UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITECNICA SUPERIOR DE GANDIA

GRADO EN ING. SIST. DE TELECOM., SONIDO E IMAGEN

"Desarrollo de una aplicación para la asignación de plazas de aparcamiento de un hotel"

TRABAJO FINAL DE GRADO

Autor/a:

Joan Tomás Martínez

Tutor/a:

Jordi Bataller Mascarell

GANDIA, 2020

RESUMEN

El propósito de este TFG es la creación de una aplicación para la gestión de un aparcamiento en el hotel donde he realizado las practicas curriculares. De esta forma mejorar el sistema actual y ayudar con la transición a un uso de las nuevas tecnologías. La aplicación debe ser capaz de controlar el estado de las plazas de aparcamiento junto con los datos asignados a esta. El desarrollo estará realizado con el *framework* Ionic, que está basado en Angular, y haciendo uso de Firebase para tener una base de datos en tiempo real.

PALABRAS CLAVE

Aplicación web, PWA (Aplicación Web Progresiva), Angular, Ionic, Firebase.

ABSTRACT

The purpose of this TFG is the development of an application for the management of the hotel's parking lot where I had my internship. The goa lis to improve the current system and help with the transition to use new technologies. The application must be able to control the status of the parking places with the data assigned to it. The development will be done using Ionic, which is based on Angular.Js, and Firebase will be used to have a real-time database.

KEYWORDS

Web application, PWA (Progressive Web Application), Angular, Ionic, Firebase.

ÍNDICE

ÍNDICE DE ILUSTRACIONES

1. INTRODUCCIÓN

El propósito de este documento es presentar los aspectos destacados del desarrollo de una aplicación para la gestión de las plazas de aparcamiento en el Oliva Nova Beach & Golf Resort, donde se realizaron las prácticas curriculares del plan de estudios de la titulación.

En el citado hotel, la gestión del aparcamiento era completamente manual. En concreto, a la llegada de un cliente a la puerta principal el personal del hotel le indicaba el número de plaza de aparcamiento que le correspondería. Estos empleados mediante una plantilla en papel sabían qué plazas estaban ocupadas y cuales no para poder realizar esta asignación. Esta forma de proceder planteaba muchos inconvenientes, como que distintos empleados asignaran la misma plaza a clientes distintos en momentos de gran afluencia o que la asignación de plaza tenía que ser comunicada mediante las plantillas en la recepción.

Ante esta situación se nos ocurre proponer a la administración del hotel el desarrollo de una aplicación cliente-servidor en la cual el personal de la entrada del hotel pueda cómodamente desde un teléfono móvil gestionar la asignación y consultar la disponibilidad de las plazas de aparcamiento. Simultáneamente el personal de recepción puede fácilmente verificar si el cliente ha ocupado una plaza de aparcamiento para incorporarlo a los datos de la reserva.

Como hemos comentado, la aplicación, que va a ser utilizada únicamente por el personal del hotel, permite la asignación de plazas para lo cual es necesario poder registrar nuevos clientes; pero también la identidad de los empleados que la usan. Con respecto a la asignación de las plazas será posible cambiar a un cliente de plaza, ampliar o reducir las fechas asignadas, vincular la habitación con la plaza y por supuesto, dejarla libre cuando el cliente abandone el hotel.

Nos hemos esforzado para que la usabilidad de la aplicación sea cómoda e intuitiva, esté disponible en varios lenguajes y, especialmente, que su manejo sea cómodo en momentos de gran afluencia al hotel. Además, estará disponible para varias plataformas, tanto teléfonos móviles (Android e iOS) como para ordenador mediante página web.

Entre los beneficios de la implantación de nuestra herramienta nos proponemos:

- Extender el uso de la tecnología entre todos los usuarios del departamento involucrado.
- Mejorar la calidad de la comunicación entre los trabajadores.
- Facilitar la realización del trabajo diario.
- Obtener información y estado en tiempo real de la instalación.

- Y no menos importante, una reducción considerable los recursos de impresión, con todo esto conseguiríamos aumentar el tiempo para otros trabajos y reducir la huella ecológica.

1.1. ORGANIZACIÓN DE LA MEMORIA

Este documento está organizado de la siguiente forma:

- En el segundo capítulo, Análisis de Requerimientos, detallamos las características y objetivos que debe cumplir la aplicación.

- El tercer punto de la memoria, Diseño de la Aplicación, mostraremos el diseño y arquitectura de la aplicación, tanto de la base de datos como de la capa visual con la que interaccionan los usuarios del hotel.

- La implementación de la aplicación se detallará en el cuarto punto de la memoria. Haremos un breve repaso de las tecnologías utilizadas.

- En el quinto capítulo, Instalación y Uso de la Aplicación, explicamos tanto la forma de instalar la aplicación como su manual de uso cotidiano para el personal del hotel.

Finalmente, en el sexto capítulo, exponemos las conclusiones de todo nuestro trabajo y planteamos algunas propuestas de ampliación y mejora como trabajo futuro.

2. ANÁLISIS DE REQUERIMIENTOS

En este capítulo de la memoria describiremos las características que debe tener la aplicación, también los objetivos que debe cumplir.

En primer lugar, debe ser una aplicación a tiempo real, se necesita que todos los usuarios involucrados en el uso del programa dispongan de la misma información en todo momento.

Debe ser capaz de mostrar todas las plazas de aparcamiento pudiendo configurar en cada una de ellas distintos valores como son las fechas de uso.

Por último, debe tener una interfaz gráfica intuitiva y sencilla para garantizar un uso eficiente de la misma y del establecimiento que se va a gestionar.

Para poder conseguir estos objetivos se plantean unos puntos sobre los cuales construir la aplicación:

- Estudio y comprensión de la programación en Angular e Ionic.
- Uso de la herramienta Firebase

Una vez definidas las herramientas que se van a utilizar para el correcto desarrollo del proyecto, el primer paso a la hora de crear la aplicación web es esbozar un boceto de la apariencia que va a tener.

Antes de crear el proyecto se dejó la hoja de ruta para crear cada una de las pantallas y funcionalidades.

Objetivos primarios a la hora de desarrollar la aplicación:

- Pantalla principal con visualización del aparcamiento.
- Listado de las plazas de aparcamiento.

Objetivos secundarios:

- Pantalla de inicio de sesión.
- Pantalla de registro de usuarios.
- Menú desplegable.

En la ilustración 1, observamos un primer boceto a baja fidelidad de cómo debería ser la aplicación.

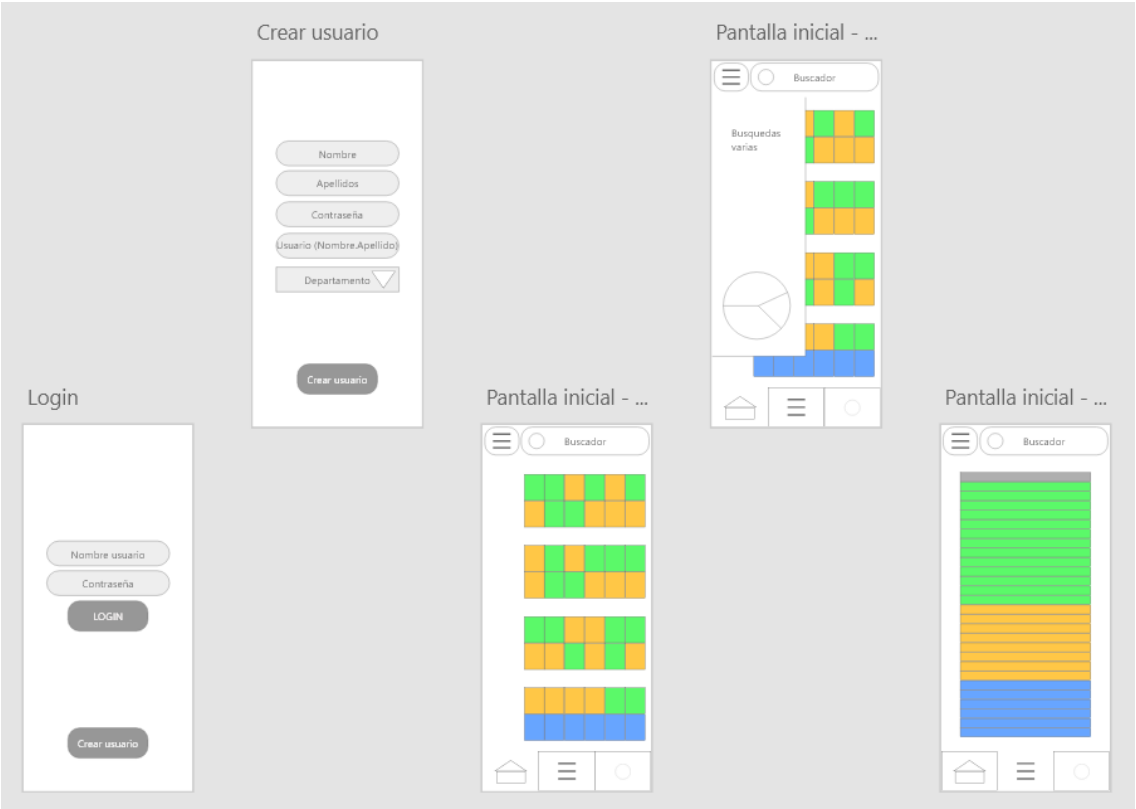


ILUSTRACIÓN 1: BOCETO A BAJA FIDELIDAD COMPLETO.

Después de tener este primer esbozo de la aplicación a baja fidelidad, se hizo un boceto intentando que tuviese las máximas funcionalidades representadas.

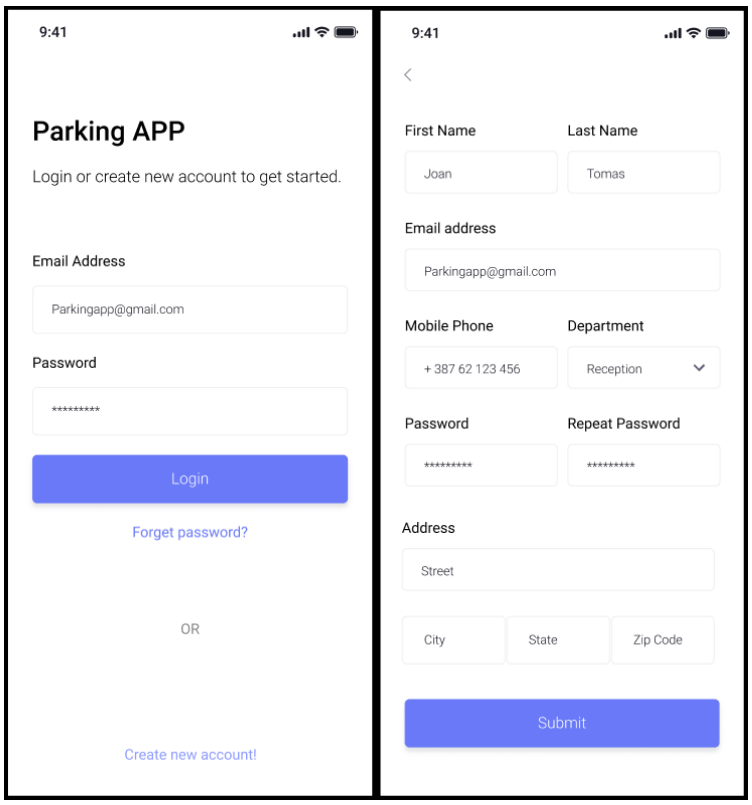
En este segundo boceto de alta fidelidad ya podemos observar algunas de las funcionalidades que se implementaran en la aplicación:

- Una página de inicio de sesión donde poder entrar a la aplicación mediante correo electrónico y contraseña.

- Un formulario al cual acceder desde la primera página para poder registrar a los usuarios.

- Las dos pantallas principales de la aplicación donde poder observar la distribución del aparcamiento y su estado a tiempo real y un listado de todas las plazas de aparcamiento con el estado y algunos de los atributos más importantes como la fecha de entrada y salida.

- Por último, una página dedicada a los ajustes de la aplicación.



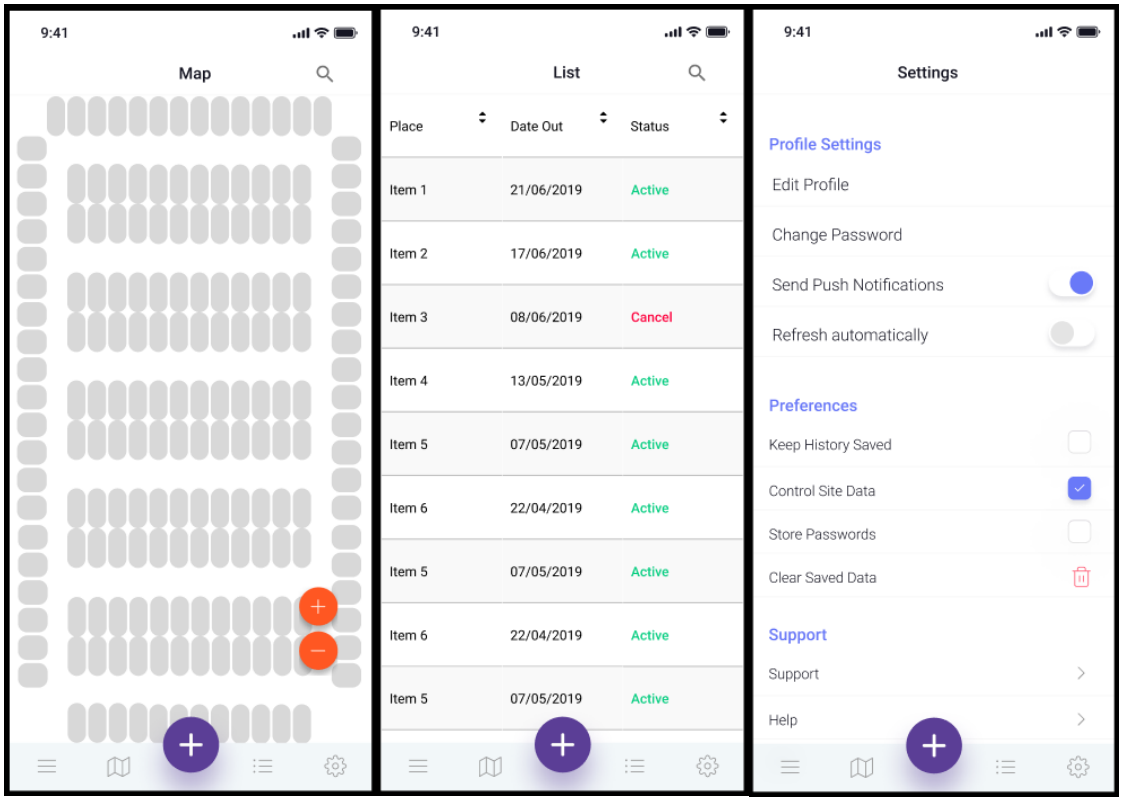


ILUSTRACIÓN 2: BOCETO A ALTA FIDELIDAD COMPLETO.

La pantalla que se presenta al ejecutar la aplicación, Inicio de sesión, requerirá de los siguientes elementos:

- Dos campos donde poder introducir texto, el primero será para introducir la variable de usuario, en este caso elegimos el correo electrónico, y el segundo campo servirá para escribir la contraseña.

- Por otra parte, se necesitará un botón el cual su función deberá ser verificar los dos campos de texto y dar acceso a la pantalla principal de la aplicación. También se necesitará otro botón que lleve a la pantalla de registro, en este caso representado en la parte inferior de la ilustración, *Create new account!*.

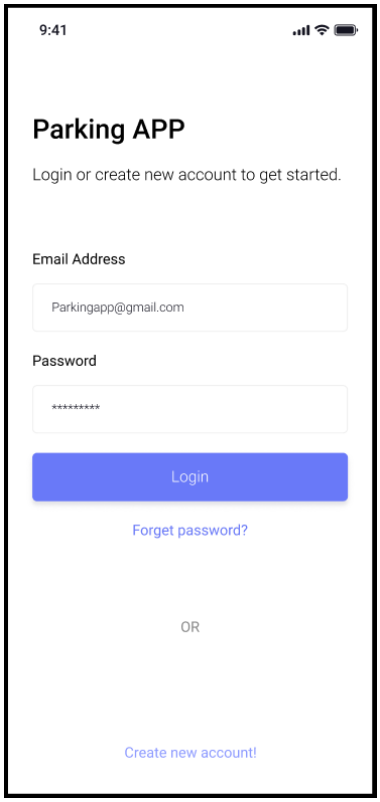


ILUSTRACIÓN 3: BOCETO DE INICIO DE SESIÓN.

El módulo de registro se diseñará mediante la implementación de un formulario, este formulario tendrá campos obligatorios y campos libres, una vez los campos obligatorios del formulario estén completos correctamente el botón para crear el usuario se activará y permitirá el envío de este.

Los campos obligatorios del formulario serán:

- Nombre.
- Apellidos.
- Email.
- Número de teléfono.
- Departamento.
- Contraseña.

Mientras que los campos que definimos en la dirección postal; Calle, Ciudad, Estado y Código postal, quedarían como opcionales.

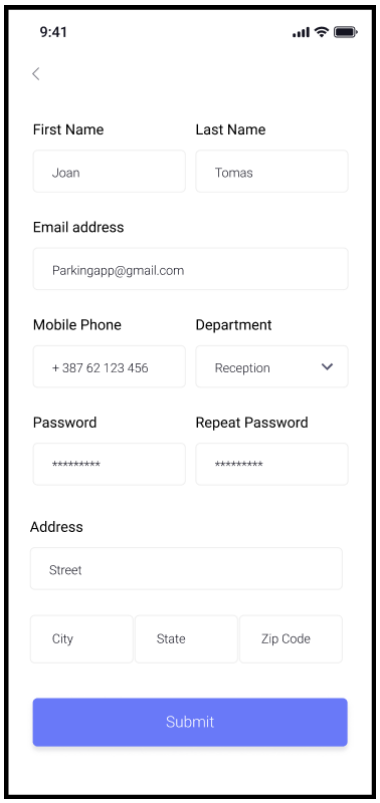


ILUSTRACIÓN 4: BOCETO DE REGISTRO.

La pantalla principal estará compuesta por un mapa interactivo, estará formado por botones que representarán las plazas del aparcamiento, los cuales permitirán acceder a los atributos de estas.

Esta pantalla principal y la secundaria compartirán un botón en la parte inferior central, el cual servirá para desplegar una ventana modal y servirá para editar las plazas de aparcamiento.

Una ventana modal es un cuadro que aparece en la pantalla, bloqueando toda funcionalidad para concentrar el foco en una acción en particular, en este caso poder crear o editar las plazas de aparcamiento.

En la pantalla secundaria aparecerá un listado, al igual que en la principal este listado será interactivo y se podrán editar los atributos de las plazas del aparcamiento.

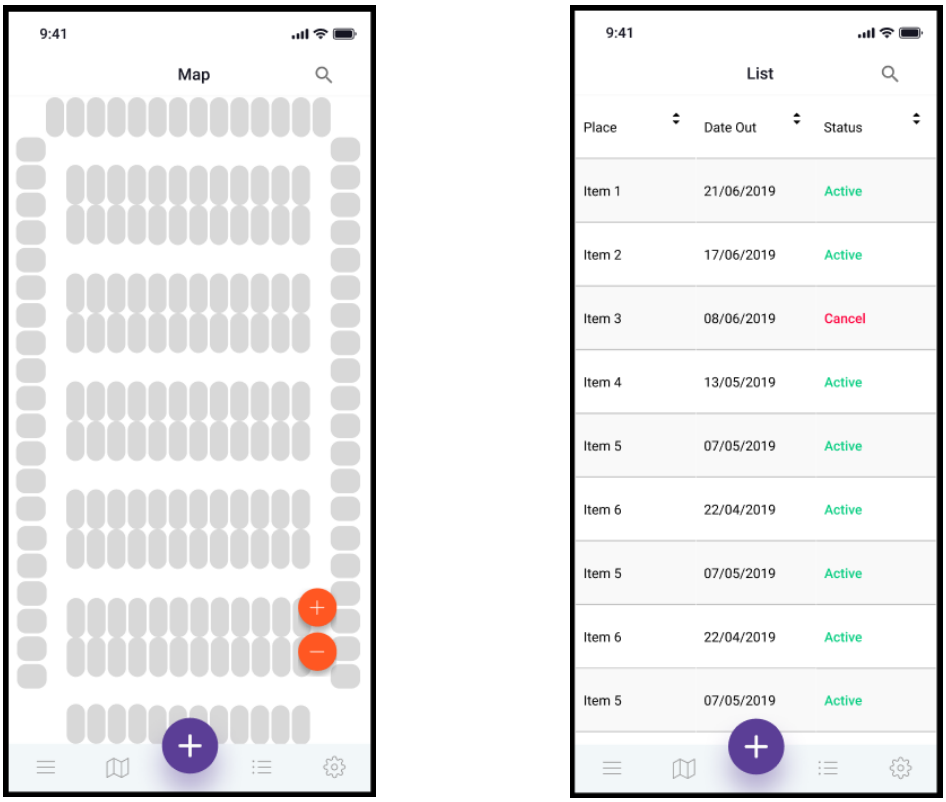


ILUSTRACIÓN 5: BOCETOS PANTALLAS PRINCIPALES, MAPA Y LISTA.

3. DISEÑO DE LA APLICACIÓN

En este capítulo, Diseño de la Aplicación, presentaremos el diseño interno del proyecto. En una primera parte expondremos la arquitectura en la cual está basada la aplicación, posteriormente explicaremos como hemos definido la estructura de la base de datos y finalmente definiremos algunas de las principales funciones lógicas de la aplicación.

3.1. ARQUITECTURA DE LA APLICACIÓN

La ilustración 6, muestra el esquema general de la aplicación y en él podemos observar cómo se comunican entre sí el *frontend* y el *backend*, también hemos detallado cómo se relacionan las credenciales de usuario en la base de datos.

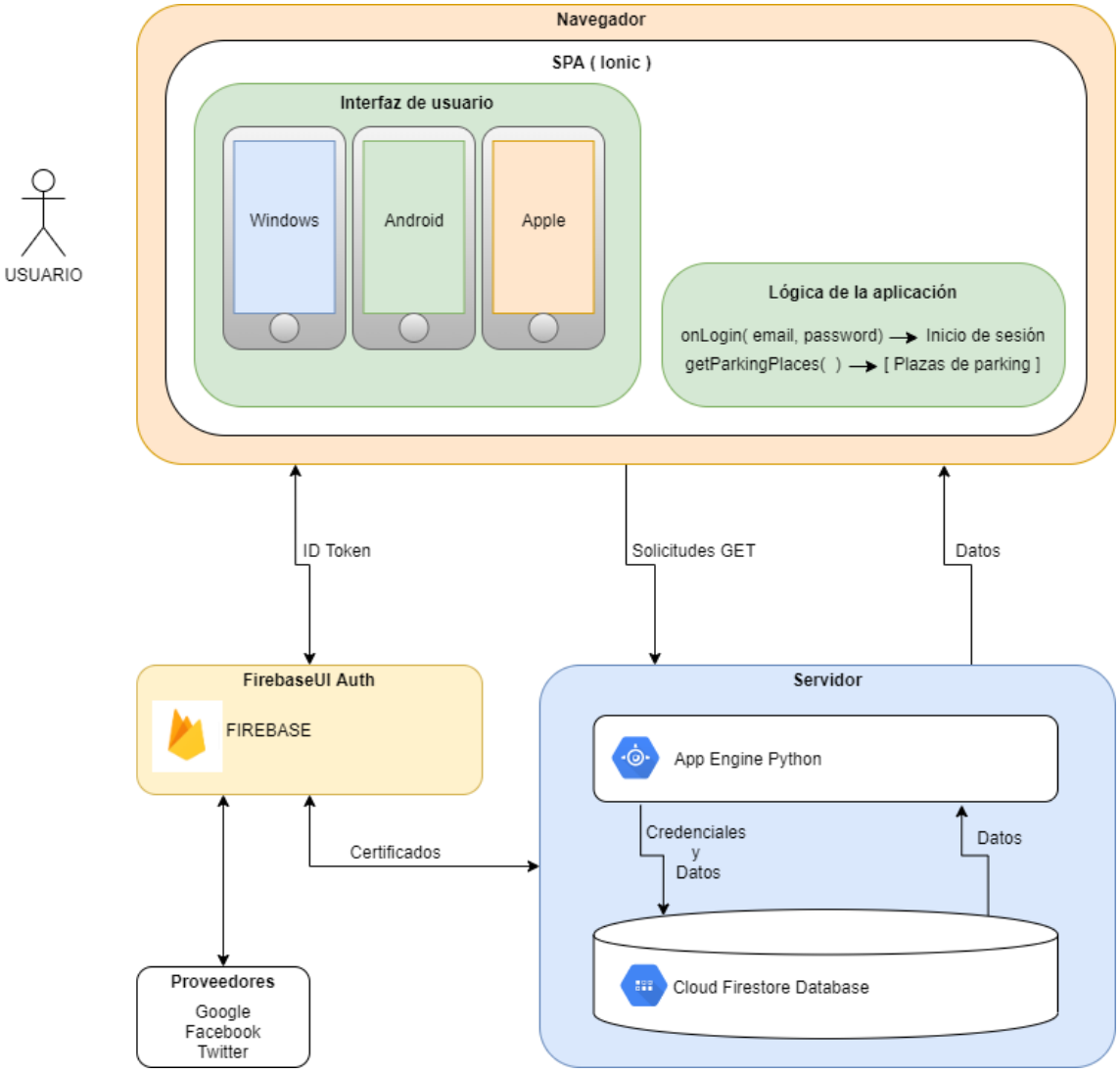


ILUSTRACIÓN 6: ESQUEMA ARQUITECTURA DE LA APLICACIÓN.

El primer bloque, nombrado navegador, es donde situamos el código de la aplicación destinado a la parte visual y a la lógica de la aplicación. Esta parte se denomina *Frontend*, es la parte donde el usuario interactúa con el programa. Aparte de implementar las funcionalidades estéticas de la aplicación, es donde está configurado el acceso de usuario y la lógica de la aplicación, facilitando la comunicación con el servidor.

Los dos bloques siguientes, están integrados dentro de la plataforma de herramientas Firebase. En amarillo, el bloque nombrado FirebaseUI Auth, es una solución de autenticación de código abierto que controla el iniciar sesión en los usuarios mediante correo electrónico y contraseña. Implementa recomendaciones de autenticación para una experiencia de acceso segura. El bloque azul, servidor, es donde está la base de datos Cloud Firestore. Aparte de esperar las peticiones por parte del usuario mediante las funciones lógicas, también verifica el estado de autenticación del usuario.

3.2. DISEÑO DE LA BASE DE DATOS

Para presentar el diseño de las estructuras de información de la aplicación se ha hecho uso de JDL (1), JDL es un lenguaje de dominio específico donde se pueden describir las entidades y sus relaciones en un archivo con una sintaxis muy descriptiva. Esto nos ayudará a plantear el diseño de la base de datos y evitar problemas de recursión o redundancia.

Por un lado, se definen las entidades que se van a tener en la aplicación con sus atributos, se crean las variables si son necesarias y por último se escribe la relación que hay entre las entidades.

Las entidades que hemos diseñado para nuestra aplicación podemos verlas en el siguiente diagrama:

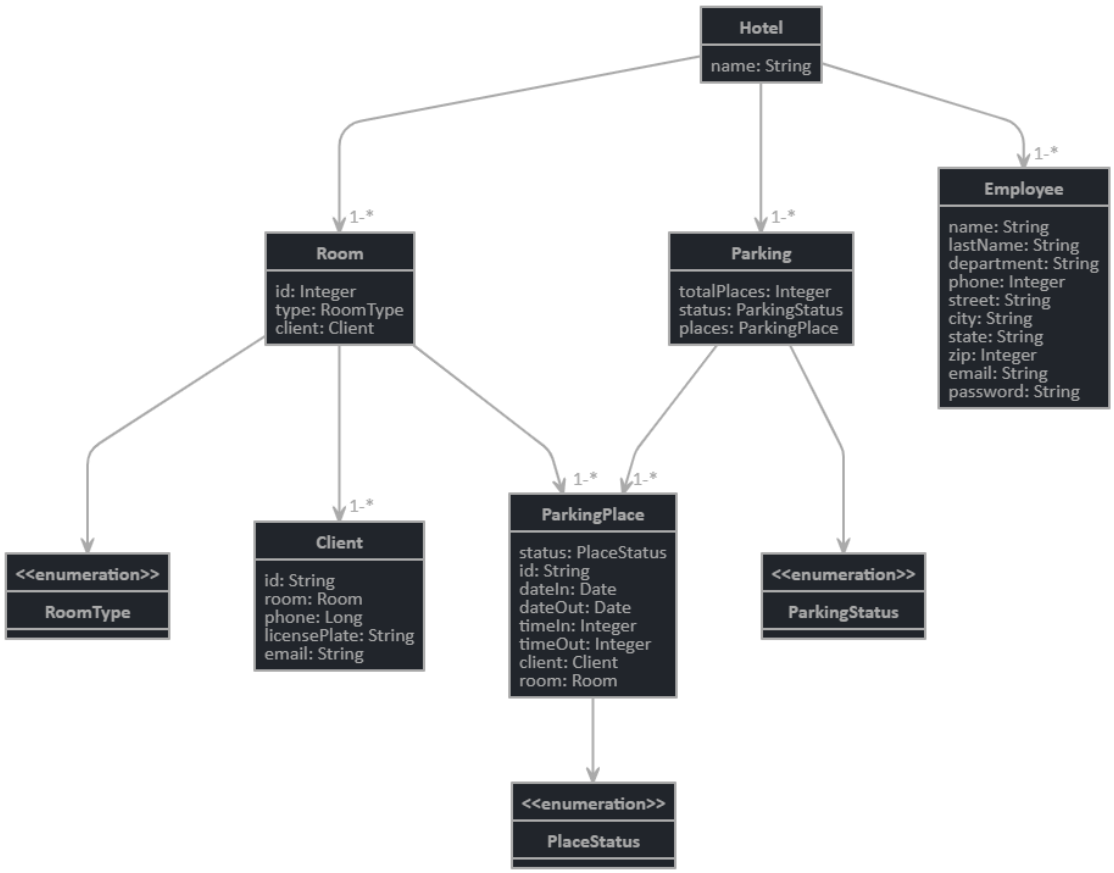


ILUSTRACIÓN 7: ESQUEMA DESCRIPCIÓN ENTIDADES

Para desarrollar este esquema hemos ido paso por paso estructurando la aplicación, pensada principalmente para un hotel, situamos esta entidad en la parte superior del esquema, este hotel tiene un aparcamiento, habitaciones y empleados, definidos como Room, Parking y Employee, como vemos desde la cajita superior salen estas tres ramas con las entidades creadas y sus respectivos atributos.

En el tercer nivel del esquema, hemos creado la entidad cliente, Client, la cual depende de Room y una entidad, ParkingPlace, la cual forma parte de las entidades superiores Parking y Room. También tenemos definidos tres tipos de variables para los atributos RoomType, PlaceStatus y ParkingStatus.

Guiados por este esquema, hemos creado las siguientes colecciones en nuestra base de datos, la primera colección, Employees, se ha creado para almacenar los atributos referentes a los usuarios de la aplicación, como hemos definido en el esquema JDL, los atributos de esta colección son:

- Nombre.

- Apellidos.
- Departamento.
- Número de teléfono.
- Email.
- Contraseña.
- Ciudad.
- Estado.
- Código postal.

Un ejemplo de un documento de la colección Employees sería la siguiente ilustración:

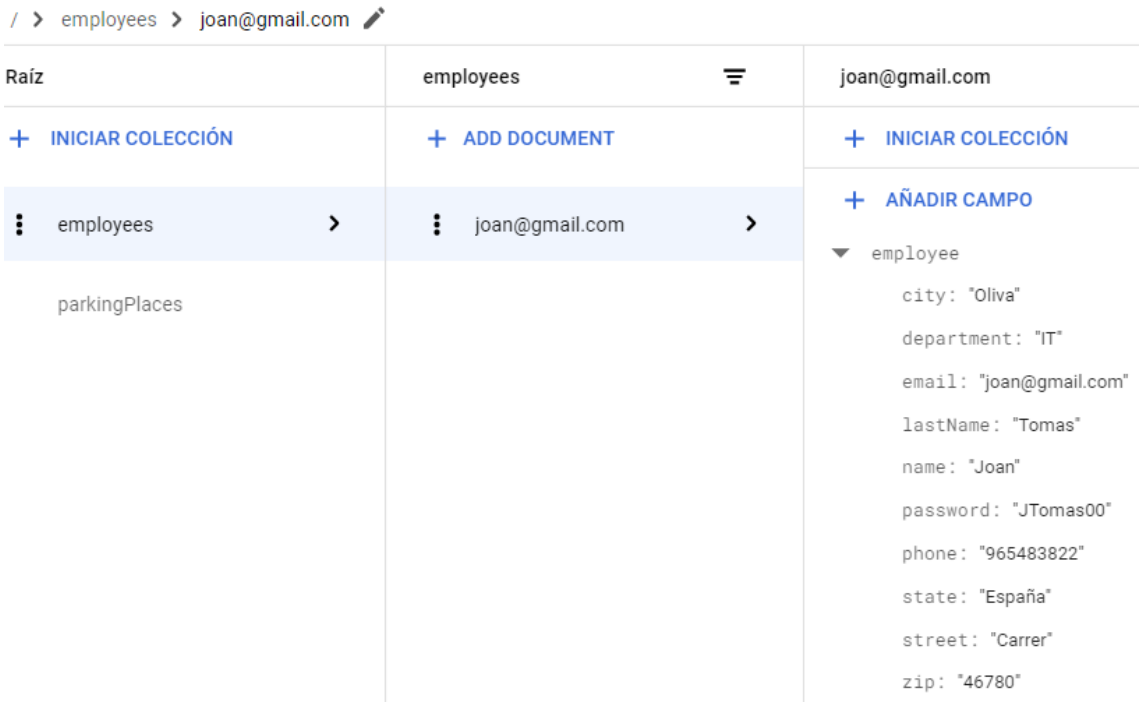


ILUSTRACIÓN 8: BBDD, COLECCIÓN EMPLOYEES.

La siguiente colección, ParkingPlaces, está creada para almacenar todos los datos relacionados con las plazas de aparcamiento. Los datos almacenados en los documentos de esta colección se han estructurado de la siguiente forma:

- Identificador de la plaza
- Fecha de entrada
- Fecha de salida
- Hora de entrada
- Hora de salida
- Estatus de la plaza
- Cliente
    - Nombre
    - Correo electrónico
    - Número de teléfono
    - Matricula
- Habitación
    - Número de habitación
    - Tipo de habitación

El ejemplo de como queda un documento de la colección ParkingPlaces es la siguiente ilustración:

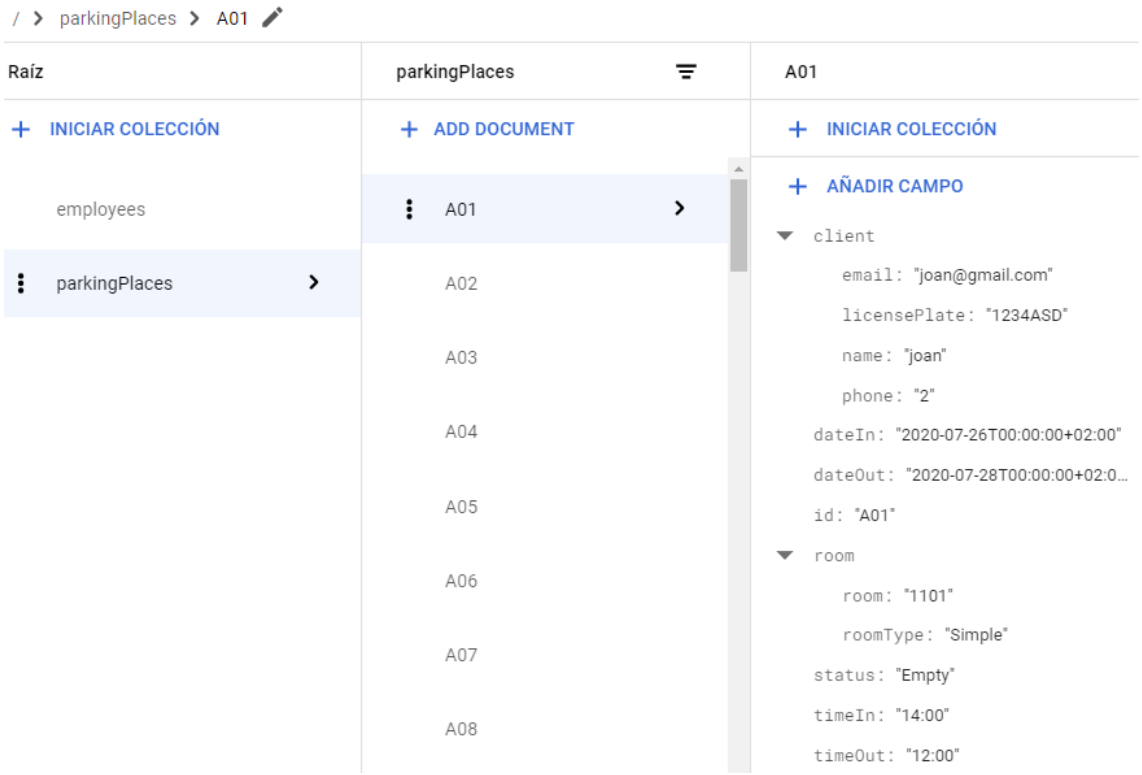


ILUSTRACIÓN 9: BBDD, COLECCIÓN PARKINGPLACES.

3.3. DISEÑO DE LA LÓGICA DE LA APLICACIÓN

En este punto del diseño de la aplicación vamos a presentar algunas de las funciones lógicas más relevantes que hemos necesitado implementar para el funcionamiento de la aplicación.

Para la parte de la aplicación donde se necesita la autenticación de usuarios hemos creado dos funciones, que, aunque su trabajo es diferente, se implementan de igual forma, onLogin() y onRegister(). Como podemos deducir del nombre de la función sirven para iniciar sesión la primera y para registrarse en la aplicación la segunda.

Las dos tienen el mismo esquema, necesitamos darle un objeto usuario, donde están definidos los campos correo electrónico y contraseña. Esta verificará los parámetros y devolverá un mensaje verificando el acceso o denegándolo.

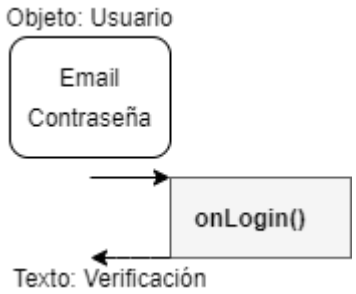


ILUSTRACIÓN 10: FUNCIÓN ONLOGIN.

Para la posible comunicación con la base de datos hemos tenido que realizar las siguientes funciones:

Añadir empleado, la hemos nombrado como addEmployee(), esta función necesita como parámetro de entrada un objeto de tipo empleado que hemos definido y estará encargada de enviar y almacenar en la base de datos la información referente a este objeto. Como en la anterior función, se devolverá un mensaje verificando o denegando el éxito de la operación.

ILUSTRACIÓN 11: FUNCIÓN ADDEMPLOYEE.

Añadir plaza de *parking*, esta función la hemos nombrado addParkingPlace(), como la anterior necesita un parámetro de entrada, esta vez un objeto de tipo parkingplace y está encargada de crear las plazas de aparcamiento de la aplicación con sus atributos.

ILUSTRACIÓN 12: FUNCIÓN ADDPARKINGPLACE.

Obtener las plazas del aparcamiento, a esta función la hemos llamado getParkingPlaces(), está encargada de proporcionarnos todas las plazas del aparcamiento que hay registradas en la base de datos. Creará un array de objetos, al cual nosotros haremos las llamadas, ahorrando carga directamente a la base de datos.

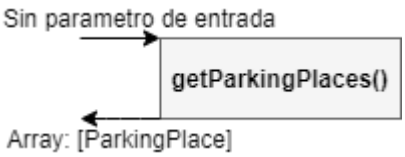


ILUSTRACIÓN 13: FUNCIÓN GETPARKINGPLACES.

4. IMPLEMENTACIÓN

En este capítulo, Implementación, expondremos como hemos ido desarrollando la aplicación, empezando por un breve resumen de las tecnologías usadas seguido de una detallada explicación de la estructura del código del proyecto, hasta exponer los diferentes problemas a los que nos hemos enfrentado y solucionado durante el transcurso del proyecto

4.1. HERRAMIENTAS Y TECNOLOGÍA UTILIZADAS

Algunos de los entornos de desarrollo para Javascript más utilizados actualmente son Vue, React y Angular. Este último, (2) es el que hemos utilizado para nuestro proyecto. Fue creado por Google y se considera uno de los marcos de trabajo más potentes y utilizados. Con él se pueden crear aplicaciones web de una sola página (SPA).

Angular es un *framework* de desarrollo creado por Google para JavaScript. Este nos facilita el desarrollo de aplicaciones web SPA (*Single-page application*), webs de una sola página en la cual la navegación entre secciones se realiza de forma dinámica, casi instantánea y sin refrescar la página en ningún momento.

Su mayor utilidad es la incorporación de los tipos de datos estáticos, que se pueden utilizar de manera opcional, aunque siempre recomendada. Javascript no permite etiquetar las variables de manera estática, mientras que Typescript sí, por lo que permite convertir Javascript en un lenguaje fuertemente *tipado*.

Ionic (3) es marco de trabajo complementario a Angular, que permite que una aplicación web pueda ser ejecutada en dispositivos móviles con la apariencia de una aplicación nativa.

Algunas de las ventajas de utilizar Ionic son las siguientes:

- Desde una única fuente podemos llegar a plataformas que soportan este *framework* (Android e iOS).
- El desarrollo se realiza en HTML junto con CSS y JS, lenguajes muy extendidos.

Aunque pocas, también hay desventajas a la hora de utilizar estos *frameworks*:

- El rendimiento suele ser peor que en aplicaciones desarrolladas de forma nativa.
- Al ser una herramienta "joven" puede ser difícil encontrar módulos compartidos.
- Debido a esta juventud, Ionic tiene un cambio constante en algunas de sus características y normas, por lo que es necesario estar actualizado.

Firebase (4) es una plataforma para el desarrollo de aplicaciones móviles y web en la nube de Google. Entre los servicios que ofrece esta plataforma hay bases de datos en tiempo real y autentificación de usuarios, los cuales se han utilizado para desarrollar la aplicación.

Firebase nace como una base de datos en tiempo real la cual podías desarrollar en el lado del cliente sin necesidad de emplear un servidor. Con este concepto surge la creación de una serie de herramientas adicionales que hoy en día forman esta plataforma en la nube.

Cloud Firestore (5) es una base de datos NoSQL que nos proporciona Google en el servicio Firebase el cual permite almacenar y sincronizar datos entre los usuarios y los dispositivos, alojada en la nube a la cual pueden acceder aplicaciones iOS, Android y Web.

También se ha hecho uso de Visual Studio Code (6), el cual es un editor de código fuente desarrollado por Microsoft. Este software incluye soporte para depuración, control de Git, resaltado de sintaxis, finalización de código inteligente y refactorización de código.

4.2. ESTRUCTURA DEL CÓDIGO DEL PROYECTO

La estructura de los proyectos en Ionic sigue los estándares recomendados por Angular, y nuestro proyecto no es una excepción.

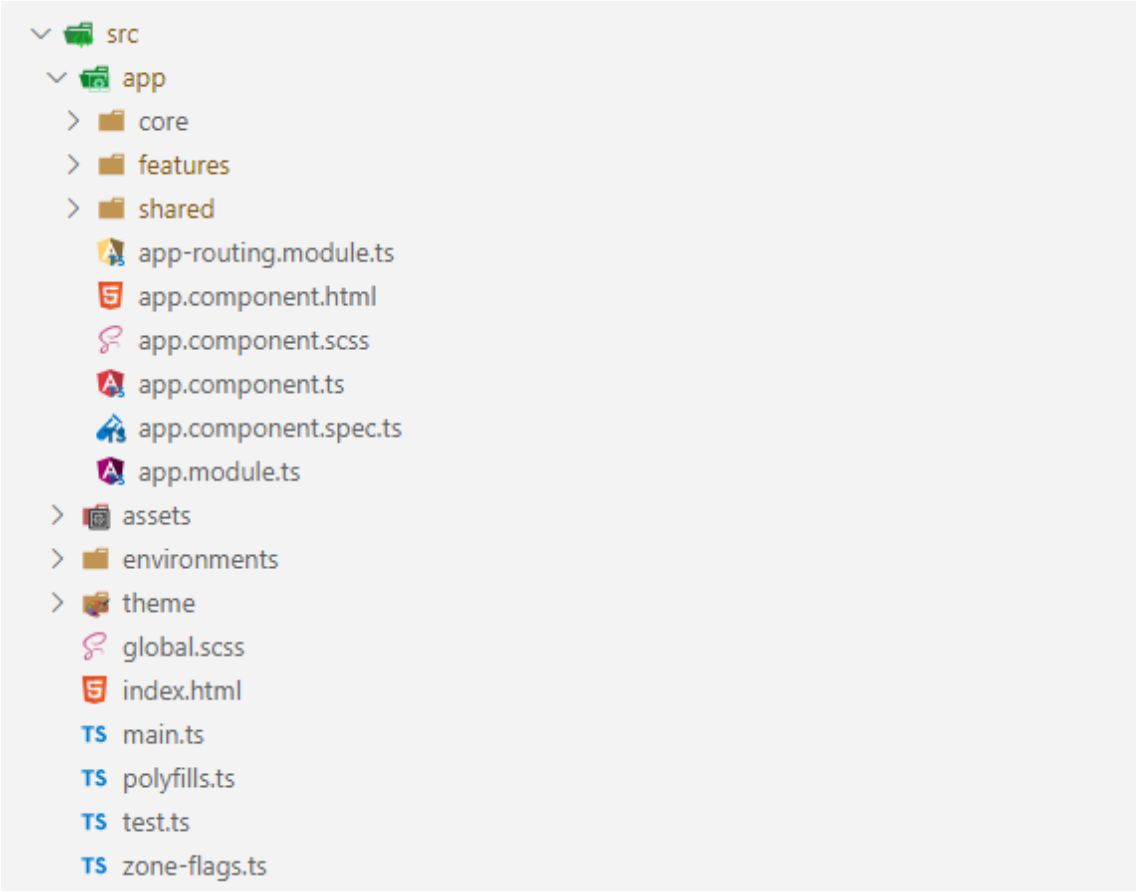


ILUSTRACIÓN 14: ESTRUCTURA DEL PROYECTO.

A continuación, vamos a detallar las carpetas y ficheros del directorio Src que se consideran más importantes y que se aprecian en la ilustración 14.

- **App**: Esta carpeta tendrá todos los ficheros de código, componentes, servicios, modelos, estilos, etc. Todos los cambios de funcionamiento o visuales tendrán lugar en esta carpeta.

- **App-routing.module.ts**: Este archivo será el encargado de determinar la vista a mostrar en cada dirección.

- **App.component.ts**: Es el primer archivo en cargar. En este archivo se suelen definir las partes que interesa tener al inicio de la aplicación.

- **Assets**: Esta carpeta contiene los recursos visuales como imágenes o iconos. También suele almacenar los ficheros de traducciones.

- **Environments**: Aquí se incluyen los ficheros de variables de configuración global de la aplicación.

- **Theme**: Fichero de estilos globales.

Como hemos comentado en el directorio App es donde nosotros añadimos y editamos gran parte de los archivos que componen el proyecto.

La parte fundamental de nuestro código, la hemos organizado en tres carpetas principales**:** Core, Features y Shared.

4.2.1. CORE

En primer lugar, encontramos el módulo core, donde encontramos los elementos (servicios y lógica) transversal a toda la aplicación. Por ese motivo es aquí donde se han implementado los servicios de la aplicación, y se han dividido en dos directorios, services y store, dentro de los cuales tenemos los siguientes servicios:

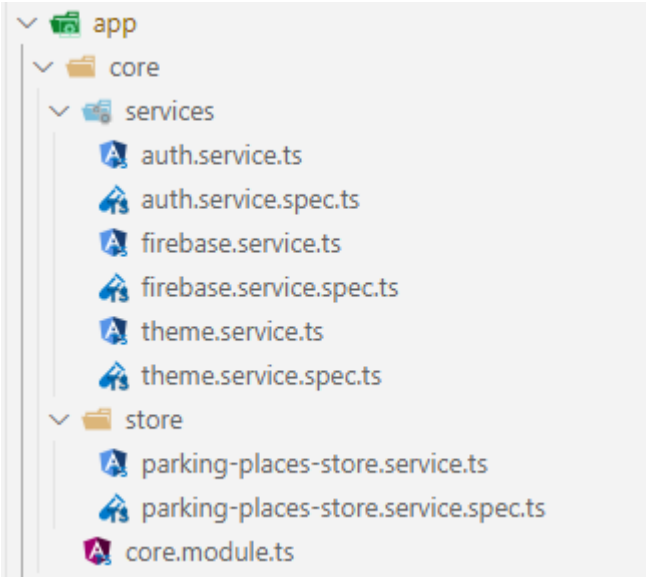


ILUSTRACIÓN 15: ESTRUCTURA CORE.

En el directorio services están implementados los servicios auth, firebase y theme:

El servicio auth como su nombre nos referencia, nos sirve para autenticar usuarios, en él se implementan las funciones encargadas de proceder a iniciar sesión en la aplicación y a registrar nuevos usuarios en ella.

En la plataforma Firebase hemos definido que la autenticación de usuarios sea mediante correo electrónico y contraseña, por lo que estas dos funciones implementadas tratan estas dos variables.

Estas dos funciones serán llamadas en sus respectivos módulos, login y register, para proceder con las acciones de inicio de sesión o registro de usuarios.

En el servicio firebase se han centralizado todas las operaciones que se hacen contra la base de datos.

En él tenemos las funciones encargadas de:

- Añadir usuarios: nombrada **addEmployee**.
- Añadir plazas de aparcamiento, **addParkingPlace**.
- Añadir clientes, **addClients**.
- Añadir habitaciones, **addRooms**.
- Obtener las plazas de aparcamiento, **getParkingPlaces**.

El servicio theme se ha implementado para poder seleccionar y cambiar el tema visual de la aplicación, este servicio nos permite almacenar en la memoria local de la aplicación una variable la cual nos indica si el tema seleccionado es un tema claro u oscuro.

El servicio parking-places-store, se ha creado para implementar un patrón de diseño global. En este patrón pretendemos centralizar y quitar llamadas innecesarias a la base de datos.

4.2.2. FEATURES

En el módulo de features, se han implementado las funcionalidades de la aplicación:

- **Login**: Primera pantalla donde iniciar sesión.
- **Register**: Pantalla donde se rellena el formulario de registro a la aplicación.
- **Menu**: Desplegable lateral.

- **Tabs**: Fila inferior donde poder desplazarnos por las diferentes páginas, Map y List.
- **Map**: Pantalla principal donde tenemos el mapa con la distribución de plazas.
- **List**: Listado de plazas de aparcamiento.
- **Settings**: Pantalla de opciones.

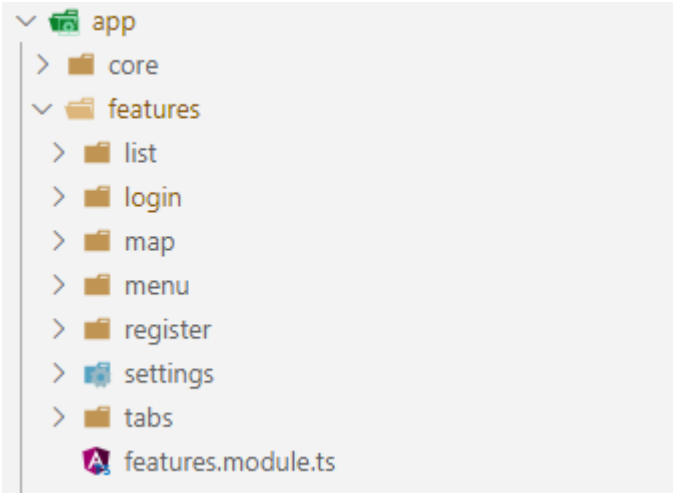


ILUSTRACIÓN 16: ESTRUCTURA FEATURES.

El código implementado en la funcionalidad login tendrá principalmente dos funciones, una función la cual presentará un formulario donde requerirá dos campos para poder iniciar sesión, los campos pedidos serán email y contraseña, y una segunda función que se encargará de comprobar, mediante el servicio antes visto auth, los datos proporcionados, si estos son correctos se presentara la pagina principal de la aplicación.

El módulo register es similar al de login, en el tenemos un formulario con los campos definidos en el apartado 3.2. para un usuario y una función que validará tanto que los campos introducidos tienen el formato correcto como también comprobar que la contraseña introducida dos veces sea idéntica.

En la funcionalidad menu se ha implementado un menú con un despliegue horizontal donde podemos navegar a la pantalla de ajustes o cerrar sesión de la aplicación. También se ha implementado en el diagrama circular donde tenemos la información de la ocupación del aparcamiento.

La funcionalidad tabs es solo el menú inferior donde están definidas las rutas a las diferentes pestañas de la aplicación; mapa o lista.

El módulo map es donde se ha implementado una reproducción del esquema del aparcamiento, para ello se ha hecho uso de una matriz donde se indican las posiciones de las plazas, cada posición aparece un botón el cual variará su color dependiendo del estado de la plaza, este botón nos permitirá editar los atributos que tiene cada plaza.

La funcionalidad list está creada para tener una lista de las plazas del *parking* el uso de esta pestaña nos proporcionará un punto de vista diferente la pestaña map siendo igual su uso. Pulsando sobre un elemento de la lista podremos editar los atributos de la plaza. También se ha implementado un buscador para facilitar el filtrado de campos, por ejemplo, escribiendo A01, nos aparecerá únicamente la plaza con identificador A01, también podremos filtrar por cliente o por estado de la plaza.

Estas dos funcionalidades, map y list, comparten un mismo botón el cual permitirá añadir atributos a una plaza desde cero.

El último módulo, settings, es una pantalla de ajustes donde se ha implementado el cambio de idioma, se podrá elegir entre inglés o español, tema visual de la aplicación, oscuro o claro para poder ajustarse a la luz y consumo de energía, o cambiar la contraseña del usuario.

4.2.3. SHARED

En el último módulo creado, Shared, hemos implementado los elementos que se comparte entre los diferentes componentes de la aplicación.

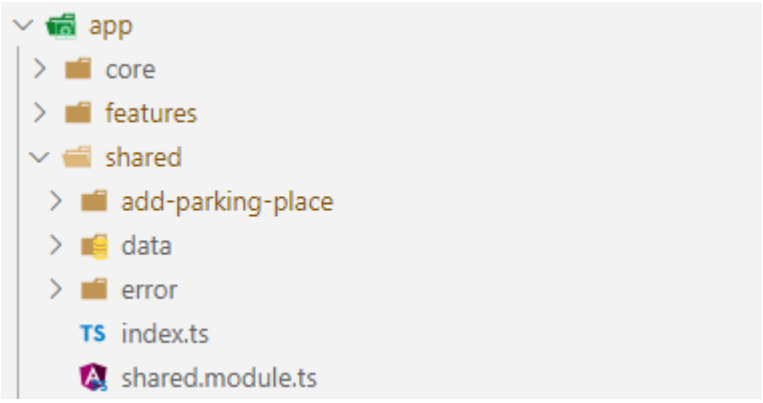


ILUSTRACIÓN 17: ESTRUCTURA SHARED.

En este modulo hemos creado tres funcionalidades, add-parking-place, data y error. Hemos creado dos componentes y un directorio donde tenemos representadas las interfaces que definen los tipos de objetos de nuestra aplicación.

La primera funcionalidad, add-parking-place, es un *pop-up*, en Ionic denominado elemento modal. Este elemento es accesible tanto desde la pestaña Map como de la de List, está creado haciendo uso de un formulario donde tenemos definidos los atributos que hacemos uso en las plazas de aparcamiento. A este componente se le ha

implementado dos formas de uso diferentes siendo un solo elemento, tiene la opción de abrir el formulario completamente vacío para que un usuario pueda añadir todas las propiedades de la plaza de aparcamiento y así crear una reserva desde cero desde el botón de añadir o bien, abrir el modal seleccionando una plaza especifica tanto en el mapa como en la lista, de esta manera el formulario se abrirá con los atributos obtenidos desde la base de datos para esa plaza en específico. En esta segunda opción, el formulario se abrirá en modo solo visualización, siendo posible editar los campos previa solicitud, consiguiendo así poder consultar los detalles de una plaza sin peligro a cambiarlos involuntariamente.

Dentro del directorio data, están definidas las entidades de la aplicación definidas en el punto 3.2. Diseño de la base de datos. Estas nos sirven a la hora de crear un objeto, en este módulo ya están creadas con sus atributos y así cuando hacemos uso de por ejemplo ParkingPlace, en algún componente de la aplicación, este componente ya sabe los atributos definidos en este objeto.

El módulo error, como su nombre indica, es donde hemos centralizado los mensajes de error en la aplicación. Por ejemplo, al crear un usuario, si intentásemos insertar un número de teléfono que no contuviese únicamente dígitos, en la aplicación nos aparecería un mensaje en ese campo indicando el tipo de error.

4.3. PROBLEMAS Y SOLUCIONES

El principal problema que surgió durante el desarrollo del proyecto fue la desaparición de datos al ser refrescada una página y no cargar correctamente elementos visuales como el gráfico donde se representaba la ocupación del *parking*.

Esto fue debido al diseño de las aplicaciones Ionic, las cuales están estructuradas jerárquicamente, un componente padre puede tener N componentes hijos y esto implica que la forma de comunicarse los componentes entre sí determina el comportamiento de la aplicación.

Gracias a esto también se observó que tanto las pantallas Mapa y Lista como el menú desplegable realizaban las mismas llamadas a la base de datos para la presentación de las plazas de aparcamiento y sus atributos, lo que suponía tener código triplicado y una carga en el servidor innecesaria.

Esto se arreglo implementando el servicio antes descrito, parking-places-store, en él se implementa un patrón de estado global. Este patrón nos permite reutilizar código de manera, que un componente, solo tiene que apoyarse en el servicio que lo implementa, accediendo a su variable compartida mediante el uso de los observables (patrón publicador-subscriptor) de manera que cada vez que se produce un cambio, se propaga en todos aquellos componentes suscritos a este. De esta manera se delega la petición a

la base de datos en este servicio y el resto de los componentes interesados solo deben suscribirse para tener la información actualizada.

5. INSTALACIÓN Y MANUAL DE USUARIO

La intención de este quinto punto de la memoria es tratar el uso de la aplicación tanto para un perfil técnico como para un perfil de usuario. En el primer caso trataremos de explicar detalladamente como dejar preparada toda la aplicación hasta el punto de poder implementarla en un escenario real. En el segundo caso, se detallará un manual de uso para que todo usuario tenga un primer contacto visual con la aplicación y sus funcionalidades.

5.1. INSTALACIÓN Y MANTENIMIENTO

Para empezar a trabajar en el proyecto, es necesario tener instalados una serie de componentes.

En primer lugar debemos descargar e instalar Node.js (7), Node.js es un entorno de ejecución para JavaScript, podemos acceder y descargar esta aplicación desde su propia página web, https://nodejs.org/es/.

Una vez instalado Node.js, tenemos que instalar Ionic, para ello podemos seguir los pasos detallados en https://ionicframework.com/getting-started/, escribiendo en la línea de comandos la siguiente instrucción se procederá a la instalación:

npm install -g @ionic/cli

Una vez en este punto, ya podremos trabajar con el código de la aplicación desarrollada, y el primer paso será configurar la integración con la base de datos en la que se va a trabajar. Para ello es necesario tener creada una cuenta de Goolge y acceder a Firebase, https://firebase.google.com/, crearemos un proyecto siguiendo los pasos que se nos indica y una vez dentro de nuestro proyecto, en el menú lateral, en el apartado Autenticación, elegiremos el método de inicio de sesión para nuestra aplicación, el cual será el implementado con correo electrónico y contraseña.

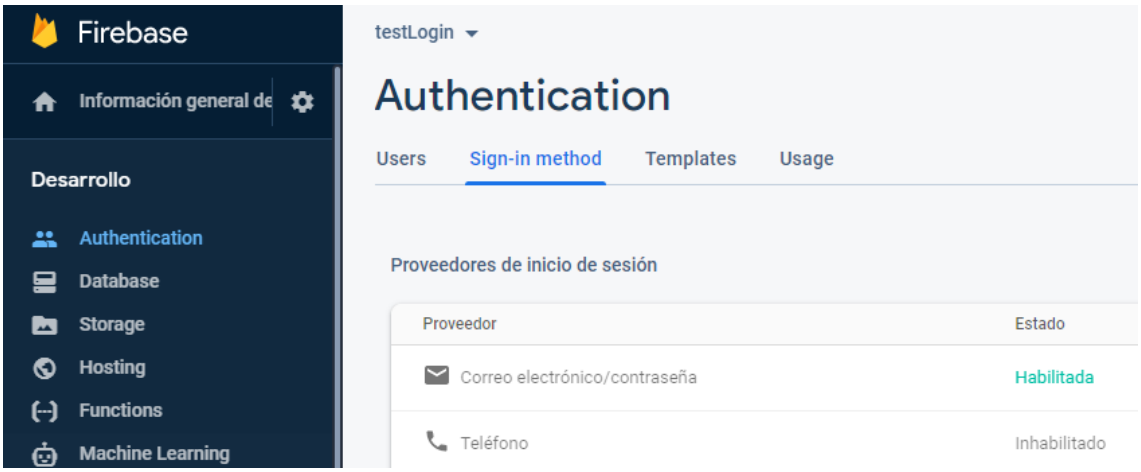


ILUSTRACIÓN 18: MENÚ FIREBASE, AUTENTICACIÓN.

El próximo paso, como hemos dicho es la integración del proyecto con la base de datos, para esto, accediendo a Ajustes tenemos un apartado nombrado Firebase SDK snippet, en él está el código que debemos añadir a nuestra aplicación para enlazarla.

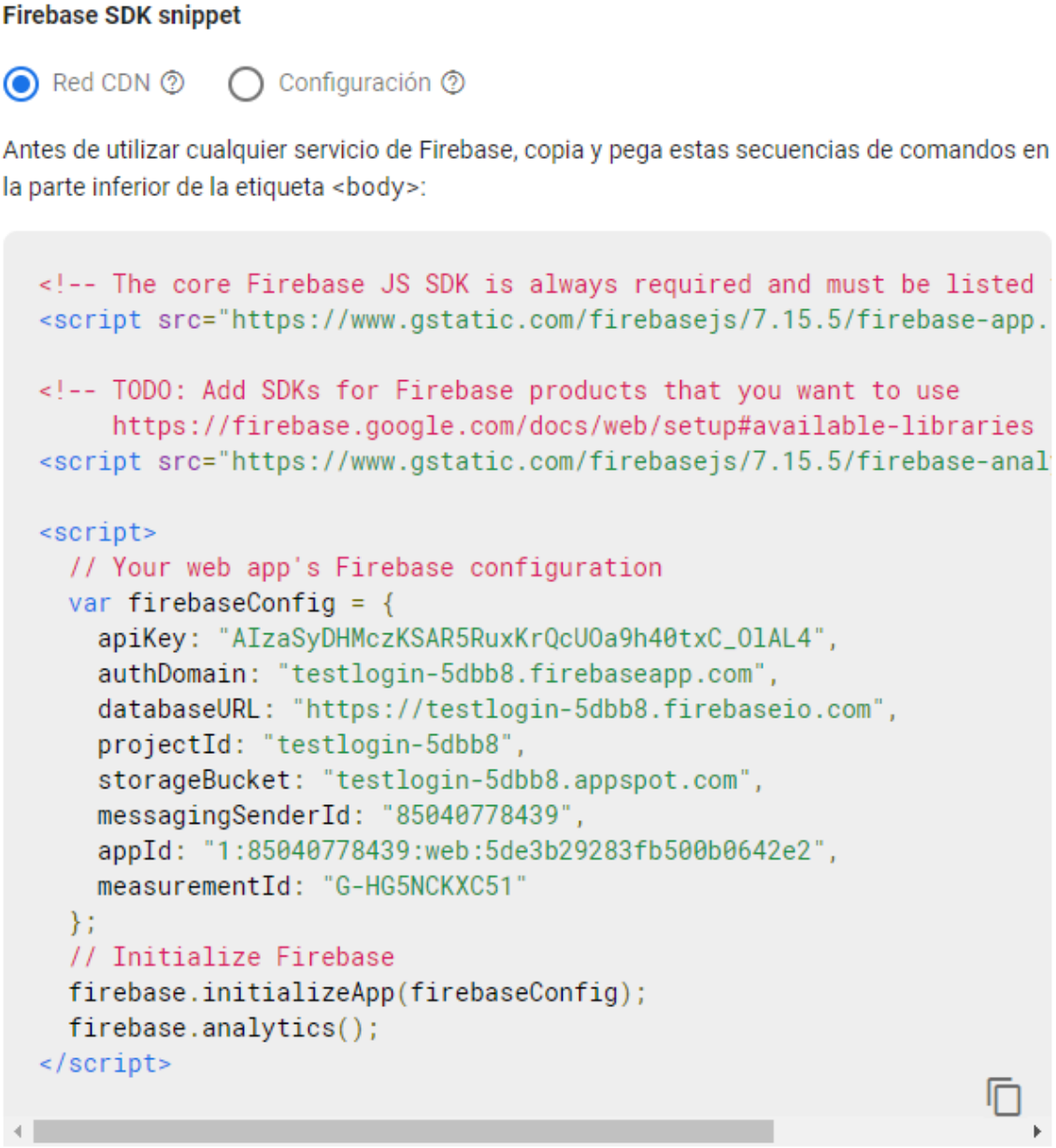


ILUSTRACIÓN 19: FIREBASE SDK SNIPPET

Este código debemos copiar y pegarlo dentro de la carpeta Environments, en el archivo environments.ts.

La creación de las colecciones en la base de datos se realiza de manera automática al ejecutar la aplicación. Firebase nos proporciona las herramientas para visualizar y editar

documentos de las colecciones, también proporciona una herramienta para poder hacer búsquedas simples, un ejemplo de esto es el siguiente, donde se muestra como buscamos en la colección Employees, los empleados donde el campo *name* sea Joan:

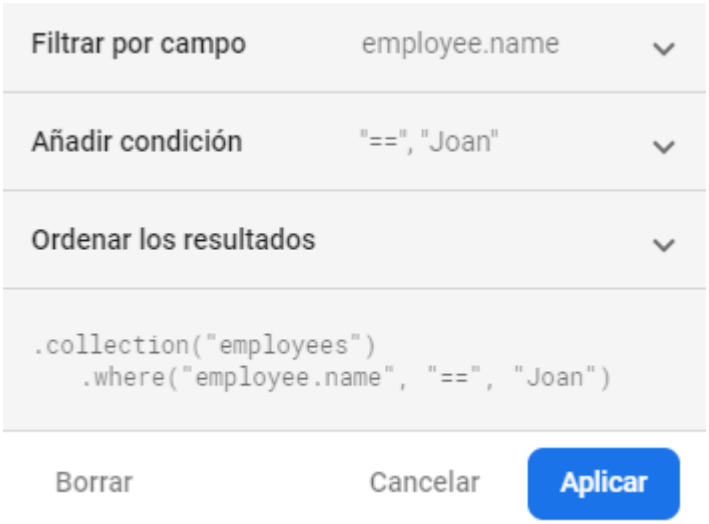


ILUSTRACIÓN 20: BÚSQUEDAS SIMPLES FIREBASE.

Para implementar el proyecto en iOS y Android seguiremos la guía que nos proporciona Ionic, https://ionicframework.com/docs/react/your-first-app/6-deploying-mobile y para acceder desde un navegador a la aplicación usaremos el servicio de hosting que nos proporciona Firebase, para ello hemos seguido el tutorial escrito por Josh Morony (8), en el menú lateral que aparece en la ilustración 18, tenemos la opción Hosting, la primera vez que se accede se indica paso por paso como implementar la aplicación.

En primer lugar, es necesario instalar las herramientas de Firebase, para ello escribiremos en nuestro terminal de comandos:

npm install -g firebase-tools

Una vez instalado podremos iniciar sesión e inicializar el proyecto con los comandos:

Firebase login y **Firebase init**

En el terminal aparecerán varias funcionalidades a elegir, marcaremos Hosting y aceptaremos. El próximo paso será escribir:

npm run build y **firebase deploy**

Con esto Firebase nos proporcionará una URL donde tendremos nuestra aplicación.

5.2. USO DE LA APLICACIÓN

Una vez explicado la instalación de la aplicación, en este punto vamos a presentar una guía para que un usuario nuevo pueda operar con facilidad permitiendo ver las funcionalidades con capturas reales de la aplicación.

En primer lugar y una vez instalada la aplicación, hay que proceder al registro de usuarios, para ello, en la parte inferior de la página de inicio hay un botón con el texto, *¡Create new account!*, pulsando sobre el iremos a la pantalla de registro, donde rellenando correctamente los campos que se nos piden se realizará el registro de usuario y ya podremos iniciar sesión en la aplicación.

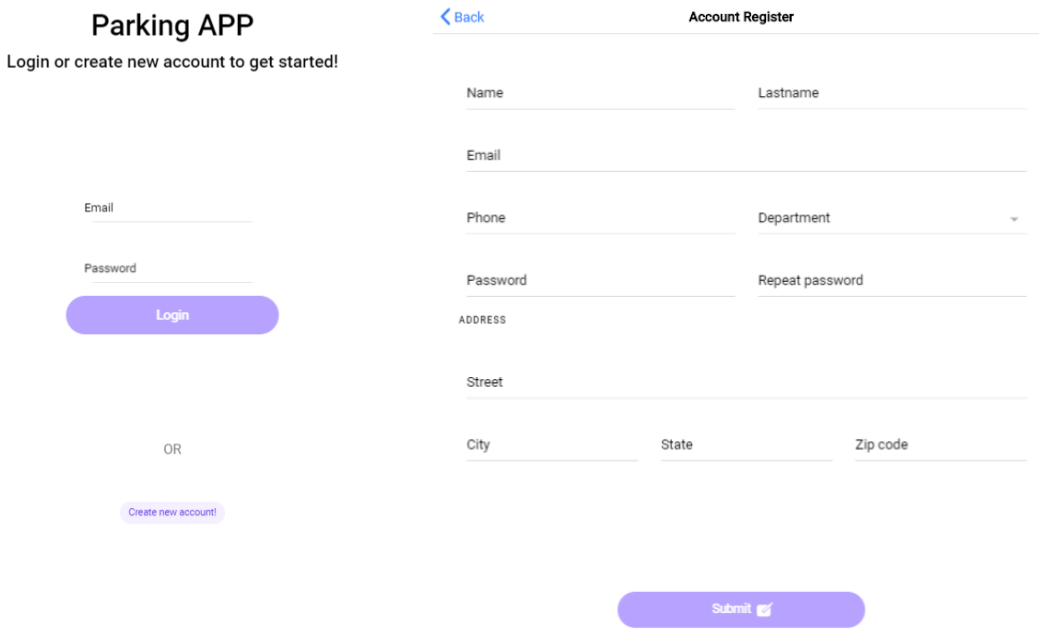


ILUSTRACIÓN 21: INICIO DE SESIÓN Y REGISTRO DE USUARIO.

Una vez realizado el inicio de sesión, la pantalla principal que nos aparece será el mapa, con su distribución y el estado de las plazas. En caso de ser la primera vez que se va a usar la aplicación el estado de ellas sería vacía y las visualizaríamos de color verde, indicando que se pueden usar. Los estados que pueden tener las plazas son: vacía, ocupada, reservada y fuera de servicio, con los colores verde, rojo, azul y gris respectivamente.

En la parte inferior tenemos un menú donde podemos elegir la pestaña, desde este menú accederemos a la segunda pantalla principal de la aplicación, Lista, donde podemos ver las plazas de aparcamiento ordenadas por identificador de la plaza.

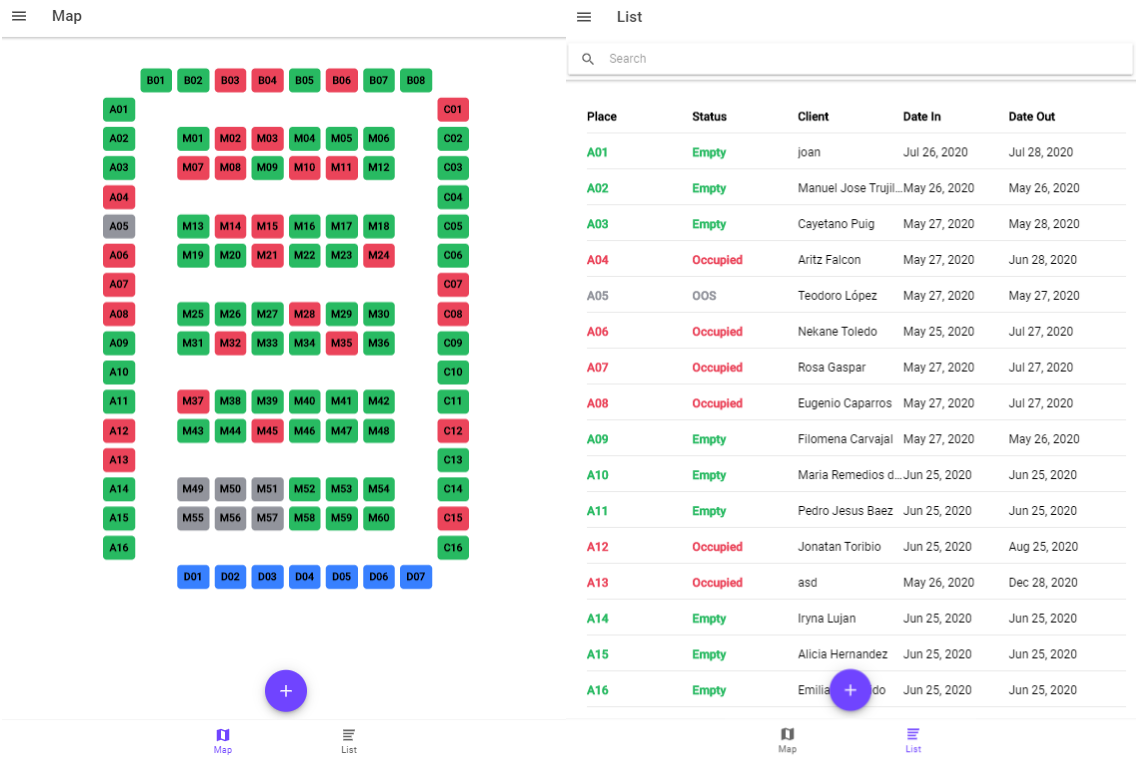


ILUSTRACIÓN 22: PANTALLAS PRINCIPALES, MAPA Y LISTA.

Usando el buscador, situado en la parte superior de la pagina Lista, podemos filtrar la lista usando los campos Plaza, Estado o Cliente.

Pulsando sobre una plaza se abrirá un formulario con los datos referentes a esa plaza y se podrán editar.

Como podemos observar en la parte inferior central de las dos pantallas principales, hay un botón el cual desplegará el formulario antes comentado, pero con los campos vacíos, permitiendo crear una reserva desde cero. La siguiente ilustración muestra este elemento en sus dos opciones de uso.

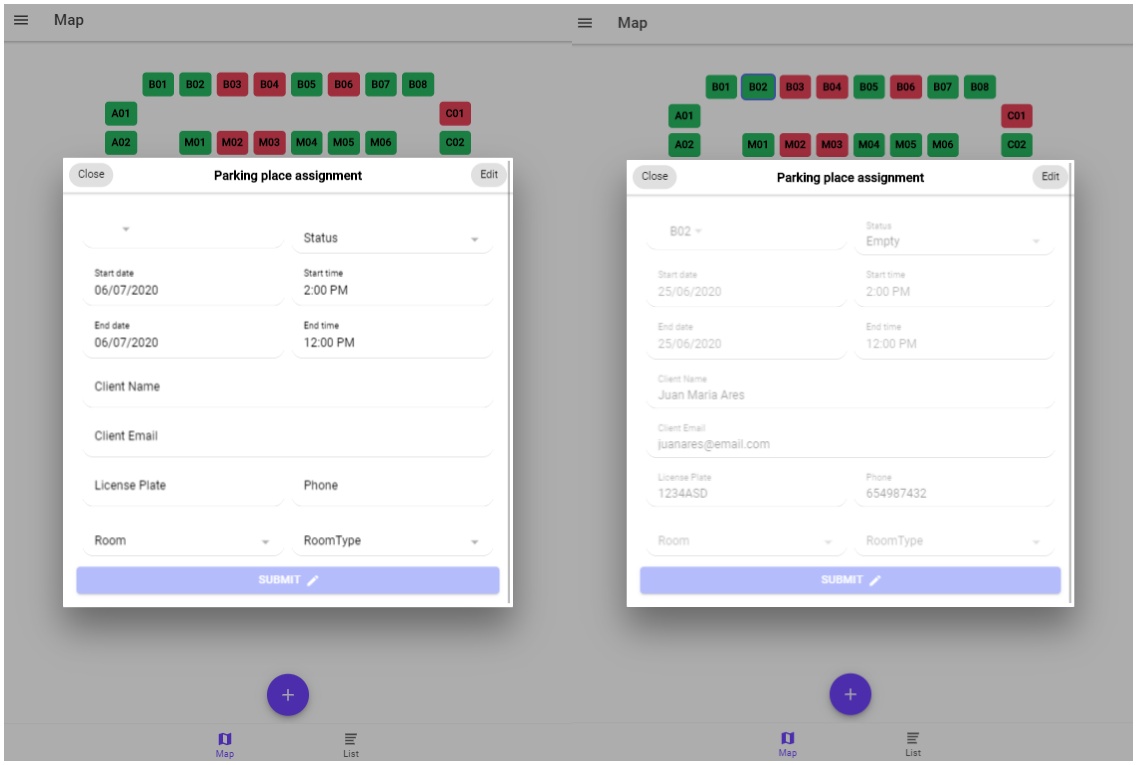


ILUSTRACIÓN 23: POP-UP ATRIBUTOS PLAZAS.

Podemos observar una pequeña diferencia entre los dos *pop-ups*, el de la izquierda esta vacío y listo para crear una reserva, mientras que el de la derecha esta bloqueado para su edición, evitando modificar algún dato innecesariamente, para su edición en la esquina superior derecha hay un botón, Edit, el cual nos permitirá editar los campos que necesitemos.

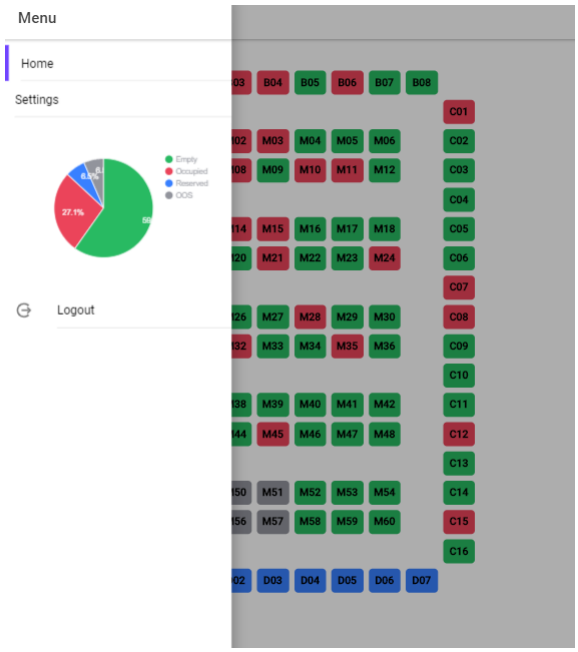


ILUSTRACIÓN 24: MENÚ DESPLEGABLE.

En la esquina superior izquierda, tenemos la opción de desplegar el menú, tal como vemos en la ilustración 21, esta funcionalidad nos permite navegar entre las diferentes pantallas que tiene la aplicación, visualizar un gráfico de ocupación o cerrar sesión en la aplicación.

Pulsando sobre Settings en el menú desplegable, abrimos una pantalla de ajustes, donde tenemos la opción de escoger un estilo visual nocturno o cambiar el idioma de la aplicación, por defecto el tema visual será claro y el idioma de la aplicación inglés.

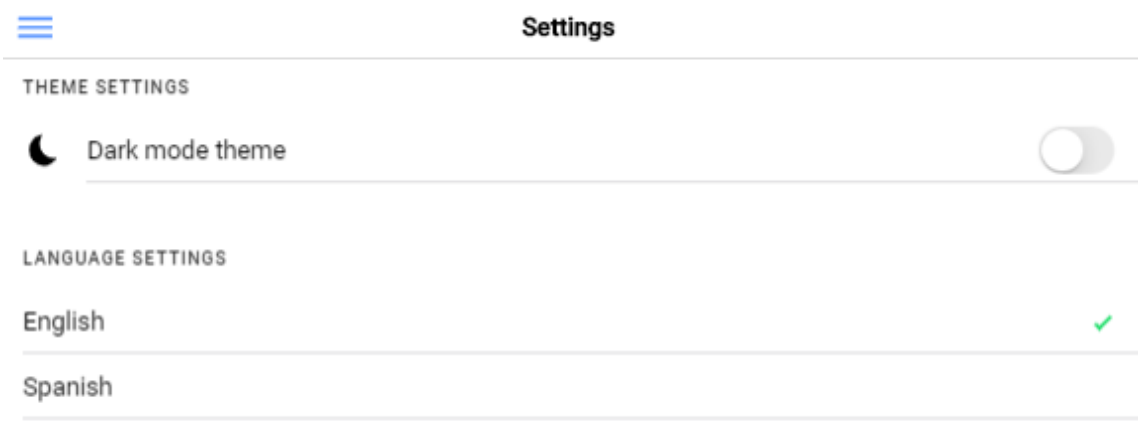


ILUSTRACIÓN 25: PANTALLA AJUSTES

6. CONCLUSIONES Y TRABAJO FUTURO

En proyecto hemos desarrollado un programa para facilitar la gestión del aparcamiento en el Hotel Oliva Nova Beach & Golf Resort, para ello hemos analizado unos requerimientos que nos exigía el cliente, permitiendo diseñar e implementar la aplicación.

Con la realización de este proyecto hemos cumplido con los objetivos que nos propusimos a la hora de decidir realizar la aplicación, con ella hemos conseguido mejorar la eficiencia en el departamento de recepción, actualizado la metodología con la que se trabajaba y conseguir reducir en una gran cantidad la huella ecológica respecto al antiguo modelo de trabajo.

Podemos concluir que el desarrollo de la aplicación ha sido satisfactorio, sin embargo, esta tiene un porcentaje de mejora muy grande, pese a ser una aplicación de apariencia sencilla su código se puede optimizar aumentando así su rendimiento y apariencia.

La realización del proyecto me ha permitido familiarizarme con las tecnologías utilizadas especialmente en el *framework* de Ionic, ampliando mis conocimientos en programación, el cual era uno de los objetivos principales a la hora de elegir este proyecto ya que mi ámbito laboral hasta la fecha se había inclinado hacia las telecomunicaciones y no hacia la programación.

Como trabajo futuro, queda trabajar en dos funcionalidades que se descartaron para ser implementadas en una primera parte de la aplicación:

- Creación y autoedición en la disposición de las plazas de aparcamiento.
- Asignación de atributo matricula mediante la cama de un dispositivo móvil.

También, como he comentado, optimizar su rendimiento y apariencia y para ello será necesario de una mayor formación en los *frameworks* Ionic y Angular.

7. BIBLIOGRAFÍA

1. JHipster Domain Language [Internet]. [citado 6 de julio de 2020]. Disponible en: https://www.jhipster.tech/jdl/

2. Devs Q. ¿Qué es Angular y para qué sirve? [Internet]. Quality Devs. 2019 [citado 4 de julio de 2020]. Disponible en: https://www.qualitydevs.com/2019/09/16/que-es-angular-y-para-que-sirve/

3. ✔ Qué es Ionic | Quality Devs | Somos Desarrolladores [Internet]. Quality Devs. 2019 [citado 6 de julio de 2020]. Disponible en: https://www.qualitydevs.com/2019/05/31/que-es-ionic-desarrollador-web/

4. Qué es Firebase: funcionalidades, ventajas y conclusiones [Internet]. DIGITAL55. 2020 [citado 6 de julio de 2020]. Disponible en: https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/

5. Mahi! Cloud Firestore introduction [Internet]. Medium. 2019 [citado 6 de julio de 2020]. Disponible en: https://medium.com/feedflood/cloud-firestore-introduction-a9efe2c8d0c

6. Visual Studio Code. En: Wikipedia, la enciclopedia libre [Internet]. 2020 [citado 6 de julio de 2020]. Disponible en: https://es.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=127378798

7. Qué es NodeJS y para qué sirve [Internet]. OpenWebinars.net. 2019 [citado 7 de julio de 2020]. Disponible en: https://openwebinars.net/blog/que-es-nodejs/

8. Hosting an Ionic PWA with Firebase Hosting [Internet]. joshmorony - Learn Ionic & Build Mobile Apps with Web Tech. 2017 [citado 7 de julio de 2020]. Disponible en: https://www.joshmorony.com/hosting-an-ionic-pwa-with-firebase-hosting/

DECLARACIÓN DE TRABAJO ORIGINAL

Yo, Joan Tomás Martínez, declaro que soy el autor de este trabajo y que no ha sido copiado o hecho por otras personas.

Añadir que las ilustraciones que se muestran en la memoria son todas de creación propia.