:

  En la pestaña de favoritos se mostrará las estaciones que previamente hemos dado al icono de favoritos (Figura 2.13).

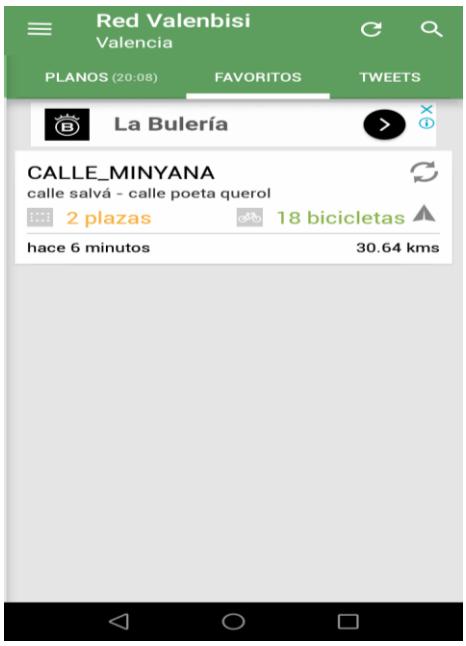

Figura 2.13: Favoritos

- Tweets:

En la pestaña de tweets podemos consultar la cuenta de twitter de Valenbisi donde saldrán las piadas que hacen los usuarios (Figura 2.14).

Figura 2.14: Tweets

- Menú:

Si pulsamos el icono de la parte izquierda superior de la aplicación se nos abrirá un menú lateral (Figura 2.15) donde podremos encontrar las diferentes funcionalidades en la aplicación.

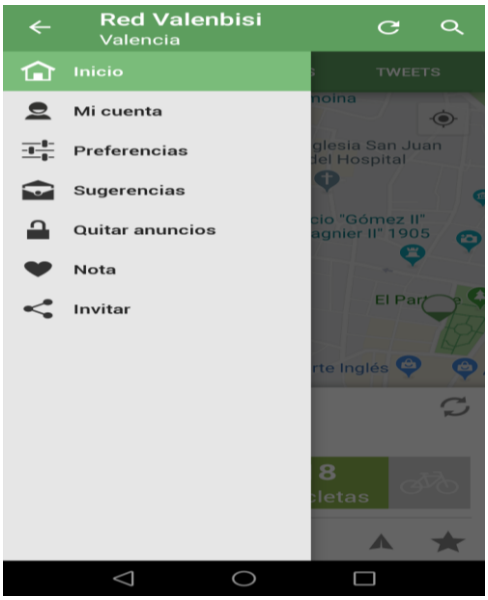

Figura 2.15: Menú

- Mi cuenta:

  En este apartado de la aplicación introduciremos nuestro correo electrónico y contraseña o iniciar sesión con tu cuenta de Google (Figura 2.16).

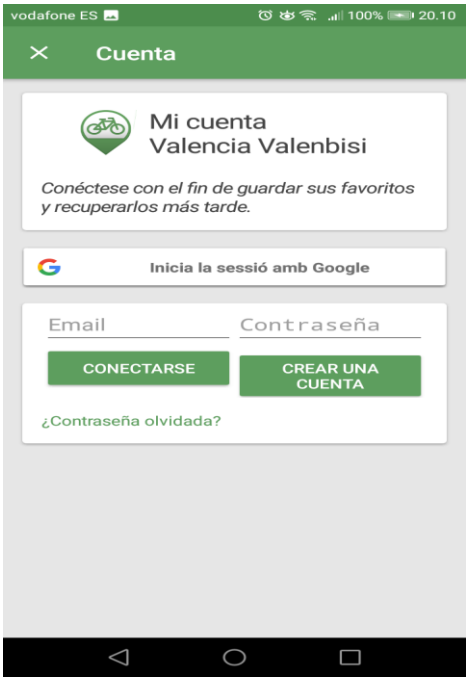

Figura 2.16: Login

- Preferencias:

  Aquí podremos configurar varias funcionalidades de la aplicación (Figura 2.17) como por ejemplo si queremos recibir notificaciones o si queremos que al iniciar se nos abra directamente nuestros favoritos.

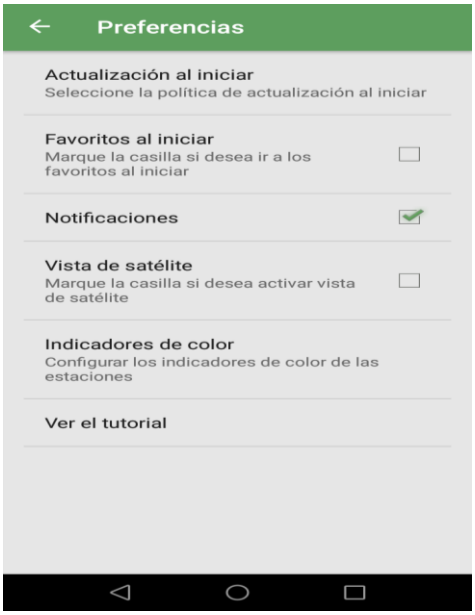

Figura 2.17: Preferencias

Esta aplicación nos da una buena información sobre las bicicletas disponibles y las estaciones repartidas en todas partes de la ciudad de Valencia para cualquier usuario del servicio ValenBisi, aun así, tiene un diseño muy simplificado y poco trabajado con la coletilla que nos aparece publicidad sobre todo al cargar por lo tanto al usuario le es un poquito angustiando poder utilizarla.

Capítulo 3
Análisis de requisitos

El propósito de esta parte de la memoria es escribir un documento en el cual tanto el cliente como el desarrollador entiendan y estén de acuerdo de los rasgos y características que tendrá el producto porque en un tiempo futuro no haya malentendidos o cuando menos, el mínimo número de rectificaciones posibles, puesto que un cambio mientras hay el desarrollo en marcha es mucho más costoso que haber planificado y acordado correctamente los requisitos y funcionalidades que tiene que ofrecer la aplicación.

La aplicación de Patival está de forma independiente a otros productos del mercado de reserva de vehículos, pero tiene mucha relación con las aplicaciones mencionadas al capítulo anterior puesto que no comparten temática pero si comparten una misma filosofía, diseño y funcionalidades. Estas aplicaciones tienen como objetivo principal ofrecer información sobre los vehículos disponibles a cada estación o incluso celular posibles rutas más cortas para realizar un trayecto entre estación y estación. La función de la aplicación es ser una versión accesible y portable, que permita acceder a la información por la calle de manera funcional.

Cómo se ha mencionado en apartados anteriores, la principal función de la aplicación es dar información sobre la cantidad de patinetes eléctricos disponibles, consultar su dirección, ver la cantidad de patinetes reservados, los datos personales y consultar la última reserva hecha por el usuario. Además, permitirá mediante la interfaz de la aplicación el cambio de los datos del usuario, reserva y anulación de patinetes.

Como tipo de usuario solo habrá uno, puesto que el objetivo de la aplicación es que el usuario, pueda consultar o modificar los datos que afectan a su cuenta o ver los datos que se ofrece en el mapa. Aun así este usuario tendrá que de identificarse o si no está creado registrarse a la aplicación.

3.1 Especificación de requisitos

Según Piattini Mario [1] "*un requisito se define como una condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado*".

Por lo tanto, el análisis de requisitos se puede definir como [1] "*el proceso del estudio de las necesidades de los usuarios para llegar a una definición de los requisitos del sistema, hardware o software, así como el proceso de estudio y refinamiento de estos requisitos*".

A continuación, realizaremos un estudio de los requisitos y analizar qué condiciones tendremos que satisfacer porqué nuestro proyecto esté bien realizado. Este apartado se fundamental que pueda ser entendido por ambas partes, el ingeniero de Software y el cliente, de forma que su lenguaje no será especialmente técnico. Además para cumplir con la normativa estándar este documento seguirá las indicaciones establecidas en el estándar IEEE 830 del institute of Electrical and Electronics Engineers o IEEE.

3.1.1 Requisitos específicos

Según el estándar IEEE, 1990 [2] la especificación de Requisitos del Software (ERS) se define como "*la documentación de los requisitos funcionales (funciones, rendimiento, diseño, restricciones y atributos) del software y de sus interfaces externas*".

Por lo tanto, seguidamente se definirán los requisitos funcionales y las interfaces externas de manera clara y con un lenguaje natural que permita a cualquier lector poder entenderlo fácilmente.

3.1.2 Interfaces Graficas

- Interfaz de usuario: La interfaz de usuario tiene que ser accesible e intuitiva para cualquier tipo de usuarios, des avanzados hasta noveles en el uso de estos tipos de aplicaciones, porque el rango de edades, profesiones o conocimientos previos es muy elevado y diferente. El diseño de la interfaz tiene que ser responsive, puesto que nos permitirá que se adapte su estilo a todo tipo de dispositivos y a cualquiera la plataforma, además, la navegación tiene que ser clara y fácil porque sepamos en cada momento donde nos encontramos. Últimamente los colores serán el moratón y el blanco puesto que son colores suaves, fáciles de reconocer por el usuario y nada agresivos a la vista del usuario. Más adelante se explicará este apartado con más profundidad.

- Interfaz de hardware: la aplicación responderá a las pulsaciones en la pantalla del dispositivo que haga el usuario.

3.1.3 Requisitos funcionales

Función	Consultar patinetes libres
Introducción	Ver la cantidad de patinetes libres a una estación
Entrada	Seleccionar cualquier estación que esté al mapa
Procesa	Acceso a la base de datos de las estaciones
Salida	Muestra un modal en el mapa donde se encuentra la información

Tabla 3.1: Requisito Funcional 01

Función	Consultar patinetes reservados
Introducción	Ver la cantidad de patinetes ya reservadas a una estación
Entrada	Seleccionar cualquier estación que esté en el mapa
Procesa	Acceso a la base de datos de las estaciones
Salida	Muestra un modal en el mapa donde se encuentra la información

Tabla 3.2: Requisito Funcional 02

Función	Consultar información de la estación
Introducción	Ver la información asociada una estación
Entrada	Seleccionar cualquier estación que esté en el mapa
Procesa	Acceso a la base de datos de las estaciones
Salida	Muestra un modal en el mapa donde se encuentra la información

Tabla 3.3: Requisito Funcional 03

Función	Reservar bicicleta libre
Introducción	Reservar un patinete libre en una estación
Entrada	Seleccionar cualquier estación que esté en el mapa
Procesa	Acceso a la base de datos de las estaciones
Salida	Muestra un modal en el mapa donde poder hacer la reserva

Tabla 3.4: Requisito Funcional 04

Función	Consultar perfil de usuario
Introducción	Visualizar los datos personales del perfil de usuario
Entrada	Seleccionar en el menú lateral izquierdo la opción perfil
Procesa	Acceso a la base de datos de los usuarios
Salida	Muestra la pantalla con los datos personales del usuario

Tabla 3.5: Requisito Funcional 05

Función	Editar perfil de usuario
Introducción	Modificar los datos personales del perfil de usuario
Entrada	Seleccionar en el menú lateral izquierdo la opción perfil y después pulsar el botón editar perfil
Procesa	Acceso a la base de datos de los usuarios
Salida	Muestra la pantalla para modificar los datos personales del usuario

Tabla 3.6: Requisito Funcional 06

Función	Consultar las reservas de patinetes
introducción	Visualizar la reserva hecha por el usuario
Entrada	Seleccionar en el menú lateral izquierdo la opción reservas
Procesa	Acceso a la base de datos de los usuarios
Salida	Muestra la pantalla con la reserva hecha por el usuario

Tabla 3.7: Requisito Funcional 07

Función	Anular las reservas de patinetes
Introducción	Visualizar la reserva hecha por el usuario
Entrada	Seleccionar en el menú lateral izquierdo la opción reservas y pulsar en Anular reserva
Procesa	Acceso a la base de datos de los usuarios
Salida	Muestra la pantalla con la reserva hecha por el usuario

Tabla 3.8: Requisito Funcional 08

Función	Acreditarse
Introducción	Verificar el usuario a la base de datos
Entrada	Al cargar la aplicación tiene que introducir email y contraseña
Procesa	Acceso a la base de datos de los usuarios
Salida	Muestra la pantalla para hacer el login del usuario

Tabla 3.9: Requisito Funcional 09

Función	Registrarse
Introducción	Dar de alta el usuario creado a la base de datos
Entrada	En la pantalla de registro tiene que introducir sus datos
Procesa	Acceso a la base de datos de los usuarios
Salida	Muestra la pantalla de registro

Tabla 3.10: Requisito Funcional 10

3.2 Casos de uso

Un caso de uso es una secuencia de acciones realizadas por el sistema, que producen un resultado observable y provechoso para un usuario, es decir, representa el comportamiento del sistema para dar respuestas a los usuarios. Proporcionan, por lo tanto, una manera clara y precisa de comunicación entre cliente y desarrollador. Desde el punto de vista del cliente proporcionan una visión “general” del sistema, es decir, como será el sistema desde el exterior sin necesidad de entrar en detalles de su construcción. Para los desarrolladores, suponen el punto de partida y el eje sobre el cual se apoya todo el desarrollo del sistema en sus procesos de análisis y diseño.

Son un elemento importantísimo para un proyecto de ingeniería de software. En este caso solo hay un actor, o entidad que realiza una acción. Cada caso de uso contará de 5 apartados describiendo así sus objetivos, su función y últimamente las diferentes acciones a realizar para completarlo.

A continuación, en la figura, se mostrarán todos los diversos casos de uso posibles del usuario y su explicación más detallada.

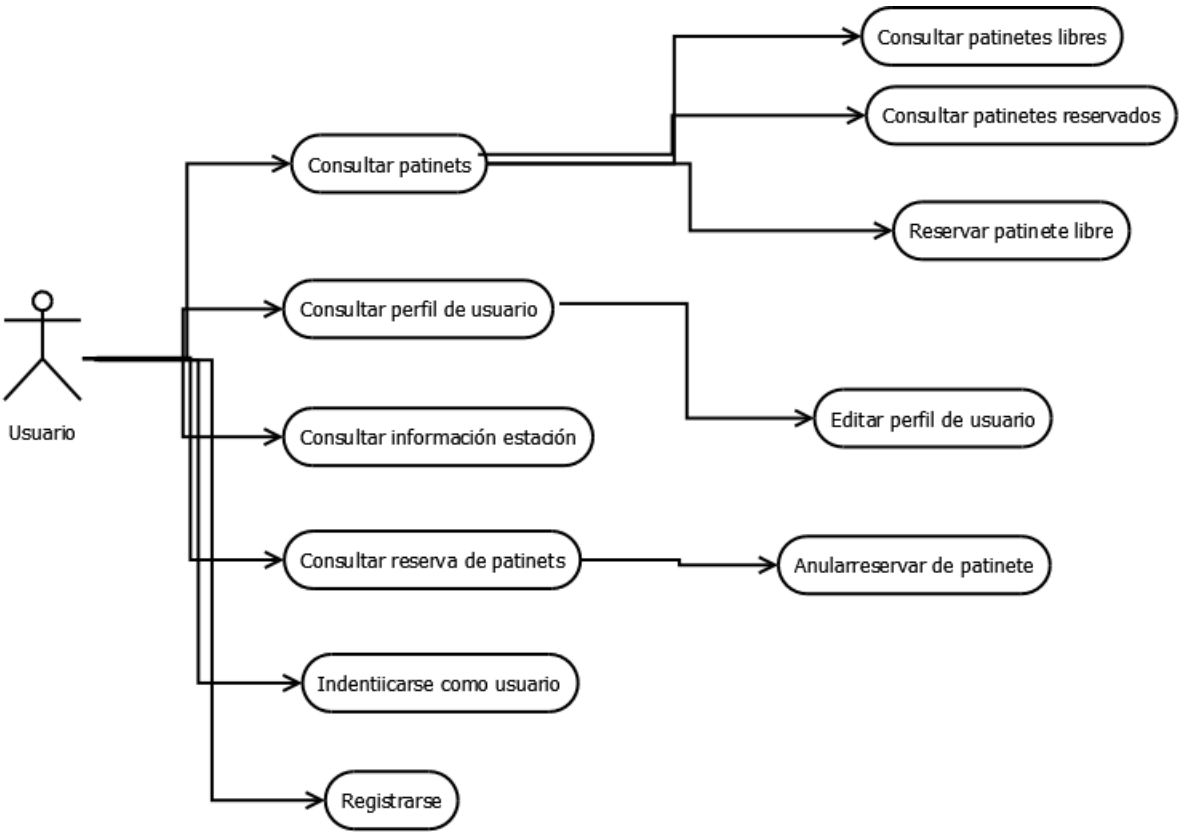

Figura 3.2: Diagrama de uso del usuario

1. Código: CU 01
   Ver el perfil de usuario
   Objetivos: Consultar el perfil
   Descripción: Permite visualizar los datos personales del usuario
   Secuencia: Se selecciona al menú lateral izquierdo el botón de Perfil

2. Código: CU 02
   Editar el perfil de usuario
   Objetivos: Modificar los datos de perfil del usuario
   Descripción: Permite cambiar los datos de perfil del usuario
   Secuencia: Se selecciona a la barra de navegación el botón de Perfil y a continuación pulsar el botón editar perfil

3. Código: CU 03
   Ver los patinetes libres disponibles
   Objetivos: Consultar las bicicletas libres disponibles a la estación
   Descripción: Permite consultar los patinetes libres disponibles a la estación seleccionada
   Secuencia: Se selecciona al mapa cualquier estación que sale marcada

4. Código: CU 04
   Ver los patinetes reservados
   Objetivos: Consultar los patinetes reservados a la estación
   Descripción: Permite consultar los patinetes reservados a la estación seleccionada
   Secuencia: Se selecciona al mapa cualquier estación que sale marcada

5. Código: CU 05
   Ver información sobre la estación
   Objetivos: Consultar información sobre la estación
   Descripción: Permite consultar información sobre la estación seleccionada
   Secuencia: Se selecciona al mapa cualquier estación que sale marcada

6. Código: CU 06
   Reservar un patinete libre
   Objetivos: Reservar un patinete libre de una estación
   Descripción: Permite reservar un patinete libre de la estación seleccionada
   Secuencia: Se selecciona al mapa cualquier estación que sale marcada y hacer clic al botón de reservar patinete

7. Código: CU 07
   Anular un patinete reservado
   Objetivos: Consultar información sobre la estación
   Descripción: Permite consultar información sobre la estación seleccionada
   Secuencia: Se selecciona al menú lateral izquierdo el botón de Reservas y pulsar el botón anular reserva

8. Código: CU 08
   Identificarse como un usuario registrado
   Objetivos: Identificarse como un usuario registrado al sistema de bases de datos
   Descripción: Permite Identificarse mediante email y contraseña
   Secuencia: Se selecciona al menú lateral izquierdo el botón de Reservas y pulsar el botón anular reserva

9. Código: CU 09
   Registrarse como usuario
   Objetivos: Consultar información sobre la estación
   Descripción: Permite consultar información sobre la estación seleccionada
   Secuencia: Se selecciona al menú lateral izquierdo el botón de Reservas y pulsar el botón anular reserva

3.3 Mockups

Los Mockups son fotomontajes que permiten a los diseñadores gráficos y programadores mostrar al cliente como quedarían sus diseños sobre los cuales se va a basar el diseño final de la aplicación. Dicho de otro modo, son la idea conceptual del diseño de la aplicación. [3] "*Permiten visualizar el flujo de la información ayudando a concretar muchos aspectos entre el cliente y la empresa que se encarga de desarrollar la aplicación. A la vez que se evita que 'queden flecos sueltos' en cualquier aspecto del desarrollo*". Cada Mockup tiene relacionado uno o varios casos de uso, puesto que a través de las pantallas se puede representar una o varias acciones contenidas en estos, además del diseño de la navegación entre las diferentes páginas de la aplicación.

3.3.1 inicio / login

Esta pantalla es la primera en aparecer, donde contendrá los campos de identificación del usuario, a banda si el usuario no está todavía registrado al sistema habrá un botón donde poder ir a la pantalla del registro donde podrá escribir sus datos, además saldrá logotipo principal de la aplicación.

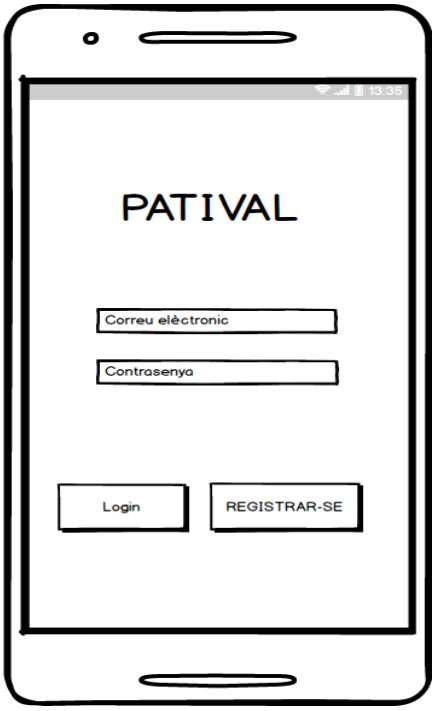

Figura 3.2: Mockup login, caso de uso CU8

3.3.2 Registro

En esta pantalla el usuario podrá introducir un nombre de usuario, una contraseña además de repetirla para confirmar la creación del usuario y últimamente su correo electrónico para identificarlo dentro del sistema.

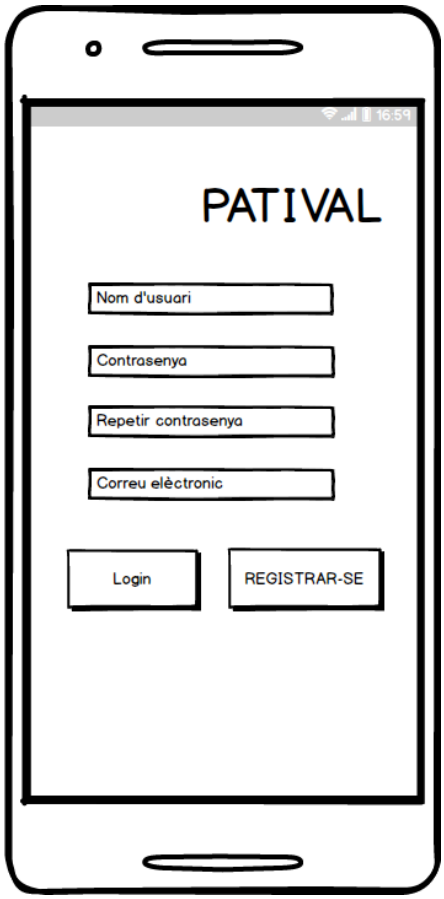

Figura 3.3: Mockup registro, caso de uso CU9

3.3.3 Home

Aquí veremos el mapa de Google maps donde el usuario podrá elegir y seleccionar cualquier estación de patinetes a la ciudad. Una vez seleccionado se mostrará un modal donde se podrá consultar la información sobre los patinetes que hay a la estación y a la vez un botón donde se podrá reservar un patinete. Esta es la pantalla principal de la aplicación.

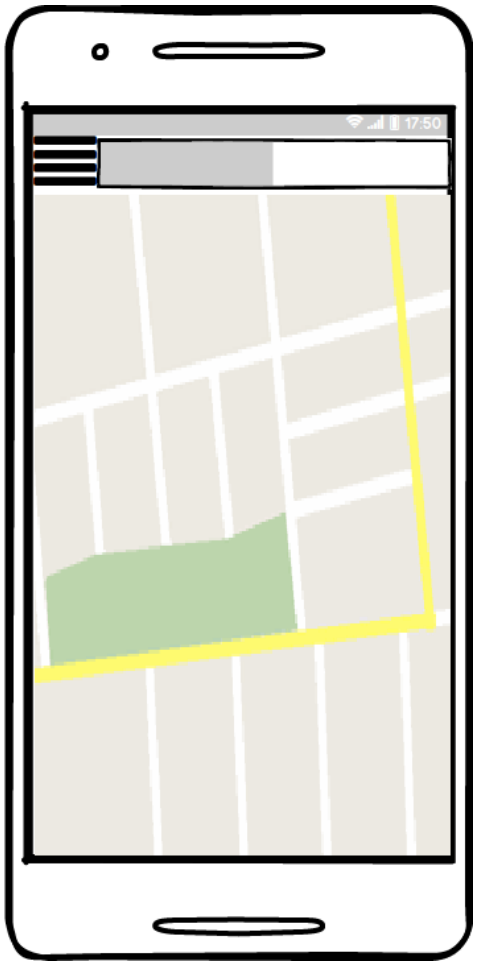

Figura 3.4: Mockup home, caso de uso CU3, CU4, CU5 i CU6

3.3.4 Perfil de usuario

En esta vista se mostrará los datos personales del perfil del usuario. Además tendrá la opción de editar los datos de su perfil a cualquier momento.

Figura 3.5: Mockup perfil de usuario, caso de uso CU1, CU2

3.3.5 Reservas

En esta pantalla podremos encontrar la reserva de un patinete hecha por el usuario anteriormente.

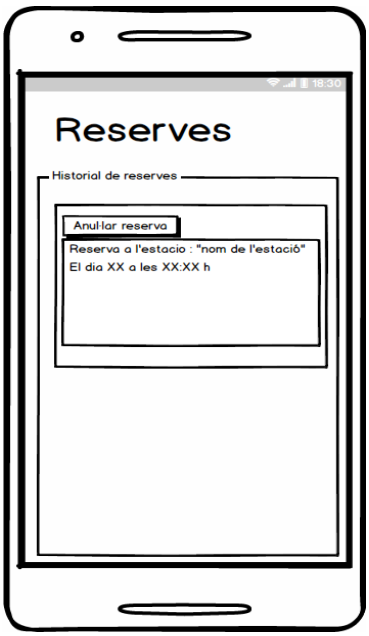

Figura 3.6: Mockup reservas, caso de uso CU4

3.3.6 Editar perfil de usuario

En la vista de edición de perfil, el usuario podrá modificar su nombre de usuario, correo electrónico, dirección donde vive, su ciudad y su código postal. En querer editar el usuario pulsa al botón editar.

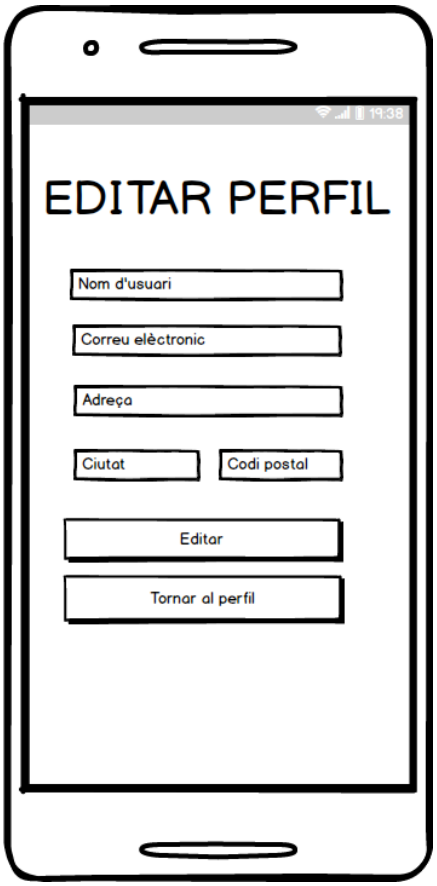

Figura 3.7: Mockup editar perfil, caso de uso CU2

Capítulo 4
Diseño

En este capítulo se hará una explicación de las tecnologías empleadas para desarrollo y diseño del proyecto además de su estructura de aplicación y últimamente una descripción de la base de datos en tiempo real de Firebase creada exclusivamente para la persistencia de datos que utiliza y muestra la aplicación junto con el tipo de seguridad que usa para la validación de los usuarios.

De los últimos diez años hacia acá la concepción que los desarrolladores de software y los usuarios finales, se a decir, los potenciales clientes de un software han cambiado muchísimo. No sirve ya la vieja idea del usuario sentado a una silla con su ordenador personal, sino que la aparición del smartphone o también la posibilidad de trabajar desde casa ha hecho que las aplicaciones y servicios web hayan experimentado una subida fortísima respecto de las aplicaciones de escritorio tradicionales. Este crecimiento ha permitido el desarrollo de multitud de aperos para su creación, o como la tecnología que se usara en este proyecto, para usar sus aperos para crear proyectos derivados. Las aplicaciones web son software donde su principal rasgo el acceso a ellas por Internet o bien por una Intranet utilizando un navegador web, esto hace que tengan la ventaja de no hacer falta ninguna instalación. Estas se componen fundamentalmente en dos partes, la parte frente-end y la parte back-end. El frente-end es el software que con el cual interacciona el usuario, mientras que las tareas de conexión con el servidor se denomina back-end, Esta ejecuta la lógica de la aplicación y la conexión con la base de datos, donde se encuentra persistencia de la aplicación.

El desarrollo con éxito de una aplicación depende en gran medida del uso de la tecnología apropiada. Hay muchos factores a considerar, desde el tipo de tecnología a utilizar y el coste hasta la selección de los desarrolladores adecuados para el proyecto. Para hacer una buena elige hace falta:

- *“Selecciona una tecnología frente-end fácil de utilizar: Considera a los usuarios antes de que en la tecnología. Se tiene que tener en cuenta la capacidad de respuesta, la interfaz de usuario (IU) y la experiencia del usuario (IU) cuando la aplicación se ejecuta en tablets, equipos de escritorio o smartphones. La tecnología también tendría que soportar los mejores lenguajes de programación para frontend y backend. Además, considera el uso de navegadores populares que funcionan en múltiples plataformas (Windows, iOS, Linux, Android).*

- *Usar librerías o frameworks: acortan el tiempo de desarrollo y conducen al desarrollo de una tecnología fácil de usar.*

- *Elegir los aperos que aumentan la velocidad de iteración: La velocidad de iteración se refiere a la velocidad con la cual se completan las fases o ciclos de desarrollo dentro de todo el proyecto de desarrollo. Cuanto antes mejor se completan estas iteraciones, antes se podrá lanzar la aplicación al mercado. También se puede reducir el tiempo de desarrollo asegurándose que tu equipo*

esté familiarizado con los lenguajes de programación y la pila de tecnología seleccionados" [4].

Un balance equilibrado entre la tecnología más apropiada para el tipo de proyecto deseado y una que te permita desarrollarlo lo más rápido posible y tener una versión estable de la aplicación en el tiempo premeditado es la clave para optimizar tiempo y adecuarse al producto pedido y diseñado.

4.1 Tecnologías utilizadas

En este apartado se hará una descripción de las tecnologías que se han usado para desarrollar este proyecto.

4.1.1 Ionic Framework

Ionic es un framework para el desarrollo de aplicaciones híbridas, sobre todo para móviles, aunque también puede utilizarse para desarrollar aplicaciones web convencionales. Este es la tecnología fundamental de esta aplicación, sin duda el coro del proyecto. Ofrece al desarrollador de software toda una base de librerías y estructuras para construir de forma eficiente y con poco coste una aplicación móvil híbrida utilizando tecnologías web que explicaremos más adelante.

Figura 4.1: logotipo Ionic

Su core emplea el framework javascript Angular, este por supuesto, va actualizándose a la vez que se actualizan las versiones de ionic añadiendo o cambiando la forma de trabajo cuando se desarrolla una aplicación con él. El cambio de versión más relevante fue de Ionic 1 a Ionic 2, donde se cambió AngularJs a Angular 2, por lo tanto dejaba de utilizar Javascript y empezó a usar Typescript junto a HTML 5 y SaSS, una evolución de CSS que mujer más opciones para el diseño, como la creación de variables CSS globales. Para hacer la compilación de todas estas tecnologías web a código nativo Android, iOS o Windows Phone, utiliza Apache Cordova, otra tecnología fundamental de Ionic conjuntamente con todas las antes mencionadas.

Para el desarrollo de una aplicación con Ionic este cuenta con su CLI también denominado interfaz de líneas de comandos, que permite probar la aplicación en local en un navegador web, compilar el código escrito o generar automáticamente los diferentes elementos que está compuesto una aplicación hecha con Ionic, como una página, un servicio o un componente. Este último es un rasgo importante puesto que cualquier aplicación desarrollada correctamente con ionic [6] "*está compuesto por un árbol de componentes, esto hace que permita que sean más fácilmente escalables y sostenibles a la vez de hacer cambios al código"*. Ionic 3, es la versión utilizada para desarrollar este proyecto.

4.1.2 TypeScript

TypeScript [7] "*es un lenguaje de programación de código abierto con aperos de programación orientada a objetos desarrollado por Microsoft. Convierte su código en Javascript común. Por açó, es denominado también como Superset de Javascript, lo cual significa que si el navegador está basado en Javascript, este nunca llegará a saber que el código original fue realizado con TypeScript y ejecutará el Javascript como lenguaje original*". Es altamente escalable, además de ser un lenguaje tipat, la cuando cosa nos permite crear estructuras de datos, aunque el uso de los tipos es opcional, su uso mejora la verificación estática del código.

Figura 4.2: logotipo TspeScript

4.1.3 Angular

Angular es el framework de Javascript por excelencia más empleado en el desarrollo web. A menudo se confunde Angular con AngularJS puesto que la versión actual es conocida como Angular frente a la primera versión de este framework que es conocida como AngularJS, habiendo grandísimas diferencias entre ellos, hasta el punto que son considerados frameworks totalmente diferentes. Angular usa TypeScript para la programación, al contrario de AngularJS que utiliza Javascript. Esto convierte Angular en un framework tipat y con clases, con todas las posibilidades que esto permite.

Figura 4.3: logotipo Angular

Ionic 3 antes mencionado usa la versión Angular 6. Las principales ventajas de Angular son que permite generación de código muy eficiente, la gran modularidad que proporciona la utilización de componentes y la capacidad de desarrollo reactivo.

Angular está compuesto por estas características fundamentales:

- NgModules: cualquier aplicación Angular está compuesta por los decoradores NgModules, que proporcionan un contexto de compilación a los componentes y agrupa código en módulos funcionales. NgModules, sirve como módulo raíz de inicio del proyecto. Los módulos pueden importar otros módulos y viceversa. Con esto podemos organizar el código en bloques funcionales que nos permite conseguir un nivel alto de reutilización y facilitar el mantenimiento para las aplicaciones más complejas.

- Template: los Template están formados por código HTML y notaciones Angular que nos permite cambiar el código HTML según la lógica de la aplicación. Las directivas de un Template abastecen la lógica para conectar nuestra aplicación con el DOM, se a decir, con el modelo en objetos para la representación de documentos. Antes de montar la vista, Angular evalúa las notaciones escritas en el Template y a continuación cambia los elementos del DOM. Estos elementos se pueden modificar en ambas direcciones, desde el usuario o desde nuestro código.

- Components: cada componente define una clase, en la cual está alojada la información y la lógica, es asociado con un documento HTML que le definirá la vista de este. Existe al menos un componente raíz por proyecto, que conecta la jerarquía de componentes con el DOM.

- Data binding: parte fundamental de angular que le permite tener reactividad, se a decir, enlazar datos bidireccionalmente, conectando el template con las partes de un componente. Para conseguirlo, se utiliza las notaciones en el documento HTML que permiten a Angular la comunicación bidireccional.

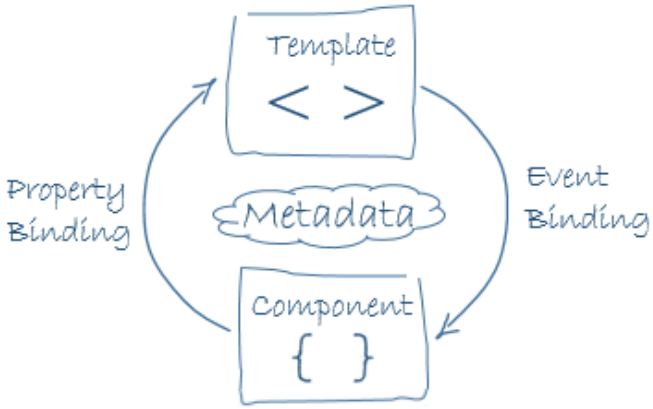

Figura 4.4: Realización del Data-Binding

- Services: Los servicios son los elemento empleados fundamentalmente para compartir información o lógica entre los componentes de nuestra aplicación.

4.1.4 Firebase

La base de datos de firebase almacena y sincroniza los datos con nuestra base de datos, todo esto se aloja en la nube (internet). [5] "*Estos datos que están en la nube son almacenadas en formato JSON (Javascript Object Notation) y se pueden agregar reglas para permitir peticiones con token o solo desde una URL y los datos de la base se sincronizan con todos los clientes en tiempo real esto ayuda mucho cuando la app no tiene conexión a Internet. Firebase responde aunque no tenga internet esto es gracias al SDK (kit de desarrollo de software) de firebase hace que nuestros datos persistan en el disco, cuando la conexión vuelve el dispositivo lo reconoce y el guarda en el servidor*".

Figura 4.5: logotipo Firebase

4.1.5 Git

Git es un sistema de control de versiones de en balde y de código abierto que nos permite almacenar en un repositorio remoto las diferentes versiones de nuestro proyecto. Para hacerlo git tiene un bash propio con sus mandos que nos permite subir nuestra versión local y compararla con la versión remota y si no tiene conflictos entre ambas versiones entonces se añade en el repositorio a git como una versión nueva del proyecto. Mediante las versiones subidas nos permite volver atrás y recuperar una versión antigua si hubiera que deshacer un paso errado. Además nos permite crear las llamadas "ramas" que permiten trabajar en paralelo con otros desarrolladores o trabajar en diferentes características al mismo tiempo del mismo proyecto.

Figura 4.6: logotipo Git

4.1.6 Framework de desarrollo

Para desarrollar y realizar todo este proyecto se ha usado el entorno de desarrollo Visual Studio Code. Lanzado en 2015, Visual Studio Code va un paso más allá de Visual Studio, presentándose como uno de los framworsks más potentes del momento y más utilizada por desarrolladores de software. Nos permite fácilmente encontrar el comienzo y el fin de nuestros bloques de código. Si trabajamos con tecnologías Web veremos que está integrado Emmet (un conjunto de atajos de código) sin necesidad de agregar ningún plugin. También encontraremos de manera integrada el acceso a GIT, el mapa de navegación de nuestro código y las opciones de agregar extensiones. Precisamente buena parte del potencial de Visual Studio Code es el uso de las extensiones. Tiene un buscador de extensiones que nos ofrecerá en los mismos resultados la posibilidad de instalarlas sin necesidad de salir del programa.

Figura 4.7: logotipo Visual Studio code

4.2 Estructura de la aplicación

La arquitectura de software es [7] "*la definición y estructuración de una solución que cumple con los requisitos técnicos y operativos. La arquitectura de software optimiza los atributos que implican una serie de decisiones, como la seguridad, el rendimiento y la capacidad de gestión. En última instancia, estas decisiones afectan la calidad, el mantenimiento, el rendimiento y el éxito general de las aplicaciones*".

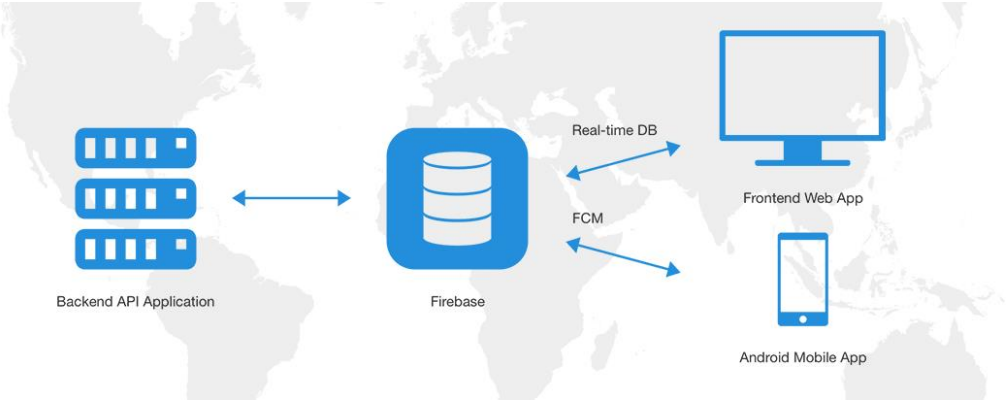

Figura 4.8: Estructura de una aplicación Firebase [8]

Este proyecto divide la arquitectura de la aplicación en dos capas principales: FrontEnd y Base de datos. La razón de esta arquitectura es porque el servicio de base de datos en tiempo real de Firebase nos permite ahorrarnos la programación de nuestro propio Backend. El controlador pasa a la parte cliente, así esta capa tendrá una estructura Modelo Vista Controlador (MVC), contendrá toda la lógica de la presentación y se comunicará con Firebase intermediando gritadas a su propia APIO, Application Programming Interface, con sus operaciones CRUD, Create, Read, Update and Delete, correspondientes. Esta estructura MVC del cliente separa los datos y la lógica de la presentación de estas, entonces encontramos al frontend una semiestructura que está formada de tres partes: el Frontend Services Layer, que es la encargada de la comunicación con el servidor además de gestionar los datos. La Controller Layer que se encarga de mantener unidos la vista y el modelo además define como la vista tiene que reaccionar a los cambios que pueda haber al modelo. Finalmente tenemos la View Layer que representa los diferentes componentes de la interfaz de usuario y de qué forma se muestran los datos.

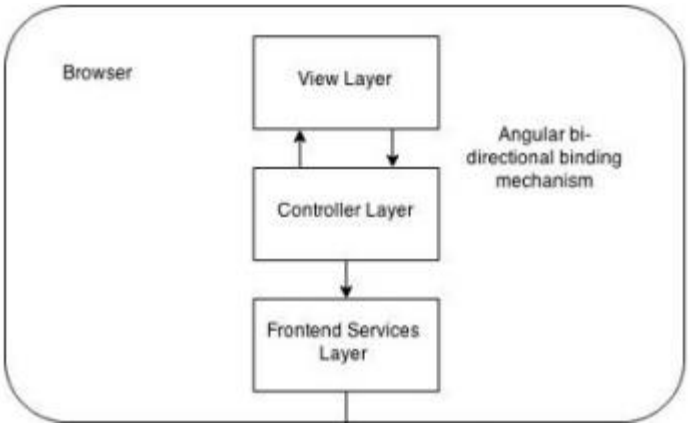

Figura 4.9: Parte del cliente

A continuación, se explicará las capas de la arquitectura de este proyecto además de cómo se organizan sus diferentes componentes.

4.2.1 Frontend

El Frontend, [10] "*una posible traducción sería fachada final. Es la especialidad para el desarrollo software, que trabaja la interfaz de usuario y hace que el usuario pueda interactuar con la aplicación. Está orientado a lenguaje de marcas y al lenguaje de programación web de ejecución en los equipos clientes como HTML, CSS y Javascript, sin necesidad de uso de servidores externos*". la estructuración de los apartados, tamaños, márgenes entre estructuras, tipos de letra, colores, adaptación para diferentes pantallas, los efectos de ratón, teclado, movimientos, desplazamientos, efectos visuales… Esto sería la baso en la cual se centra el frontend, dar formato a contenidos, desarrollo del aspecto de la web y manipular resultados de datos obtenidos.

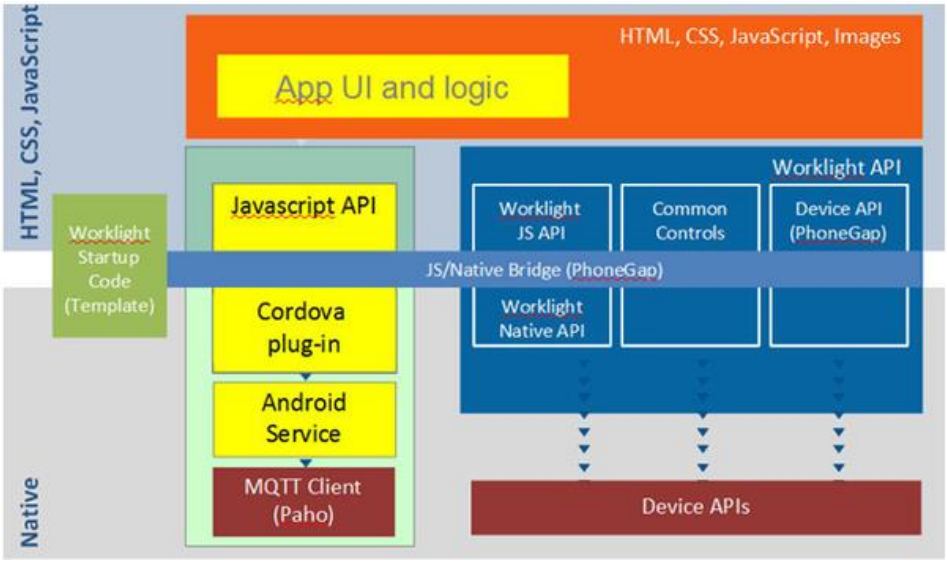

Figura 4.10: Estructura de una app hibrida [9]

En este proyecto cambia un poquito ya que aunque se utiliza lenguajes de programación web, sobre todo frameworks potentes como Angular que ayudan a su desarrollo, nuestro producto final estará en código de Android o IOs nativo, porque intermediando Ionic Cordova nos traducirá todo el código escrito en código nativo de estos. Aun así, a pesar de que se haga esta traducción la estructura que se describirá a continuación es la creada por Ionic que se asemeja a la empleada por Angular.

- Page: una página o Page como se denomina en Ionic, tendrá asociada una vista, que está formado por cuatro ficheros: .htlm, module.ts, .scss y .ts. El fichero html contiene todo el cuerpo de la presentación de la página donde su fuente de estilos propia es el fichero scss. La lógica de la vista está al fichero .ts que se encargará de proporcionar al html mediante el 'fecha-binding' los datos que necesita y comunicarse con otros componentes además una página puede importar diferentes componentes y/o servicios que es declararon en el fichero module.ts para poder emplearlos.

  Pondremos como ejemplo la página home (como podemos ver en la imagen) donde hay un fichero html llamado home.html que utiliza dos componentes "maps-layou-hoja" y "alert-info", que mostrarán la información utilizando los datos proporcionados por el fichero home.ts intermediando el fecha binding. Estos datos se solicitarán al iniciarse la vista, a un servicio que se encargará de obtenerlos.

Una vez obtenidas todos los datos, el archivo home.html usará el fichero home.scss para aplicar sus estilos personalizados aunque los estilos que comparte con el resto de la aplicación se encuentran al archivo main.scss contenido en la carpeta theme. (Veáis el apéndice - Elementos del frontend, sección Page).

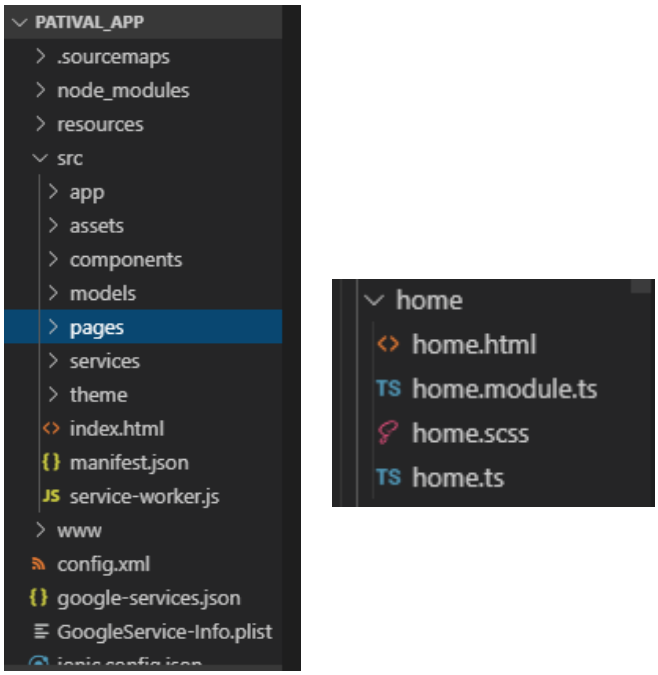

Figura 4.11: Páginas

- Services: els serveis són classes que s'empren per a connectar l'aplicació amb la base de dades de Firebase, i seran importats per les distintes pàgines per a accedir a les dades emmagatzemades en ella. Aquests serveis utilitzen els mòduls AngularFireDatabase, AngularFireAuth AngularFirestore i en el cas de les dades del mapa utilitza el mòdul GeoFire per a realitzar les diferents peticions a la base de dades que posteriorment processarà la page que els sol·licite. (Vegeu l'apèndix - Elements del Frontend, secció Services)

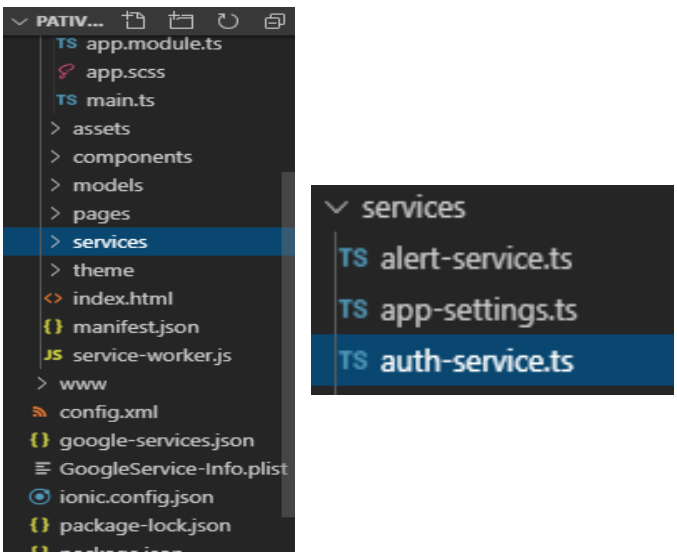

Figura 4.12: Serveis

- Components: los componentes son una parte fundamental básica de construcción en una aplicación desarrollada con Angular puesto que permite dividir en trozos diferentes funcionalidades de la aplicación para después ser usados como si fueran hashtags HTML dentro del .html de las paginas antes mencionadas. Se definen con el decorador @componiendo donde podremos definir sus propiedades entre ellas su nombre para ser utilizado después dentro de una vista.

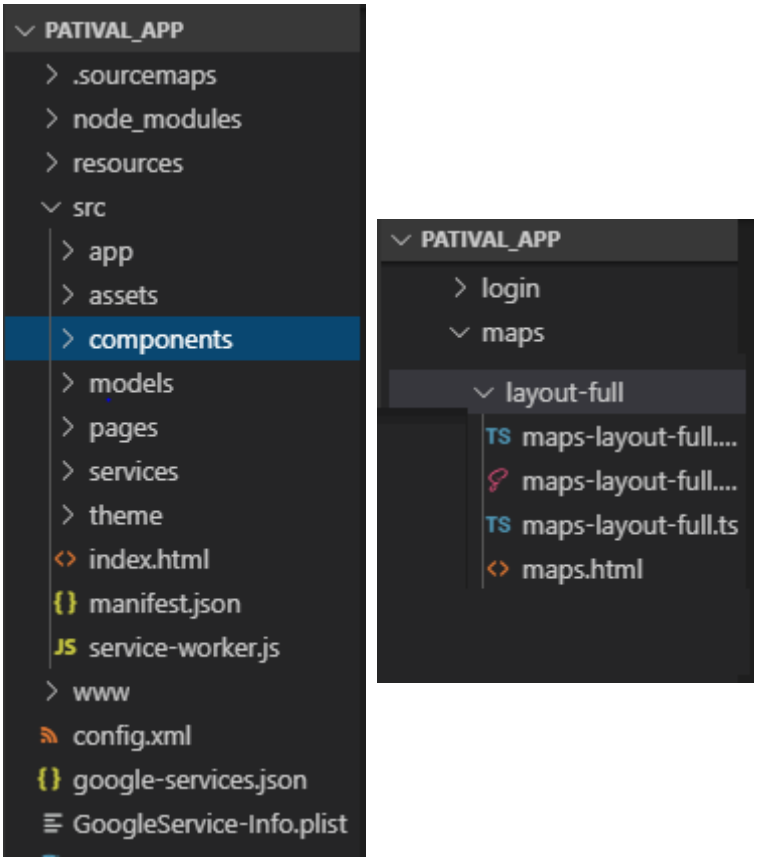

Figura 4.13: Componentes

Otros archivos importantes

- app.module.ts: Este es un archivo fundamental, se declaran todos los componentes y servicios utilizados en toda la aplicación además los componentes de entrada. Además cuenta con todas las importaciones.

- package.json: se almacenan todas las dependencias de la aplicación indicando además las versiones utilizadas de cada una.

- main.scss: en este archivo scss están almacenadas todas las variables globales de css, los colores principales, secundarios, tamaños y cualquier otro estilo compartido por todas las vistas.

- user.class.ts: aquí está definida la clase User con todos sus atributos que corresponden a los que serán insertados dentro de la base de datos de Firebase además dispone de su constructor para instanciar un objeto User.

4.2.2 Estructura de la base de datos

La base de datos en tiempo real alojada en Firebase está formada por dos objetos JSON, puesto que esta trabaja como un array de objetos JSON no como una tabla relacional convencional. Por lo tanto, para leer, insertar o modificar cualquier dato de la base de datos se utilizan las funciones de la API de Firebase que automáticamente transforma un objeto en otro en formato JSON.

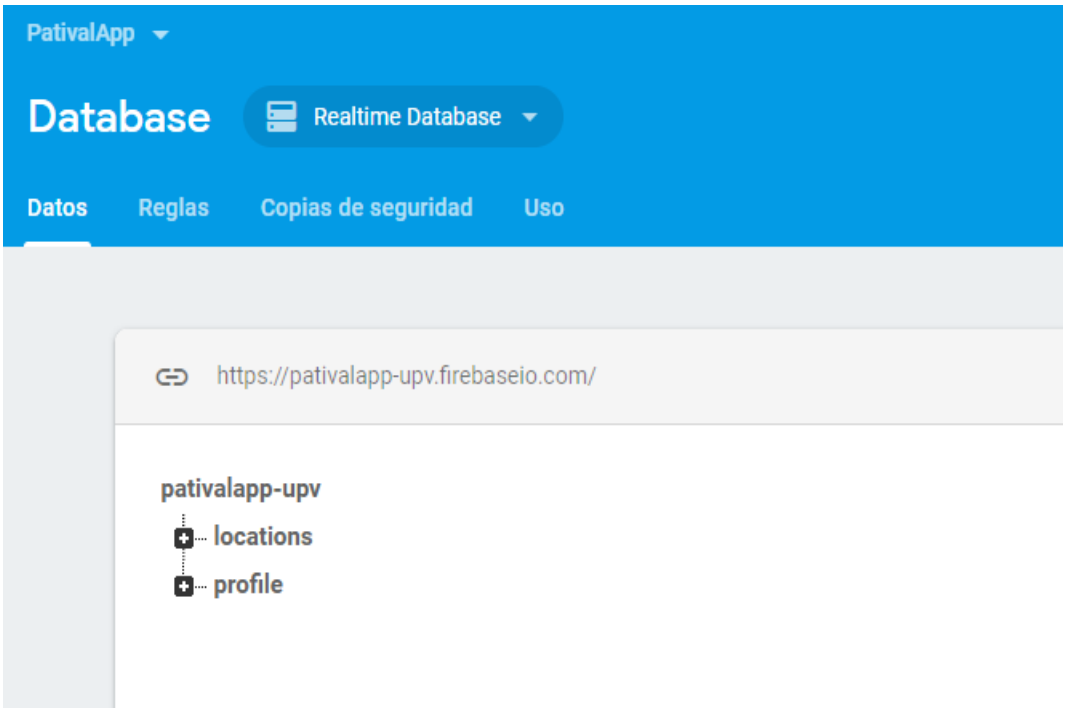

Figura 4.14: Base de datos en Firebase

4.2.2.1 Objecto Locations

En este objeto se almacena toda la información relacionada con las estaciones de patinetes repartidas por la ciudad. Está formada por un campo identificador con el formato: location-X donde X sería un número empezando por el cero.

Cada objeto location-X tiene definidas las siguientes propiedades:

- address: esta propiedad contiene la dirección completa donde está la estación.
- g: esta propiedad es una campo identificador que la librería Geofire crea automáticamente para indexar los puntos de coordenadas de cada estación.
- i_free: se almacena los patinetes libres disponibles a la estación correspondiente.
- i_reserved: se almacena los patinetes reservados a la estación correspondiente.
- i_total: se almacena la cantidad máxima de patinetes que hay a la estación correspondiente.
- l : se almacena las coordenadas exactas, latitud y longitud, para que después se pueda posar la marca de la estación correspondiente al mapa de la aplicación.
- name : se almacena el nombre de la estación.

4.2.2.2 Objecto Profile

En este objeto se almacena toda la información relacionada con el usuario. Está formada por un campo identificador denominado UID creado automáticamente por Firebase cuando se registra el usuario.

Cada objeto tiene definidas las siguientes propiedades:

- address: esta propiedad contiene la dirección completa donde vive el usuario.
- city: contiene la ciudad donde vive el usuario.
- date_reserved: se almacena la fecha en que el usuario ha hecho una reserva de patinete.
- email: se almacena el correo electrónico del usuario correspondiente.
- hour_reserved: se almacena la hora en que el usuario ha hecho una reserva de patinete.
- password: contiene la contraseña del usuario.
- reserved_booblean: nos indica si el usuario tiene una reserva activa o no.
- reserved_item_adress_station: se almacena la dirección de la estación donde el usuario ha hecho una reserva de patinete.
- reserved_item_station: se almacena el nombre de la estación donde el usuario ha hecho una reserva de patinete.
- username: contiene el nombre de usuario.
- zipCode: contiene el código postal introducido por el usuario.

4.2.2 Seguridad

Para realizar la autenticación de usuarios Firebase cuenta con un sistema de validación de usuarios mediante su correo electrónico y su contraseña. Además dispone del identificador único por usuario, UID, como verificador extra para garantizar su validación.

Firebase nos ofrece también un listado de los usuarios registrados al sistema además de fecha de creación del usuario, la fecha de cuando el usuario correspondiente inició su sesión por última vez y el origen de esta cuenta, Google, Facebook, Instagram o correo electrónico.

Al cargar la aplicación directamente se nos abrirá el Login, que después de introducir nuestras credenciales el servicio AngularFireAuth las validará mediante los métodos de signInWithEmailAndPassword o signInWithPopup(new auth.GoogleAuthProvider()) en el caso de login con una cuenta de Google, que nos creará una sesión y entonces podremos acceder en la página principal. El servicio de autenticación de Firebase se instala a través del mando npm, lo cual nos añadirá una nueva dependencia, este internamiento gestiona la lógica y que además nos permite usar el método authState para comprobar si el usuario está autenticado.

4.2.3 Diseño de la Interfaz

Según Georgia Arminsen el diseño [11] "*junto con la experiencia de navegación de la app, es el que el usuario percibe y, por lo tanto, lo cual hay que mimar, puesto que el usuario no voz todo el que pasa por debajo del contenido. Define al diseño mobile como el puente entre el interior y el exterior y asegura que es este el que ayuda al hecho que la experiencia de usuario sea completa*".

Por lo tanto, siguiendo estos criterios se ha realizado las siguientes configuraciones en cuanto que a diseño y experiencia de usuario:

- Colores: los colores de la aplicación serán azul (#0090d0) además de otro moratón más suave (#0091D2) por cabeceras y barras de navegación y el blanco para los fondos (#f7f7f7) puesto que he considerado que eran colores que al juntarlos no le hace al usuario la sensación de una experiencia desagradable al utilizarla o leer información. Respeto los textos los colores utilizados será el moratón antes mencionado si los fondos se blanco y será blanco si el fondo es azul. El resultado es un conjunto de colores instintivos y elegante según la librería material design [12].

- Navegación: para la navegación entre pantallas se ha decidido hacer un menú lateral, puesto que de manera muy intuitiva nos permite navegar por las diferentes secciones (Figura 2.5). El menú lateral nos permite de un solo toque cambiar fácilmente de pantalla y por tanto agilizar la consulta de información del usuario. Esta forma de navegación es la más empleada por muchísimas aplicaciones por su sencillez.

  - Diseño de página: cada página tiene una barra superior o cabecera que indica a qué plana se encuentra el usuario y si es una llanura a la cual se ha accedido por pulsar en una opción del menú lateral contiene un botón para volver a llanura anterior. Las llanuras basadas en formularios tienen cabeceras del color principal de la aplicación y si tienen una acción a realizar el botón será del mismo color.

Capítol 5

Ejecución de los casos de uso

En este apartado relacionaremos los casos de uso propuestos en la fase de análisis de requisitos con el proyecto finalizado, explicando escenario detrás escenario donde el usuario interactúa con la aplicación además se comentará la navegación entre páginas que estará acompañada de la explicación de cada pantalla de la aplicación.

5.1 Login

Al cargar la aplicación se mostrará la pantalla de login (Figura 5.1), aquí el usuario antes de acceder a la pantalla principal de la aplicación se validará con sus credenciales, correo electrónico y contraseña. Si falla su validación se mostrará un mensaje como que los datos introducidos no son correctos. El usuario nuevo que todavía no tiene una cuenta creada podrá acceder al registro pulsando en el botón de “registrarse”. El login de usuario corresponde al caso de uso CU08.

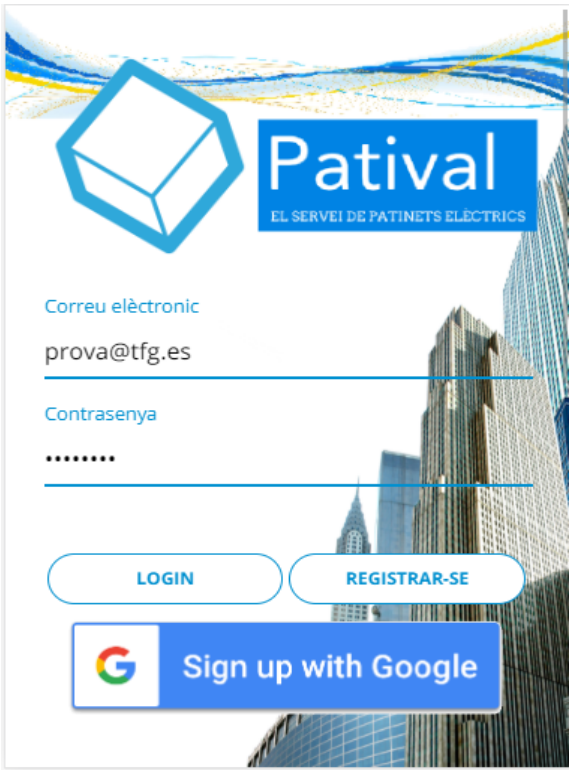

Figura 5.1: Pantalla de login de Patival

5.2 Registro

El usuario nuevo o no registrado antes a la aplicación podrá introducir tres parámetros, nombre de usuario, contraseña y correo electrónico que lo identificarán como usuario único e irrepetible (Figura 5.2). Una vez registrado se nos redirigirá a la pantalla principal de aplicación donde podremos empezar a utilizarla. El login de usuario corresponde al caso de uso CU09.

Figura 5.2: Pantalla de registre de Pativa

5.3 Reservar un patinete libre

Una vez validado el usuario correctamente la aplicación nos llevará a la pantalla principal (Figura 5.3). En esta se nos mostrará un mapa de Google maps donde automáticamente nos calculará la nuestra geo-posición. A continuación, podremos seleccionar una estación al mapa y nos aparecerá un modal de información sobre la estación correspondiente y sus patinetes. Para reservar un patinete libre habremos solo de hacer clic al botón de “reservar patinete” y confirmar que queremos hacerla.

En el caso de que no queden disponibles nos saldrá una alerta diciéndonos que no hay patinetes libres y por tanto no se puede hacer la reserva.

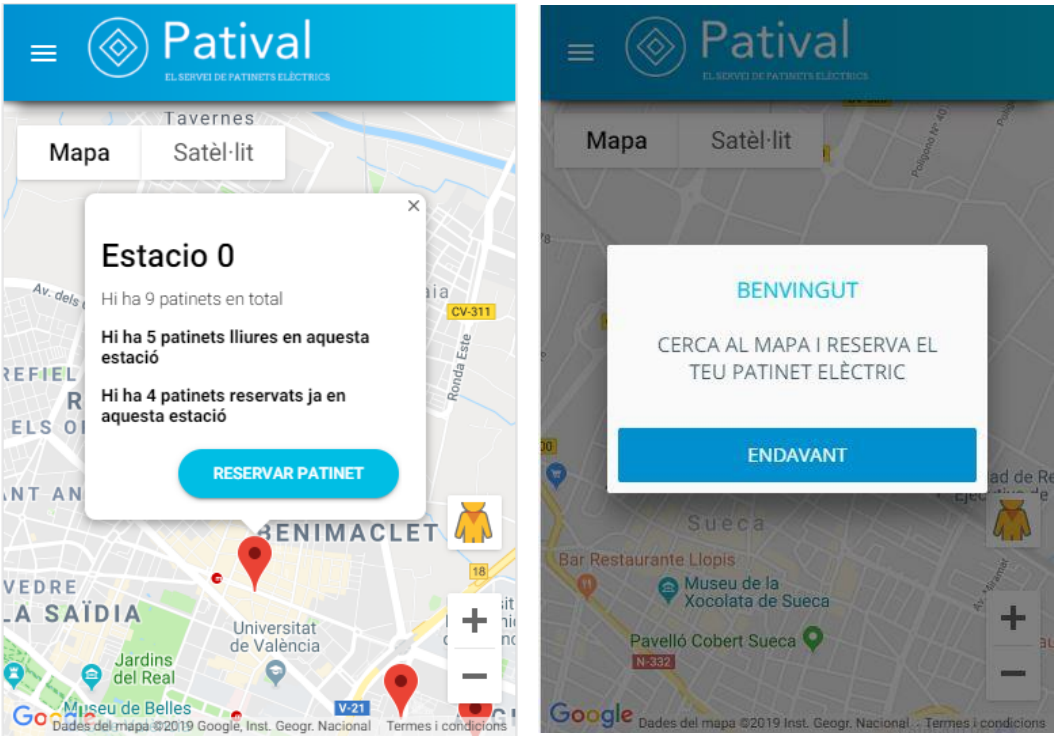

Figura 5.3: Pantalla principal de Pativa

5.4 Perfil

Para ver sus datos personales, el usuario tendrá que entrar al menú lateral pulsando en el icono superior izquierdo. Lo cual hará abrirlo y seleccionando Perfil nos llevará a la pantalla de perfil de usuario (Figura 5.4). Se nos mostrará la información correspondiente como también si tiene alguna reserva hecha.

En el caso de que quiera cambiar sus datos podrá hacerlo pulsando al Editar perfil. Al acabar, tendrá que pulsar a Editar y se nos confirmará que los datos han sido modificados correctamente (Figura 5.5).

Figura 5.4: Pantalla de perfil d'usuari de Patival

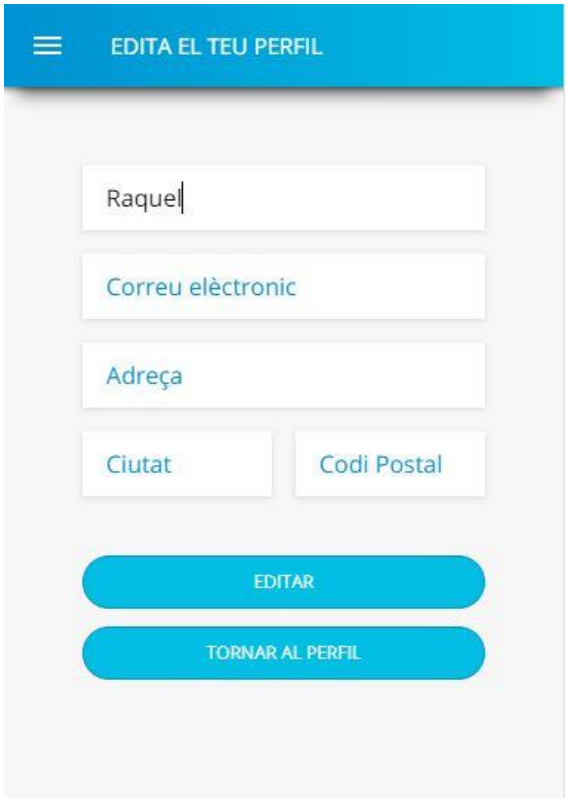

Figura 5.5: Pantalla d'edició de perfil d'usuari de Patival

5.5 Anular reserva

Esta es la pantalla donde el usuario podrá cancelar su reserva hecha anteriormente. Además, tendremos en un simple vistazo toda la información relacionada con la reserva, fecha, hora ,dirección, nombre de la estación. Al pulsar sobre Anular Reserva se nos mostrará un mensaje para confirmar o no la anulación de la reserva hecha.

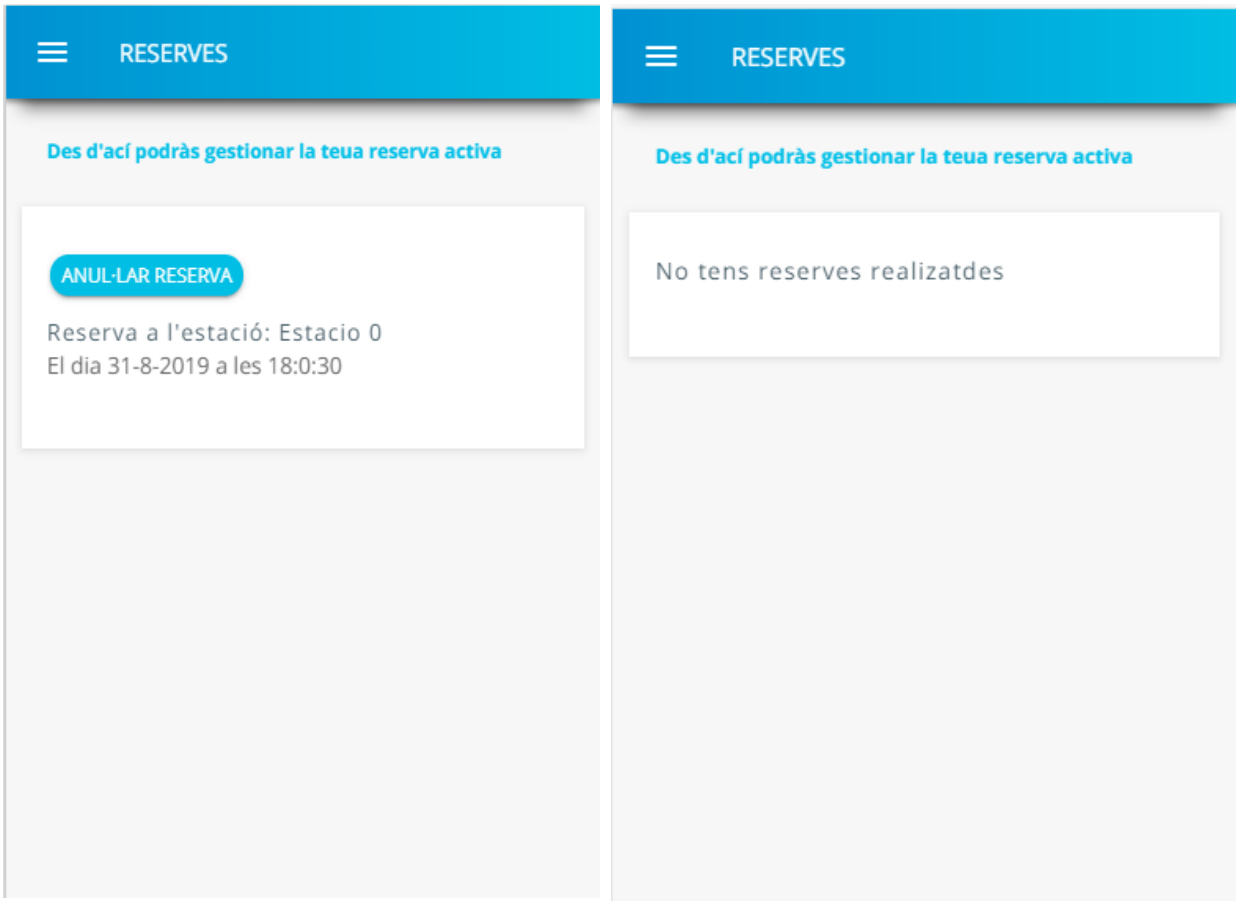

Figura 5.6: Pantalla cancelación de reserva de Patival

Capítulo 6
Conclusiones

El diseño y desarrollo de una aplicación partiendo de cero y con poca experiencia propia añadido a toda la tarea de aprender muchísimas tecnologías y frameworks utilizadas en un tiempo limitado puesto que tenía que trabajar y hacer el proyecto a la vez ha supuesto para mí un gran desafío. Aun así, ha sido un place poder hacerlo y saco una gran experiencia. Mi curva de aprendizaje en este sentido ha estado casi exponencial porque a comienzos de todo el proyecto cualquier problema o funcionalidad tardaba mucho de tiempo al resolverse y en las postrimerías aconteció justo el contrario, puesto que también la utilización de unas buenas prácticas de programación hizo que supiera fácilmente donde tenía que hacer cambios fácilmente sin tocar muchos archivos a la vez. Por ejemplo, el desarrollo de complementos para realizar las funcionalidades o también usar los servicios de Ionic que permiten compartir información entre vistas o complementos de manera sencilla, entonces cualquier cambio o error era fácil de detectar o hacer y por tanto no había que tocar nada más.

También todo no ha estado coser y cantar, durante el desarrollo ha habido momentos de frustración, sobre todo en la comunicación entre componentes también denominada entre padre e hijo. Aun así teniendo mucha paciencia probando posibles soluciones, haciendo investigaciones por internet o preguntando a mi tutora de prácticas que tiene suficiente experiencia, sobre todo en la parte de frontend, consiguieron resolver tanto este como otros problemas que surgieron. Todo esto ha permitido ensanchar mi conocimiento en todas las tecnologías utilizadas que son clave en lo mi laboral de horas de ahora.

En relación de este proyecto con la carrera de ingeniería informática y mi especialización, decir que buena parte de todas las cosas empleadas las he aprendido fuera de esta, sobre todo a los lugares donde he estado trabajando y también para ser autodidacta. La arquitectura, además de ciertos lenguajes y conceptos de programación si que venden de haber cursado el grado en esta universidad. Como Javascript, HTML, CSS o Java, puesto que los frameworks utilizados tienen como base algunos de estos o el conocimiento en Java que usa clases, herencias o variables tipadas ha ayudado a comprender TypeScript. Conceptos de diseño o el análisis de requisitos han sido estudiados en asignaturas como Interfaz Persona Computador o Desarrollo centrado en el usuario.

En resumidas cuentas, estoy muy satisfecho con todo el trabajo hecho en este proyecto tanto desde un punto de vista personal que sirve de aprendizaje personal y mejoramiento de mis conocimientos además de un reto personal importante por la cantidad de horas invertidas y como usuario de un servicio similar a la ciudad de Barcelona.

6.1 Trabajo futuro

La aplicación presentada para este TFG es una versión estable o producto mínimo viable, MVP, del que sería una versión final con los añadidos que se describirán más adelante, puesto que con el tiempo real propio disponible mientras trabajaba a la vez se acercaba demasiado a la fecha de entrega del TFG. Esto se hace notorio sobre todo en el hecho de poder añadir más funcionalidades y pulir todavía más ciertas que ya están implementadas. Así que como trabajo futuro, a continuación se expondrá un listado de mejoras que en una nueva versión serían implementadas.

6.2 Versión Futura

- Añadir un apartado de estaciones favoritas: el usuario mediante un icono de una estrella podría seleccionar qué estaciones le gustan más, utiliza o consulta información más a menudo. Se podría consultarlas en una opción en el menú lateral izquierdo cuando se despliega.

- Implementar un sistema de pago: cuando se hace la reserva tendrá la opción de hacer el pago del servicio directamente desde la aplicación o también implementar una suscripción mensual o trimestral para facilitarle la tarea a quién es un habitual de utilizar los patinetes.

- Permitir logarse con una cuenta de Facebook: esta característica está implementada, pero no se hace uso de ella, puesto que por motivos de espacio he decidido no utilizarla. Simplemente se poder dar la posibilidad de autenticación por Facebook igualmente como se hace con una cuenta de Google.

- Calcular una ruta: el usuario mediante una barra de busca donde poder escribir el punto de salida y el punto de llegada, la aplicación calculara la mejor ruta posible. También que mostrara las estaciones más próximas para hacer la ruta más rápida.

Bibliografía

[1] Piattini Mario G. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* RAMA Editorial, Madrid, primera edición, 1996.

[2] IEEE. "*IEEE Software Engineering Standards Collection 1999 Edition. Volume 1: Customer and Terminology Standards*". IEEE Computer Society Press, 1999.

[3] Mockups: la importancia de los bocetos web https://linube.com/blog/mockups-bocetos-web/.

[4] ¿Cómo elegir la mejor tecnología para mi aplicación? https://www.scio.com.mx/blog/como-elegir-la-mejor-tecnologia-para-mi-aplicacion/.

[5] ¿Qué es firebase y qué nos aporta? https://arpentechnologies.com/es/blog/aplicaciones-movil/que-es-firebase-y-que-nos-aporta/.

[6] Ionic 2, un framework diseñado para las aplicaciones híbridas que da el salto a las aplicaciones web https://www.arsys.es/blog/ programacion/ionic-2/.

[7] What does Software Architecture mean? https://www.techopedia.com/definition/24596/software-architecture.

[8] Firebase Realtime Database – Installation and Setup https://www.ficode.co.uk/firebase-realtime-database-installation-setup

[9] What is the structure of a Mobile App? https://www. quora.com/What-is-the-structure-of-a-Mobile-App.

[10] ¿Qué es desarrollo frontend? https://desarrollofrontend.com/que-es-desarrollo-frontend/.

[11] ¿Qué importancia tiene el diseño en el desarrollo de una App? Consultado el 01/09/18 https://slashmobility.com/blog/2014/04/que-importancia-tiene-el-diseno-en-el-desarrollo-de-una-app/.

[12] The color system https://material.io/design/color/ the-color-system.html

Apéndice A
Elementos del frontend

Este apéndice muestra un ejemplo de cada elemento de la parte frontend con su código fuente correspondiente.

A.1 Services

Se usará el servicio Auth-service como ejemplo.

- Auth-service.ts

```
import { AngularFireDatabase } from 'angularfire2/database';
import { AngularFireAuth } from 'angularfire2/auth';
import { AngularFirestore, AngularFirestoreDocument } from 'angularfire2/firestore';
import { Injectable } from '@angular/core';
import { User } from '../models/user.class';
import { auth } from 'firebase/app';
import { map, take } from 'rxjs/operators';
import { Observable } from 'rxjs/Observable';
import { AppSettings } from './app-settings'


@Injectable()
export class AuthService  {

  dbRef: any;
  user: any = [];

    constructor(public db: AngularFireDatabase, public afAuth: AngularFireAuth,
 public afs: AngularFirestore) {

    }

    //login
     Login(user:User){
        try {
            console.log(user);
            return this.afAuth.auth.signInWithEmailAndPassword(user.email.toString(),
user.password.toString());
        } catch (error) {
            console.log('Error login', error);
        }
    }

    //registrer
    Registrer(user:User){
        try {
            return
this.afAuth.auth.createUserAndRetrieveDataWithEmailAndPassword(user.email,
user.password);
        } catch (error) {
            console.log('Error registrer', error);
        }
    }


    registerUser(user:User) {
        return new Promise((resolve, reject) => {
          this.afAuth.auth.createUserWithEmailAndPassword(user.email.toString(),
user.password.toString())
            .then(userData => {
              resolve(userData);
              this.afAuth.authState.subscribe(auth => {
                if(auth) this.db.object(`profile/${auth.uid}`).update(user);
              })
            }).catch(err => console.log(reject(err)))
        });

      }

      loginEmailUser(user:User) {
```

```
56          return new Promise((resolve, reject) => {
57            this.afAuth.auth.signInWithEmailAndPassword(user.email.toString(),
user.password.toString())
58               .then(userData => resolve(userData),
59               err => reject(err));
60          });
61        }
62
63        loginFacebookUser() {
64          return this.afAuth.auth.signInWithPopup(new auth.FacebookAuthProvider());
65           // .then(credential => this.updateUserData(credential.user))
66        }
67
68        loginGoogleUser() {
69          return this.afAuth.auth.signInWithPopup(new auth.GoogleAuthProvider())
70          .then(credential => {
71            this.afAuth.authState.pipe(
72              take(1),
73            ).subscribe(auth => {
74                    if(auth) {
75
76
 this.db.list(`/profile/${credential.user.uid}`).valueChanges().pipe(
77                        take(1),
78                      ).subscribe(info_user => {
79
80                            var data: User = {
81                              id: credential.user.uid,
82                              email: credential.user.email,
83                              username: credential.user.displayName,
84                              money : 0,
85                              reserved_item_adress_station: "",
86                              reserved_item_station: "",
87                              date_reserved: "",
88                              hour_reserved: "",
89                              reserved_boolean: 0,
90                              address: "",
91                              city: "",
92                              zipCode : ""
93                            }
94
95                          if(info_user.length == 9) {
96                            this.afs.doc(`profile/${auth.uid}`).set(data, { merge: true
})
97                          }else {
98                            this.db.object(`profile/${auth.uid}`).update(data);
99                          }
100
101                       });
102
103
104                 }
105             })
106
107         })
108       }
109
110       logoutUser() {
111         return this.afAuth.auth.signOut();
112       }
```

```
113
114       isAuth() {
115         return this.afAuth.authState.pipe(map(auth => auth));
116       }
117
118
119
120 }
121
```

A.2 Page

Se utilizará la página Hombre como ejemplo.

- Home.ts

```
import { Component, OnInit } from '@angular/core';
import { IonicPage, NavController } from 'ionic-angular';
import { HttpClient } from '@angular/common/http';
import { HomeService } from '../../services/home-service';
import { AppSettings } from '../../services/app-settings';
import { MapsService } from '../../services/maps-service';
import { AuthService } from '../../services/auth-service';

interface Location {
  latitude: number;
  longitude: number;
}

@IonicPage()
@Component({
  selector: 'page-home',
  templateUrl: 'home.html',
  providers: [HomeService, MapsService, AuthService]

})

export class HomePage implements OnInit{

  data: any = {};
  params_map: any = {};
  params_alert: any = {};
  lat: number;
  lng: number;
  public isLogged: any = false;

  constructor(public navCtrl: NavController, public service: HomeService, private
http: HttpClient, private geo: MapsService, private authSvc: AuthService) {
    service.load().subscribe(snapshot => {
      this.data = snapshot;
    });
    // this.setUserLocationOnMap();
  }

  ngOnInit(){
    this.params_alert.events = true;
    this.params_alert.view = "Home";
    this.getUserLocation();
  }

  private getUserLocation(){

    if(navigator.geolocation){
      navigator.geolocation.getCurrentPosition(position => {

        this.lat = position.coords.latitude;
        this.lng = position.coords.longitude;

        console.log(this.lat,this.lng);
          this.params_map.data = {
          "map": {
              "lat": this.lat,
              "lng": this.lng,
              "zoom": 15,
              "mapTypeControl": true,
              "streetViewControl": true
```

- Home.html

```
<ion-header>
    <ion-navbar>
        <button ion-button menuToggle>
            <ion-icon class="icon-menu" name="menu"></ion-icon>
        </button>

        <img [src]="data.logo" alt="">

    </ion-navbar>
</ion-header>

<ion-content>

        <maps-layout-full
        [data]="params_map.data">
        </maps-layout-full>

        <alert-info
        [events]="params_alert.events"
        [view]="params_alert.view">
        </alert-info>
</ion-content>
```

- Home.module.ts

```
import { NgModule, CUSTOM_ELEMENTS_SCHEMA } from '@angular/core';
import { IonicPageModule } from 'ionic-angular';
import { HomePage } from './home';
import { MapsLayoutFullModule } from '../../components/maps/layout-full/maps-layout-full.module';
import { AlertInfoModule } from '../../components/alert/info/alert-info.module';

@NgModule({
    declarations: [
        HomePage
    ],
    imports: [
        IonicPageModule.forChild(HomePage),
        AlertInfoModule,
        MapsLayoutFullModule
    ],
    schemas: [CUSTOM_ELEMENTS_SCHEMA]
})

export class HomePageModule { }
```