UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de APP multiplataforma para compartir información entre usuarios en tiempo real

Javier Martínez Hernández
Tutor: Juan José Sánchez Peña
Ponente: Daniel Hernández Lobato

Junio 2018

Desarrollo de APP multiplataforma para compartir información entre usuarios en tiempo real

AUTOR: Javier Martínez Hernández
TUTOR: Juan José Sánchez Peña

Escuela Politécnica Superior
Universidad Autónoma de Madrid

Junio de 2018

Resumen (castellano)

En los últimos años el uso de los teléfonos inteligentes o smartphones aumentó de manera significativa gracias a que nos facilitan conectarnos a internet desde casi cualquier lugar, permitiéndonos compartir información y/o nuestras opiniones con gran facilidad.

En España el 'tapeo' forma parte de la cultura popular desde hace muchos años, posibilitándonos interaccionar socialmente y descubrir nuevos lugares afines a nuestros gustos y/o expectativas, siendo esta actividad una manera divertida de conocer zonas destacadas de otras localidades y en muchas ocasiones conocer productos gastronómicos de la zona.

Para ayudar a mantener esta parte de la cultura y descubrir nuevos lugares donde disfrutar de un buen trato hemos desarrollado la siguiente aplicación multiplataforma que nos dará información en tiempo real, trabajando directamente con la nube, sobre lugares donde podremos interaccionar socialmente conociendo de antemano información variada. La información que nos proporcionará de cada sitio mostrará los datos de ubicación del local, una galería de fotos subidas por los usuarios, los comentarios de estos y una puntuación general.

Inicialmente, gracias al uso de la ubicación gps nos mostrará lugares cerca de nuestra ubicación, aunque esto no será una limitación para ver sitios de otras localidades gracias a dos extensibles que nos permitirán filtrar por provincias y localidades. Además de esta información, nos permitirá añadir nuevos sitios a la aplicación, haciendo uso de la localización gps y el uso de algunas APIs de Google, mediante un proceso semi-automático usando información proporcionada por Google. En caso de no encontrar ningún local registrado en nuestra aplicación próximo a la ubicación actual nos mostrará un listado de posibles lugares que podamos añadir a nuestra aplicación y nos dará información sobre ellos sin tener que añadirlo. Todo esto se mostrará con una interfaz sencilla e intuitiva.

Abstract (English)

In recent years the use of smartphones has increased significantly due to it allow us to get connected to Internet from almost any place, it allows us to share information and/or our opinions with great ease.

In Spain the "tapeo" is part of the popular culture for many years, enabling to us to interact socially and to discover new places related to our preferences and/or expectations, being this activity an entertaining way of knowing highlight areas from other localities and in many cases to know gastronomic products of the area.

To help to maintain this part of the culture and discover new places where enjoy a good treatment, we have developed the following multiplatform application that will give us information in real-time, working directly with the cloud about places where we will interact socially knowing in advance varied information. The information provided by each place will show the location data, a gallery of photos uploaded by users, the comments of these and a general rating.

Initially, through using of the GPS location it will show us places close to our placement, although this will not be a limitation to see places from other localities thanks to two extensible ones that allow us filter provinces and localities. In addition to this information, it will allow us for adding new places to the application, making use of the GPS location and the use of some APIs of Google by means of a semiautomatic process using information provided by Google. In case of not finding any registered place next to the current place in our application, it will display us a list of possible places that we could add to our application and it will give us information about them without adding. All this will be showed with a simple and intuitive interface.

Palabras clave (castellano)

Smartphone, Aplicación móvil, Multiplataforma.

Keywords (inglés)

Smartphone, Mobile application, Multiplatform.

Agradecimientos

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

1 Introducción

1.1 Motivación

Hoy en día vivimos rodeados de tecnología y en la actualidad el uso de smartphones se ha vuelto algo cotidiano. Según los estudios realizados por 'Google Consumer Barometer'[1] en los últimos 5 años el uso de estos dispositivos aumentó en más de un 40%, mostrando un porcentaje de uso de la población superior al 70% en muchos países, entre ellos España, estando cerca del 85%. Esto nos muestra la importancia del área de desarrollo para aplicaciones y nos da una idea de su progresión en el futuro.

Como consecuencia del incremento de uso de los dispositivos móviles se ha producido un aumento del uso y desarrollo de Apps. Estas Apps no siempre están dirigidas a todos los tipos de dispositivos ya que esto depende del tipo de App y su estilo de desarrollo. Esos estilos podemos dividirlos en tres categorías [2]:

- PWA: estas aplicaciones son accedidas a través de la web desde un navegador.
  Soportadas por la mayor parte de los navegadores del mercado, siendo plenamente indexables (la accesibilidad y transparencia que ofrece) por los buscadores, facilitando la búsqueda y el descubrimiento de nuevas Apps. No tenemos necesidad de descargarlas en nuestros dispositivos.

- Aplicaciones híbridas: estas aplicaciones nos permiten desarrollar un único código fuente para múltiples plataformas reduciendo notablemente los recursos tanto de desarrollo como de mantenimiento reduciendo el 'time-to-market', es decir, tiempo que tarda un producto desde que es concebido hasta que está a la venta.

- Aplicaciones nativas: estas aplicaciones se desarrollan específicamente para cada sistema operativo adaptando cada uno al lenguaje en el cual se desarrollan.

Cada uno de estos estilos de desarrollo tiene sus propias características y está orientado a aprovechar en mayor o menor medida la funcionalidad de los dispositivos para ofrecer distintas experiencias a los usuarios en función del tipo de aplicación que se desee desarrollar y a qué público queramos llegar.

Por otra parte, también se ha producido un incremento de uso de las bases de datos no relacionales (NoSQL) para el desarrollo de las Apps gracias a las características que nos ofrecen tales como la variabilidad de los datos, velocidad y volumen de la información, permitiéndonos almacenar campos distintos en cada documento y su gran escalabilidad.

1.2 Objetivos

Los objetivos principales de este proyecto han sido:

- Adquirir una buena base de conocimientos tanto para el desarrollo Frontend como Backend.
- Mejorar las buenas prácticas de programación.
- Aprender a tomar buenas decisiones sobre el tipo de base de datos que debemos usar en función de los requisitos de nuestra aplicación.
- Manejo de datos en bases de datos no relacionales, NoSQL.
- Interacción con las distintas APIs (**Application Programming Interface**) que se pueden encontrar en el mercado y sus limitaciones.

Todos estos objetivos se usarán para desarrollar una aplicación híbrida para Android e IOS que permita encontrar y/o planificar distintos lugares donde poder interaccionar socialmente mediante el 'tapeo' ya sea en la ubicación actual como en otros lugares que se tenga planificado visitar.

1.3 Organización de la memoria

Este documento se divide en 8 apartados, donde se incluye este capítulo. En él se realiza una introducción al proyecto.

El siguiente capítulo llamado "Estado del Arte", coloca el sistema desarrollado en un contexto explicando en qué se diferencia de su competencia.

A continuación, el capítulo "Objetivos y funcionalidades" detalla las principales funcionalidades detectadas para nuestra aplicación y los objetivos que se esperan conseguir.

Después el capítulo titulado "Herramientas y tecnologías" describe las herramientas y tecnologías utilizadas para el desarrollo. Por otra parte, el capítulo "Metodología y análisis" explica el uso de las metodologías utilizadas y un análisis de requisitos y tareas realizado para la aplicación.

A continuación, tenemos el capítulo titulado "Diseño e implementación" donde se desarrollan el diseño y la implementación del proyecto detallando la arquitectura utilizada y sus componentes, el diseño que se ha seguido para la base de datos, la navegación, los diseños conceptuales de las maquetas...

Después, en el capítulo "Pruebas y resultados" se realiza un análisis de las pruebas realizadas sobre la aplicación, detallando los resultados y las conclusiones obtenidas de las mismas y para finalizar tenemos como punto final el capítulo "Conclusiones y trabajo futuro" donde se exponen las diferentes conclusiones derivadas de la realización del proyecto. Además, se discuten las líneas de trabajo futuro que ofrece este proyecto.

2 Estado del arte

Las tecnologías avanzan a un gran ritmo teniendo como objetivo facilitar la vida cotidiana de las personas. Gracias al uso de los sensores de los dispositivos se han desarrollado muchas aplicaciones que nos ayudan en el día a día y eso es lo que haremos en nuestra aplicación.

Antes de comenzar con la realización de este proyecto se realizó una búsqueda de aplicaciones existentes que estuviesen orientadas a nuestro objetivo, encontrar buenos lugares donde interaccionar socialmente y disfrutar del trato, compañía y calidad del establecimiento, pudiendo realizar nosotros el registro de los locales que nos gustan y añadir comentarios, todo esto usando la geolocalización y evitando que el usuario tenga que introducir la información manualmente hasta donde sea posible.

2.1 De Tapeo

Esta aplicación solo está desarrollada para Android y nada más acceder nos muestra un listado de todas las provincias de la península independientemente de que exista algún local registrado para esa provincia.

Los locales registrados muestran únicamente la información de ubicación. No ofrece la posibilidad de realizar registros a los usuarios, deben ser los propios establecimientos los que se inserten de forma manual su información.

No hace ningún uso de la geolocalización y además tiene una gran cantidad de publicidad intrusiva la cual nos da una pésima experiencia de navegación.

[1]

Figura 2-1: Icono de la app "De Tapeo"

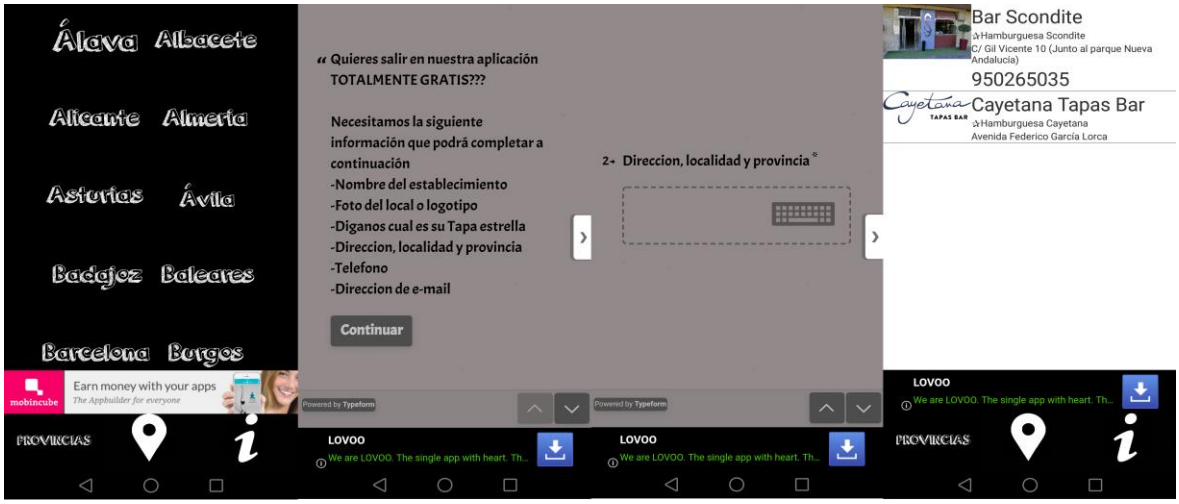


Figura 2-2: Aplicación "De Tapeo"

[1] https://play.google.com/store/apps/details?id=com.detapeo

2.2 Tapitas

Esta aplicación está desarrollada para Android e IOS y ofrece la posibilidad de buscar locales en toda la península con el inconveniente de no usar ningún filtro ya que se basa únicamente en la ubicación actual. Para ver los locales se debe deslizar el scroll y únicamente marca la distancia desde la ubicación actual sin indicar información de provincia y/o localidad. Esto imposibilita la capacidad de planear locales que visitar en un viaje.

Los locales registrados muestran información sobre la ubicación y las tapas de este. Ofrece la posibilidad de realizar registros de los locales y tapas por parte de los usuarios, en este proceso se realiza limitando al mínimo la necesidad de introducir información por parte de usuario haciendo uso de la geolocalización.

[2]

Figura 2-3: Icono de la app "Tapitas"

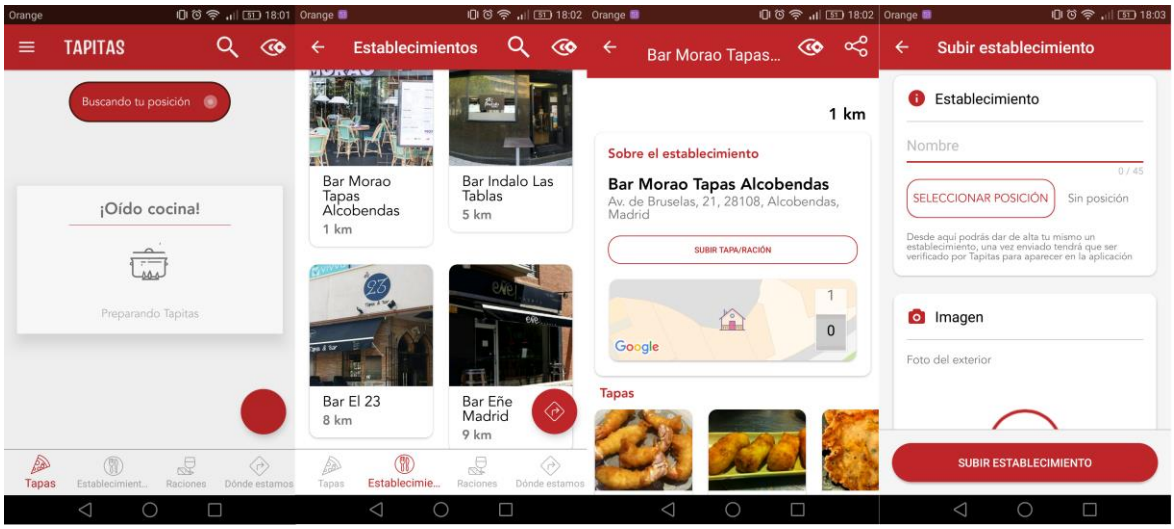


Figura 2-4: Aplicación "Tapitas"

[2] https://play.google.com/store/apps/details?id=com.tapitasapp.tapitas

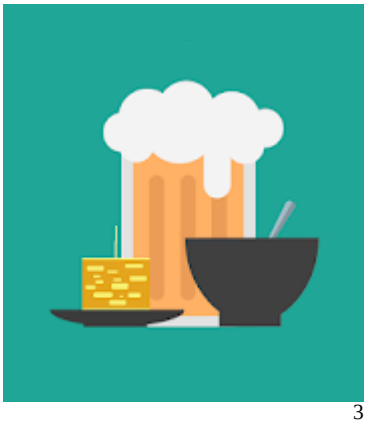
2.3 Tappear

Esta aplicación está desarrollada para Android e IOS e inicialmente solicita acceso a la ubicación mediante el uso del gps.

Está orientada solamente a Galicia, permite filtrar por provincias mostrando la información en un listado ordenado por distancia desde la ubicación actual.

Los locales registrados muestran información de ubicación sobre el local. Además, ofrece la posibilidad de realizar registros de nuevos locales siendo necesario para ello introducir la mayor parte de la información y la geolocalización para este registro no es precisa.

Está desarrollado con un paquete de idiomas: Castellano, Gallego e Inglés, el cual no funciona.



[3]

Figura 2-5: Icono de la app "Tappear"

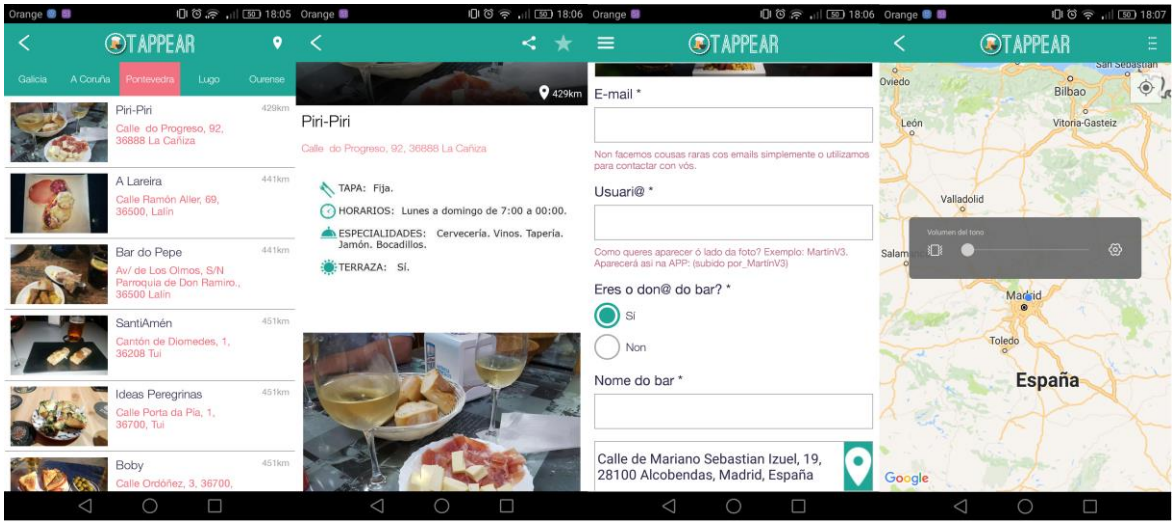


Figura 2-6: Aplicación "Tappear"

[3] https://play.google.com/store/apps/details?id=com.tappear

2.4 Tapeando: Tapas y pinchos

Esta aplicación está desarrollada para Android y ofrece la posibilidad de buscar locales en toda la península usando un filtro por provincia y zona. No usa la localización gps para cargar inicialmente información de lugares cercanos.

Los locales registrados muestran gran cantidad de información sobre el local, rutas y ferias de tapeo, sin embargo, te obliga a permitir el acceso a llamadas telefónicas para poder visualizar la información.

Además, no permite añadir nuevos locales como usuario, este debe ponerse en contacto con los responsables.

[4]

Figura 2-7: Icono de la app "Tapeando: Tapas y pinchos"

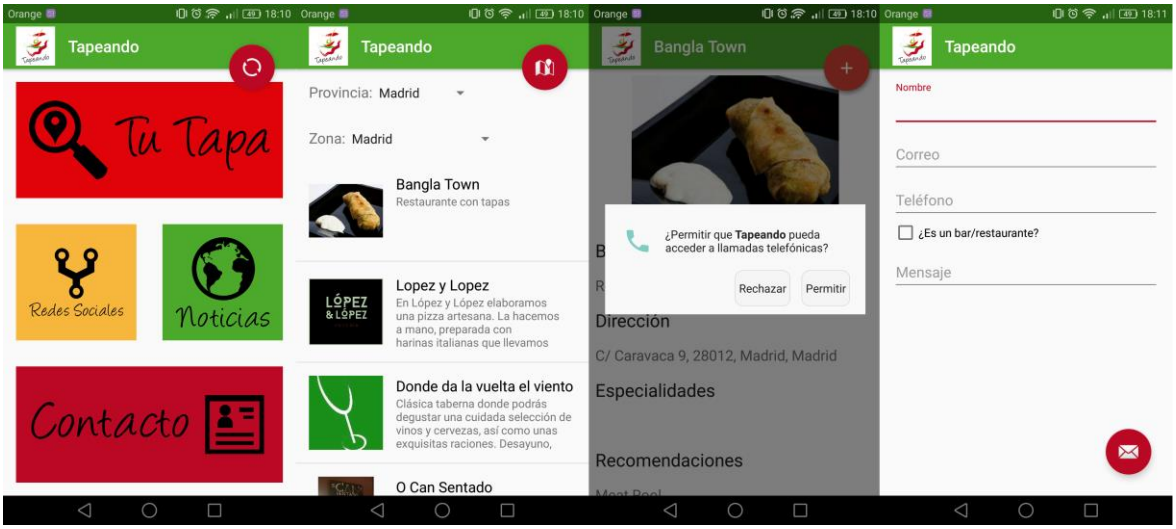


Figura 2-8: Aplicación "Tapeando: Tapas y pinchos"

[4] https://play.google.com/store/apps/details?id=com.jldes.tapeando

2.5 Bares tapas gratis

Esta aplicación está desarrollada para Android, ofrece un completo buscador permitiendo filtrar los locales por ubicación actual, dirección, localidad, ciudad y nombre introducido por texto.

Nos muestra una gran cantidad de información sobre los locales pudiendo ver opiniones.

Además, no permite añadir nuevos locales como usuario, es necesario que se realice un registro previo para poder añadir nuevos establecimientos, introduciendo gran cantidad de datos a mano. Estos deben ser validados por los administradores de la aplicación.

El registro te indica que sea por email sin embargo para acceder puedes usar otros medios como validación de cuenta de Google.

5

Figura 2-9: Icono de la app "Bares tapas gratis"

Figura 2-10: Aplicación "Bares tapas gratis"

[5] https://play.google.com/store/apps/details?id=es.taapas.app

2.6 Rutappa

Esta aplicación está desarrollada para Android e IOS, no ofrece la posibilidad de buscar locales. Únicamente muestra la ruta de tapeo más cercana sin ser posible agregar como usuario ningún nuevo establecimiento.

Además, la información la muestra en función de la tapa, no del establecimiento.

Está únicamente orientada a eventos gastronómicos.

[6]

Figura 2-11: Icono de la app "Rutappa"

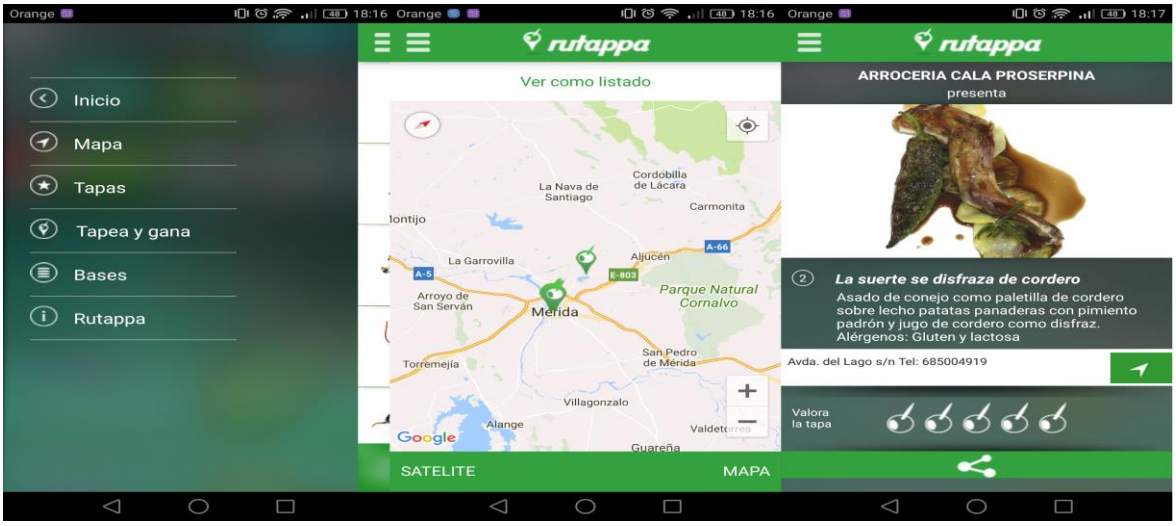


Figura 2-12: Aplicación "Rutappa"

[6] https://play.google.com/store/apps/details?id=retalis.com.rutappa

3 Objetivos y funcionalidades

3.1 Introducción

Como ya se especificó en el primer capítulo, este trabajo de fin de grado tiene la finalidad de definir, implementar y presentar una aplicación híbrida para Android e IOS que permita encontrar y/o planificar distintos lugares donde poder interaccionar socialmente.

3.2 Funcionalidades

F-1. Filtro de locales

En la pantalla de locales se podrán filtrar los locales por localidad.

F-2. Búsqueda

En la pantalla de locales el usuario podrá realizar una búsqueda por nombre tras realizarse el filtro inicial por localidad.

F-3. Añadir nuevo local

En la pantalla de locales el usuario podrá añadir nuevos establecimientos que se encuentren próximos a la ubicación actual, siendo la información facilitada directamente por la API de Google.

F-4. Añadir comentario y valoración

En la pantalla de detalles de los locales el usuario podrá añadir un nuevo comentario y valoración, modificando estos cuando desee.

F-5. Añadir imágenes

En la pantalla de detalles de los locales el usuario podrá añadir tantas imágenes como desee, siendo estas visibles para todos los usuarios.

F-6. Añadir como favorito

En la pantalla de detalles de los locales los usuarios podrán marcar establecimientos como favoritos pudiendo acceder a los detalles de estos desde la opción de ‘Favoritos’ del menú lateral.

F-7. Mapa

En la pantalla de mapa el usuario podrá ver los locales próximos a su ubicación actual haciendo uso de un mapa de la localidad.

F-8. Selección de idioma.

El usuario podrá configurar el idioma de la aplicación de forma persistente tantas veces como desee.

F-9. Reportar error.

El usuario podrá reportar errores de los locales a los administradores a través de la aplicación.

3.3 Objetivos

En esta aplicación se definen un gran número de objetivos, los cuales podemos agruparlos en los siguientes objetivos genéricos:

O-1. Mejorar interacción social

Este objetivo se consigue desarrollando una aplicación usable que incentive a quedar para interaccionar socialmente con otras personas además de promocionarse los propios locales.

O-2. Interfaz intuitiva

Este objetivo se consigue creando pantallas descriptivas en las cuales el usuario sepa que acción se realiza en esa pantalla, además de incluir el menor número de acciones por pantalla.

O-3. Experiencia satisfactoria

Es fundamental escuchar, cuidar y atender al cliente debido a los niveles de competencia. Por ello se debe dar una experiencia satisfactoria a los clientes ofreciendo buenos tiempo de respuesta tanto por parte de la aplicación como por parte del equipo de desarrollo para arreglar cualquier posible ‘bug’ que se detecte. Además de cuidar el diseño, accesibilidad, imágenes y calidad de los contenidos.

4 Herramientas y tecnologías

4.1 Plataformas

4.1.1 Android Studio

Android Studio [3] es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android. Este nos ofrece un editor de código inteligente además de integrar un AVD Manager (Android Virtual Devices Manager) que nos permite simular la aplicación sobre terminales virtuales pudiéndose modificar sus especificaciones.

Está disponible para Microsoft Windows, Mac OS X y Linux de manera gratuita.

4.1.2 Notepad++

Notepadd++ [4] es un editor de texto de soporte nativo para Microsoft Windows y de código fuente libre con soporte para varios lenguajes de programación. A pesar de la facilidad ofrecida por Android Studio se ha elegido esta herramienta como principal para el desarrollo de la aplicación por ser una herramienta rápida y sencilla que te permita trabajar desde cualquier equipo con Microsoft Windows instalado y la poca memoria necesaria para abrir los distintos ficheros.

4.2 Framework

4.2.1 Angular

Angular [5] es una plataforma de desarrollo para crear aplicaciones utilizando estándares web modernos, en nuestro caso hemos usado la versión 4. Los motivos por los cuales hemos elegido este framework frente a otros como React o Vue son:

- Es mantenido por Google.
- Puedes usar cualquiera de los lenguajes: ECMAScript 5, ES6, TypeScript o Dart.
- Se denomina un software evolutivo, es decir, permite crear mejoras sin afectar la forma en la que se crean aplicaciones.
- Renderiza el mismo código de manera distinta en navegadores webs y en apps móviles.
- Otro motivo es la alta compatibilidad que ofrece con otro de nuestro frameworks de desarrollo, Ionic, y nuestra base de datos.

4.2.2 Ionic

Ionic [6] es un framework orientado al desarrollo de aplicaciones hibridas, permitiéndonos a su vez desarrollar aplicaciones web, en nuestro caso usamos la versión 3.9.

Hemos elegido esta plataforma de desarrollo por los siguientes motivos:

- Una herramienta reciente que nos permite llegar a varias plataformas con una única fuente, teniendo un único proceso de desarrollo e implementación.

- El uso de tecnologías web nos permite usar el navegador para visualizar las aplicaciones, permitiendo un flujo de trabajo muy productivo.

4.3 Lenguajes de programación

4.3.1 JavaScript

Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, débilmente tipado y dinámico.

Los navegadores modernos interpretan el código desarrollado en este lenguaje mediante la implementación del Document Object Model (DOM).

4.3.2 TypeScript

Es un lenguaje de programación de código abierto que cuenta con herramientas de programación orientada a objeto. Es llamado Superset de Javascript, esto significa que si el navegador está basado en Javascript para este será transparente que el código original se desarrolló en TypeScript y ejecutará Javascript como lenguaje original.

4.3.3 HTML

Significa HyperText Markup Language y hace referencia al lenguaje de marcado para la elaboración de páginas web. Se considera el lenguaje web más importante sirviendo de referencia como estándar y siendo crucial para el desarrollo y expansión de la World Wide Web.

4.3.4 CSS

Significa Cascading Style Sheets, es un lenguaje que se usa para diseñar la forma de presentar páginas web con un diseño agradable. Suele ser un archivo de texto estructurado que presenta una serie de propiedades para actualizar la presentación del contenido HTML o XML (eXtensible Markup Language).

4.3.5 SASS

Significa Syntactically Awesome Stylesheets, es un metalenguaje de hojas de estilos en cascada (CSS) diseñado por Hampton Catlin y desarrollado por Nathan Weizenbaum. Tras la compilación se generan los ficheros CSS correspondientes.

4.3.6 JSON

Es el acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos.

Utiliza una sintaxis dedicada que se usa para identificar y gestionar los datos. Una de las principales ventajas que ofrece es que puede ser leído por cualquier lenguaje de programación, lo cual permite el intercambio de información entre distintas tecnologías.

En nuestro caso lo usaremos para leer los datos recibidos por la API de Google y de nuestra base de datos.

4.4 APIs

4.4.1 Google Places API for Web Services

Para realizar los rastreos de los establecimientos y la obtención detallada de los mismos, hemos utilizado la API de Google near places [7]. Para poder usar esta API debemos crear un proyecto en Google Developer y obtener una clave para acceder a la API.

De manera gratuita nos permite:

- Realizar mil llamadas a la API para obtener la información que necesitemos sobre los distintos establecimientos.
- Nos permite realizar búsquedas de sitios usando como base nuestra ubicación usando una serie de parámetros para filtrar la información.
- Además, nos permite solicitar detalles de los sitios donde podemos obtener imágenes, comentarios y gran cantidad de información.
- Por último, podemos añadir información a la base de datos de Google Places con los datos de nuestra aplicación.

4.5 Bases de datos

Para tomar la decisión de la base de datos por los motivos explicados en el apartado de diseño se realizó un estudio y pruebas de distintas bases de datos que se indican a continuación:

4.5.1 MongoDB Atlas

MongoDB Atlas [8] gracias a su motor de almacenamiento "MongoDB WiredTiger" que ofrece compresión y control de simultaneidad garantiza que cumpla con los SLA más exigentes.

A nivel de seguridad nos ofrece cifrado de datos en tránsito TSL/SSL y control de autorización y automatización.

4.5.2 Firebase Realtime Database

Realtime Database [9] es la base de datos original de Firebase, esta nos ofrece una solución eficiente para apps de dispositivos móviles que necesitan estados sincronizados entre los clientes en tiempo real gracias a su baja latencia.

Los datos son almacenados en un gran árbol JSON permitiendo almacenar datos simples de manera fácil, por otra parte, los datos complejos y jerárquicos son más difíciles de organizar.

Las consultas son profundas, motivo por el cual el resultado que ofrece siempre es un subárbol completo y no permite ordenar y filtrar en una única consulta.

A nivel de seguridad transmite en cascada las reglas de lectura y escritura, y los datos se deben validar por separado.

4.5.3 Firebase Cloud Firestore (BETA)

Cloud Firestore es la nueva base de datos insignia de Firebase para la programación de apps de dispositivos móviles. Aprovecha lo mejor de Realtime Database ofreciendo un nuevo modelo de datos más intuitivo.

Los datos son almacenados en documentos organizados en colecciones, permitiendo almacenar datos simples de manera fácil, por otra parte, los datos complejos y jerárquicos son más fáciles de organizar a escala, con subcolecciones dentro de los documentos.

Permite consultas superficiales para subcolecciones, motivo por el cual el resultado que ofrece no necesariamente es un colección o documento completo y permite ordenar y filtrar en una única consulta. Por todo esto el rendimiento es proporcional al tamaño del conjunto de resultados, no del conjunto de datos.

A nivel de seguridad las reglas no se aplican en cascada a menos que se indique, y los datos se validan de forma automática. En general la seguridad se vuelve más sencilla y potente para los SDK para dispositivos móviles, web y servidores.

5 Metodología y análisis

Este capítulo describe la metodología y análisis seguida en el desarrollo de la aplicación. En nuestro caso se ha seguido una metodología ágil [10,11], esta se refiere a métodos de ingeniería del software basados en un desarrollo iterativo e incremental donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto.

El "desarrollo ágil" es un término derivado del Manifiesto Ágil escrito en 2001, este estableció un conjunto de valores y principios dominantes para todas las metodologías ágiles. Esta se sustenta en 12 principios que podemos agrupar en cuatro valores:

- Los individuos y sus interacciones, siendo esencial una buena comunicación.
- Entregar software que funciona, sobre la documentación completa.
- Colaboración del cliente, sobre la negociación del contrato.
- Responder al cambio, por encima del seguimiento de un plan.

En cualquier metodología ágil existe un papel muy importante por parte del cliente o propietario del producto debido a la interacción constante y la realimentación continua. En nuestro caso no tenemos un equipo de desarrollo ni un cliente real. Por ello hemos realizado una metodología ágil adaptada.

Al utilizar este tipo de metodologías el análisis de requisitos desaparece, siendo sustituido por las "historias de usuario" que se obtienen mediante las conversaciones con los clientes. En nuestro caso se ha realizado un análisis de requisitos para obtener una idea sobre la funcionalidad básica del sistema, pudiendo añadir nuevas funcionalidades debido a la base iterativa e incremental de la metodología ágil. Tras realizar el análisis de los requisitos, las tareas y los casos de uso se indican en las siguientes subsecciones.

5.1 Casos de uso

Se han detectado y definido los siguientes casos de uso para el proyecto, estos se presentan en el siguiente diagrama:

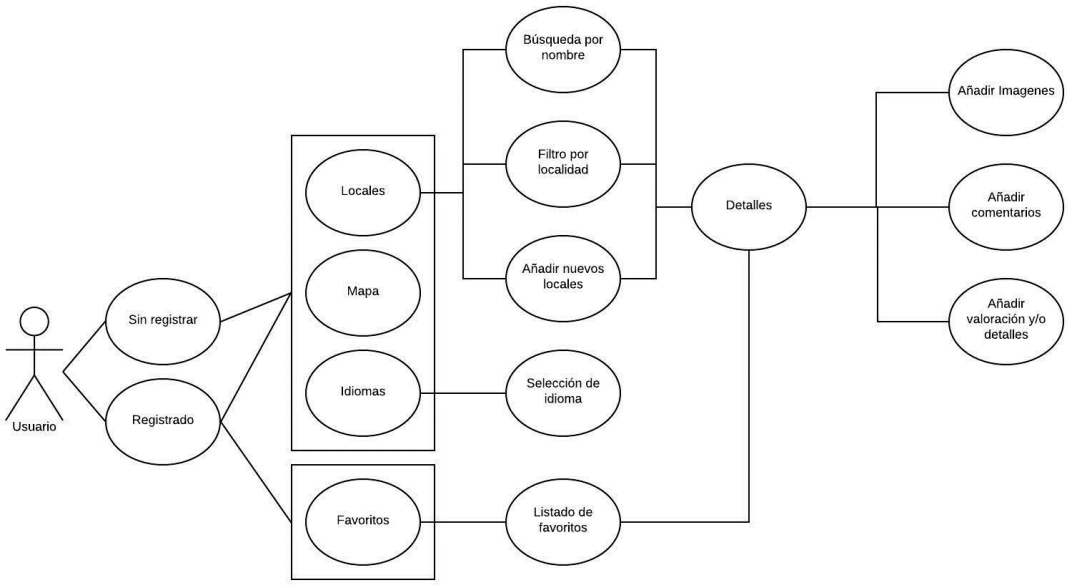


Figura 5-1: Diagrama de casos de uso para los usuarios

5.2 Requisitos funcionales

Los requisitos funcionales se encargan de definir las funciones del sistema de software o sus componentes.

RF-1. Registro de usuarios en el sistema.

Se registrará un usuario para darle de alta en el sistema usando una cuenta de Google y contraseña o una cuenta de Facebook para realizar un registro automático.

RF-2. Acceso del usuario en el sistema.

Acceso al sistema a través del uso de una cuenta de correo electrónico de Google y contraseña o Facebook, único para cada usuario para funcionalidad específica, en caso contrario no necesitará acceder con su cuenta registrada.

RF-3. Actualización de la información en la base de datos.

El sistema tendrá actualizada la información correspondiente a los establecimientos, su información general y datos específicos introducidos por cada usuario.

RF-4. Establecimientos

La aplicación mostrará un listado de establecimientos.

RF-4-1. Registro de nuevos locales.

Cualquier usuario podrá añadir nuevos establecimientos a la aplicación usando la geolocalización. Los datos se incluirán de manera automática sin duplicados.

RF-4.2. Listado de establecimientos.

La aplicación permitirá consultar un listado de establecimientos registrados en nuestra aplicación, esta mostrará la siguiente información de cada establecimiento: nombre, dirección, teléfono, valoración media de los usuarios, horarios, comentarios y galería de imágenes.

RF-5. Acceso a la API Google places

La aplicación debe obtener los establecimientos que se encuentran cerca de la geolocalización del usuario.

RF-6. Favoritos

Los usuarios registrados tendrán acceso a un listado de locales favoritos.

RF-6.1. Selección de establecimientos favoritos.

La aplicación permitirá a los usuarios logados seleccionar establecimientos como favoritos para su perfil y eliminarlos.

RF-6.2. Mostrar establecimientos favoritos.

La aplicación permitirá a los usuarios consultar sus establecimientos favoritos.

RF-7. Comentarios.

Los usuarios tendrán acceso a un listado de comentarios.

RF-7.1. Añadir comentarios

La aplicación permitirá añadir comentarios a los establecimientos una vez realizado su login en la aplicación.

RF-7.2. Mostrar comentarios.

La aplicación permitirá a los usuarios consultar todos los comentarios de los establecimientos.

RF-8. Galería

Los usuarios tendrán acceso a una lista de imágenes de los establecimientos.

RF-8.1. Subida de imágenes

Los usuarios registrados podrán añadir nuevas imágenes a la galería usando la cámara y la galería del dispositivo.

RF-8.2. Mostrar galería

Cualquier usuario tendrá disponible visualizar todas las imágenes de la galería.

RF-9. Compatibilidad

La aplicación deberá dar compatibilidad para los siguientes sistemas operativos implementados en los dispositivos móviles.

RF-9.1. Android

Para Android la aplicación deberá ser compatible con las SDKs iguales o superiores a 26.

RF-9.2. IOS

Para IOS la aplicación deberá ser compatible con versiones iguales o superiores a la 8.

RF-10. Reporte de error

La aplicación permitirá a los usuarios reportar errores sobre la información de los establecimientos.

RF-11. Conexión a internet

La conexión a internet es necesaria y fundamental para acceder a toda la funcionalidad de la aplicación, ya que esta usa una base de datos alojadas en la nube y acceso a la API de Google.

5.3 Requisitos no funcionales

Los requisitos no funcionales o también llamados atributos de calidad de un sistema de software se consideran aquellas restricciones o condiciones que impone el cliente al sistema que necesita, siendo cualidades que el producto debe tener.

RNF-1. Interfaz y usabilidad

Se creará una aplicación sencilla y usable desde el punto de vista de la agilidad, con una interfaz gráfica intuitiva basada en ventanas.

RNF-2. Seguridad

Un usuario no registrado podrá acceder a la aplicación para informarse de los establecimientos ya registrados y añadir más. Pero solo accederá a la funcionalidad especifica como favoritos y añadir comentarios los usuarios registrados gracias al sistema de autenticación.

RNF-3. Mantenibilidad y seguridad

La aplicación será programada en un lenguaje fácilmente portable a otras plataformas (Windows, Linux, Mac…) y a diferentes tipos de dispositivos (móviles inteligentes y tablets).

Se facilita el añadido de nuevas funcionalidades, en el caso que se desee en un futuro.

RNF-4. Disponibilidad y recursos

Para esta aplicación se ha asegurado que la base de datos tenga alta disponibilidad estando alojada en un servicio web que nos lo garantiza donde se conectará cada terminal para su uso.

RNF-5. Documentación

La aplicación estará disponible en español e inglés facilitando su uso a las personas que visiten nuestro país, además se debe poder incluir nuevos idiomas de manera sencilla para la aplicación.

5.4 Tareas a realizar

Las tareas favorecen el desarrollo distribuido del sistema pudiendo asignar distintas tareas a los desarrolladores. En nuestro caso al no tener un equipo de desarrollo nos ayuda a definir de manera clara las tareas y el orden para el desarrollo de la aplicación además de marcar unas pautas para el desarrollo.

T-1. Investigación de lenguajes para el desarrollo

Investigar distintos lenguajes de desarrollo para realizar la aplicación en función de nuestras necesidades, de manera sencilla e intuitiva.

T-2. Investigación, modelado y realización de las llamadas a la base de datos

Investigar distintas bases de datos que se ajusten a nuestras necesidades ofreciendo actualizaciones en tiempo real y estando alojada en la nube. Además, definición de un modelo de datos válido para nuestro sistema y la implementación de la clase y llamadas para acceder a nuestra información.

T-3. Investigación y realización de las llamadas a la API

Investigar cómo realizar las llamadas a la API de Google, información que nos facilita la respuesta de la llamada e implementar la clase y las llamadas pertinentes que se deben realizar desde la aplicación.

T-4. Investigación Geolocalización

Acceso a la geolocalización actual del dispositivo para realizar la búsqueda de locales en función de nuestra ubicación actual.

T-5. Menú lateral

Generar el esqueleto de la aplicación implementando el menú lateral de navegación para el usuario.

T-6. Sección 'Locales'

Listado de locales cercanos registrados en nuestra aplicación y/o nuevos locales cercanos a nuestra ubicación usando la geolocalización para añadir a la misma.

T-7. Sección 'Añadir locales'

Mostrar información del nuevo local seleccionado para añadir y completar información restante.

T-8. Sección 'Detalles de locales'

A partir de la lista de locales registrados en la aplicación mostrar información detallada de las características del local seleccionado.

T-9. Añadir imágenes

Selección de imágenes desde la galería o la cámara del dispositivo y añadir a los detalles de cada establecimiento guardando las imágenes en nuestra base de datos.

T-10. Sección 'Favoritos'

Mostrar un listado con los locales marcados como favoritos, pudiendo consultar su información y eliminarlos desde los detalles del local.

T-11. Sección 'Mapa'

Mostrar un mapa con los locales registrados en la aplicación en función de la localidad seleccionada de las existentes en nuestra base de datos.

6 Diseño e implementación

6.1 Diseño

6.1.1 Arquitectura de la aplicación

El sistema sigue un diseño arquitectónico basado en tres capas usando un patrón de diseño modelo-vista-controlador (MVC) [12]. Este diseño nos permite realizar una división de la lógica gracias a su modularidad, fácil colaboración y reutilización del código construyendo una aplicación flexible.

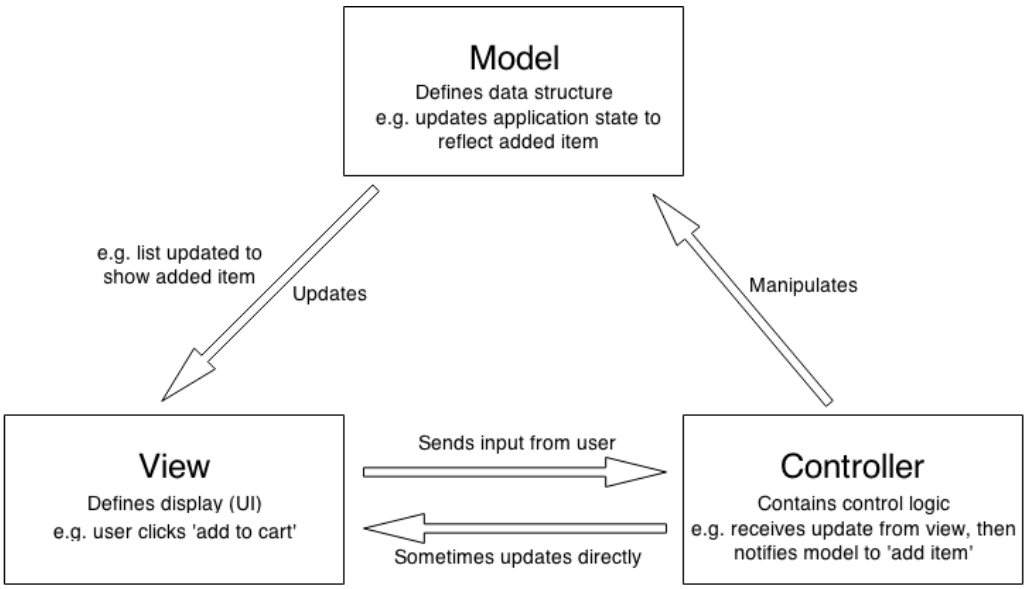


Figura 6-1: Patrón MVC

A continuación, explicaremos en detalle cada una de las partes que componen este patrón:

Modelo

El modelo define los datos que debe contener la aplicación, en el caso que el estado de los datos se modifique este generalmente notificará estos cambios a la vista. Se encarga de recibir las peticiones y mostrar los resultados en la vista, además de recuperar y almacenar los nuevos datos.

Vista

La vista define qué datos deben mostrarse, presenta el sistema al usuario comunicando e introduciendo la información por el mismo. Esta incluye la interfaz de usuario y debe cumplir una característica fundamental, ser “amigable”.

Controlador

El controlador contiene la lógica que actualiza el modelo y/o la vista en respuesta a la entrada de los usuarios en la aplicación.

6.1.2 Diseño de la base de datos

A la hora de realizar el diseño de la base de datos inicialmente se valoró si usar un base de datos relacional (SQL) o no relacional (NoSQL), en nuestro caso se optó por una base de datos NoSQL la cual no garantiza completamente obtener atomicidad, consistencia, aislamiento y durabilidad, aunque todo lo anterior lo vemos compensado gracias a su escalabilidad y rendimiento frente a determinados modelos de datos.

Tras esta decisión y teniendo presente que buscábamos una base de datos que nos permitiese actualizar los datos en tiempo real, se realizó un estudio y pruebas de distintas bases de datos que se indican en el apartado de herramientas y tecnologías se decidió para almacenar la información una base de datos NoSQL Firebase Cloud Firestore (BETA) orientada a los documentos, esta no tiene tablas ni filas como sucede en las bases de datos SQL.

Almacena los datos con pares de clave-valor que se almacenan en documentos organizados en colecciones. Estos documentos pueden contener a su vez subcolecciones y objetos anidados hasta una profundidad máxima de 100 niveles.

Esta base de datos es altamente escalable teniendo en cuenta que el rendimiento de una consulta es proporcional al tamaño de su conjunto de resultados y no a su conjunto de datos, por tanto, la búsqueda se mantendrá rápida independientemente de lo grande que pueda ser su conjunto de datos.

Además, nos permite escuchar tanto datos en tiempo real como realizar búsquedas manuales, es decir, si se desean los datos una sola vez.

También debemos destacar la seguridad que ofrece, esta es una seguridad simple pero potente que realiza una validación automática de los datos.

Para nuestra aplicación, se ha diseñado el siguiente modelo de datos:

En primer lugar, se ha generado una colección llamada “Provincias”, en ella se almacenan varios documentos que indicarán los códigos ISO2 de los países. Cada documento contiene varios registros claves-valor, estos registros tendrán como clave el nombre de una provincia y como valor, un array de las localidades registradas de cada provincia.

Colección	Documento	Campos		
Provincias	Código ISO2 del país	Clave	Valor	Tipo
		Nombre de provincia	Localidades	Array

Figura 6-2: Colección ‘Provincias’

En segundo lugar, se ha generado una colección llamada “Locales”, en ella se almacenan varios documentos que vendrán indicados por identificadores únicos. En nuestro caso los identificadores coinciden con las claves facilitadas por la API de Google para mantener un mejor control en el futuro y evitar duplicar datos facilitados por las peticiones, excepto para los locales registrados de manera manual que no tendrán el

identificador facilitado por la API de Google hasta ser validados y registrados por nuestros administradores.

Cada documento contiene varios registros clave-valor, siendo estos los siguientes:

- Clave **"Name"** es de tipo string y contiene el nombre del local.
- Clave **"Vinicity"** es de tipo string y contiene la dirección del local.
- Clave **"Lat"** es de tipo number y contiene la latitud del local, es decir, la distancia angular entre la línea ecuatorial y el local.
- Clave **"Lng"** es de tipo number y contiene la longitud del local, es decir, la distancia angular entre el meridiano y el local.
- Clave **"Locality"** es de tipo string y contiene el nombre de la localidad donde se encuentra el local.
- Clave **"Validate"** es de tipo boolean, este valor nos indica si el local está validado por los administradores. En caso de realizarse un registro automático se define como validado.
- Clave **"UrlImg"** es de tipo string y contiene la imagen de carga inicial del local.
- Clave **"Place_Id"** es de tipo string y contiene el id del local facilitado por la API de Google.
- Clave **"Favoritos"** es un array que contendrá los nombres de usuarios que tienen marcado el local como favorito.

Colección	Documento	Campos		
Locales	Id del local (Clave de Google)	**Clave**	**Valor**	**Tipo**
		Name	Nombre del local	String
		Vinicity	Dirección del local	String
		Lat	Latitud	Number
		Lng	Longitud	Number
		Locality	Localidad del local	String
		Validate	Bandera de validación	Boolean
		UrlImg	url de la imagen de carga del local	String
		place_id	Id del local por Google	String
		Favoritos	Nombres de usuarios	Array

Figura 6-3: Colección 'Locales'

En tercer lugar, se ha generado una colección llamada "LocalesDetails", en ella se almacenan varios documentos que vendrán indicados por identificadores únicos, en este caso comparten el mismo identificador con el que se ha añadido a la colección "Locales".

Cada documento contiene varios registros clave-valor, estos son los siguientes:

- Clave **"Administrative_area_level_2"** es de tipo string y contiene el nombre de la provincia donde se encuentra el local.
- Clave **"CountryCode"** es de tipo string y contiene el código ISO2 del país del local.
- Clave **"CountryName"** es de tipo string y contiene el nombre del país del local.

- Clave **“RatingTotal”** es de tipo number y contiene la valoración media dada por los usuarios sobre el local.
- Clave **“Weekday_text”** es de tipo array/string, en caso de no registrar un horario, almacenamos un string indicando que no tenemos el horario; si se ha registrado el horario, se almacena un array de string con los horarios.
- Clave **“Collection”** es un array, donde almacenamos las urls de las imágenes registradas para ese local en nuestra aplicación. Las imágenes se alojan en nuestra base de datos.
- Clave **“Website”** es de tipo string, donde se indica la url del local.
- Clave **“Formatted_phone_number”** es de tipo string, donde se almacena el número de contacto del local.
- Clave **“Reviews”** contiene una subcolección. Esta contiene los siguientes campos:

    - Clave **“Comentario”** es de tipo string donde se incluye un comentario dado por un usuario.
    - Clave **“Author_name”** es un string donde se registra el nombre del usuario que ha registrado el comentario.
    - Clave **“Rating”** es un number, esta incluye la valoración personal del usuario sobre el local.

Colección	Documento	Campos		
LocalesDetails	Id del local (Clave de Google)	**Clave**	**Valor**	**Tipo**
		Administrative_area_level_2	Provincia	String
		CountryCode	Código ISO2 del país	String
		CountryName	Nombre del país	String
		RatingTotal	Valoración total	Number
		Weekday_text	Horario	Array/String
		Website	Url del local	String
		Formatted_phone_number	Número de contacto	String
		Collection	Urls de las imágenes subidas	Array
		Reviews	Comentarios y valoraciones	Obj

Clave	Valor	Tipo
Comentario	Comentario	String
Author_name	Nombre de usuario	String
Rating	Valoración personal	Number

Figura 6-4: Colección ‘LocalesDestails’

Por último, se ha generado una colección llamada “Errors”, en ella se almacenan varios documentos que contienen el registro de errores reportados por los usuarios.

Cada documento contiene varios registros clave-valor, estos son los siguientes:

- Clave **“Id_documento”** es de tipo string y contiene el id que hace referencia al local.
- Clave **“Comment”** es de tipo string y contiene el comentario añadido por el usuario.
- Clave **“Error”** es de tipo string y contiene el tipo de error.

Colección	Documento	Campos		
Errors	Id generado automáticamente	Clave	Valor	Tipo
		Id_documento	Id de referencia al local	String
		Comment	Comentario del usuario	String
		Error	Tipo de error	String

Figura 6-5: Colección 'Errors'

Como se puede observar este modelo de datos está orientado para tener un buen escalado y rendimiento pudiendo acceder a la información de manera sencilla.

6.1.3 Carga de nuevos datos

Para realizar la carga en nuestra base de datos de nuevos datos se decidió facilitar lo máximo posible la actividad de los usuarios buscando una carga semi-automática de la información. Para ello se ha valorado el uso de la API "Google Places API for Web Services" la cual supone una funcionalidad clave de la aplicación.

Gracias a distintas llamadas realizadas en la API de Google, que se explicarán en el apartado de implementación, obtenemos entorno al 80% de los datos de registro de información, además nos aseguramos de que esa información es real gracias a su validación interna de los establecimientos.

6.1.4 Diagrama de navegación

A continuación, se muestra un diagrama de navegación que representa cada una de las pantallas de visualización del usuario.

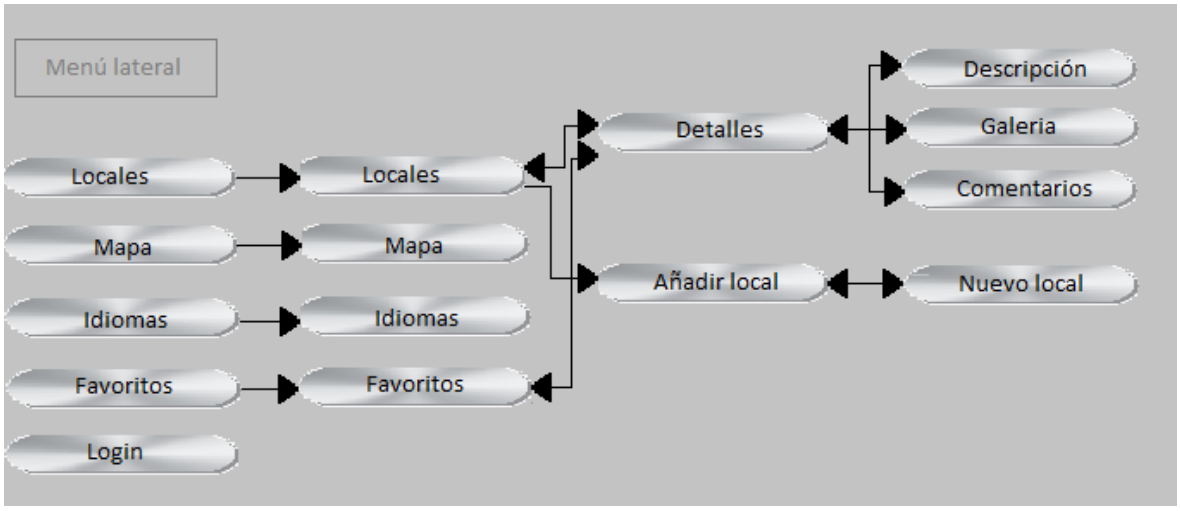


Figura 6-6: Diagrama de navegación

La página principal será "Locales", la cual nos muestra información sobre los locales registrados en nuestra base de datos independientemente de haber realizado login en la aplicación.

Como podemos observar tenemos un menú lateral desde donde el usuario podrá navegar a cualquiera de las pantallas principales desde donde se encuentre, además podrá realizar login en la aplicación en cualquier momento.

Desde cada una de las listas de establecimientos registrados en nuestra base de datos podemos acceder a la actividad que muestra de manera detallada el establecimiento seleccionado, además si el usuario ha realizado login en la aplicación podrá añadir un nuevo comentario y valoración sobre el local seleccionado.

Únicamente desde la página de "Locales" los usuarios podrán acceder a un nuevo listado de locales facilitados por la API de Google en función de nuestra ubicación actual para posteriormente añadir un nuevo local a nuestra base de datos, tras finalizar este proceso el usuario volverá a la pantalla principal.

6.2 Implementación

6.2.1 Base de datos

Para realizar la implementación de la base de datos, lo primero que debemos hacer es acceder a la página de Firebase y crear un nuevo proyecto, este nos generará una key para poder establecer la conexión entre nuestra APP y la base de datos. Tras esto debemos agregar las bibliotecas y dependencias necesarias de nuestra app antes de inicializar la instancia a Cloud Firestore usando nuestra key como se muestra a continuación.

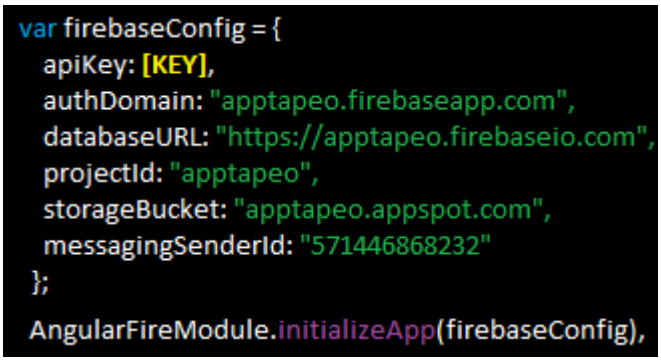


Figura 6-7: Configuración Firebase

Para realizar las peticiones debemos tener en cuenta que según la estructura que nos ofrece este tipo de base de datos podemos acceder a las colecciones completas o a los documentos de cada colección en función de la información que necesitemos para así optimizar el tiempo de respuesta, además de poder sobrescribir o actualizar los documentos. A continuación, vamos a explicar cómo insertar y cargar la información de la BD.

Para añadir nuevos documentos a nuestra colección tenemos dos opciones, si no deseamos especificar el id del documento podemos llamar al método 'add' en caso contrario usaremos el método 'set', ambos métodos nos permiten insertar la información que deseemos en formato JSON en una colección como se indica a continuación.

```
this.afs.collection([Nombre de la colección])
.add(OBJ)
.then(function() {
        console.log("Document successfully written!");
})
.catch(function(error) {
        console.error("Error writing document: ", error);
});

this.afs.collection([Nombre de la colección]).doc([Nombre del documento])
.set(OBJ)
.then(function() {
        console.log("Document successfully written!");
})
.catch(function(error) {
        console.error("Error writing document: ", error);
});
```

Figura 6-8: Métodos para añadir datos de Firebase

Si el documento no existe en esa colección lo creará, en caso contrario sobrescribirá su contenido. Para evitar sobrescribir la información en el caso de querer añadir nueva información tenemos dos alternativas, realizar un 'merge' entre los documentos en el método 'set' o realizar la petición con el método 'update', según la documentación facilitada por Firebase es recomendable usar la segunda opción para actualizar información y en ambos casos debemos conocer el id del documento.

Es recomendable tener claro qué acción vamos a realizar en función de la existencia o no de los documentos para así optar por los métodos que más se ajusten a nuestras necesidades. Ahora podemos observar cómo se realizan ambas peticiones.

```
this.afs.collection([Nombre de la colección]).doc([Nombre del documento])
.set(OBJ)
.then(function() {
        console.log("Document successfully written!");
})
.catch(function(error) {
        console.error("Error writing document: ", error);
});

this.afs.collection([Nombre de la colección])).doc([Nombre del documento])
.update(OBJ)
.then(function() {
        console.log("Document successfully update!");
})
.catch(function(error) {
        console.error("Error updating document: ", error);
});
```

Figura 6-9: Métodos para actualizar datos de Firebase

Una vez tenemos información añadida en nuestra BD también podemos borrar esa información. Tenemos dos alternativas, por un lado, eliminar un documento entero usando el método 'delete' como se indica a continuación.

```
this.afs.collection([Nombre de la colección]).doc([Nombre del documento])
.delete()
.then(function() {
        console.log("Document successfully deleted!");
}).catch(function(error) {
        console.error("Error removing document: ", error);
});
```

Figura 6-10: Método para eliminar datos de Firebase

Por otro lado, podemos borrar campos específicos de un documento con el método 'FieldValue.delete' durante la actualización de un documento.

```
this.afs.collection([Nombre de la colección]).doc([Nombre del documento])
.update({
        [Nombre del campo]: firebase.firestore.FieldValue.delete()
})
.then(function() {
        console.log("Document successfully update!");
})
.catch(function(error) {
        console.error("Error updating document: ", error);
});
```

Figura 6-11: Método para eliminar campos de Firebase

Por último, las peticiones para obtener los datos nos ofrecen varias posibilidades, entre ellas tenemos las peticiones básicas de datos y las peticiones que nos permiten obtener actualizaciones en tiempo real de los datos. Estas peticiones además nos permiten filtrar por los parámetros que deseemos la información solicitada y obtener una respuesta especifica e incluso ordenar o limitar los datos de respuesta. El siguiente ejemplo nos muestra realizar una petición de todos los documentos de una colección al no especificar el id del mismo en la petición, aunque tras la petición hemos realizado un filtro. Esta petición no nos ofrece actualización en tiempo real de los datos.

```
this.afs.collection([Nombre de la colección]).doc()
.ref
.get()
.then(function(doc) {
        if (doc.exists) {
                doc.data();
        } else {
                console.log("No such document!");
        }
}).catch(function(error) {
        console.log("Error getting document:", error);
});
```

Figura 6-12: Método para leer datos de Firebase

Para realizar una petición que nos permita obtener actualización de los datos en tiempo real tenemos la opción de usar el método 'onSnapshot' después de indicar el id del documento para realizar una escucha solo sobre él o en su defecto para escuchar los cambios en varios documentos sin especificar el id. También podemos usar el método 'snapshotChanges'.

6.2.2 Peticiones Google Places API for Web Service

Para realizar la implementación de las peticiones a la API de Google, primero debemos ingresar en la página de 'Google API Console' donde crearemos nuestro proyecto. Tras crear el proyecto debemos habilitar la API y los servicios relacionados para poder obtener nuestra clave de API. Esta clave es necesaria para realizar la conexión con la API, esta nos permite controlar el uso de la API además de permitir que Google pueda ponerse en contacto con nosotros por temas relacionados con nuestra aplicación.

Una vez realizado este primer paso y teniendo en cuenta que el código de JavaScript para la Google Maps API realiza la carga de todos los objetos y símbolos principales de JavaScript que se usarán a través de una URL de arranque donde se especifica además de la clave API de nuestro proyecto las bibliotecas a las cuales queremos acceder. Están disponibles las siguientes bibliotecas:

- Drawing, la cual proporciona una interfaz para dibujar polígonos, círculos, marcadores y más posibilidades a los usuarios.
- Geometry, esta ofrece funcionalidades para calcular valores geométricos escalares.
- Places, que nos permite buscar establecimientos, sitios, puntos de interés… dentro de un área definida.
- Visualization, la cual ofrece datos visuales como mapas de calor.

En nuestro caso hemos accedido únicamente a la librería 'Places' para así obtener información sobre establecimientos dentro de un área especifico. Esta solicitud de arranque se indica directamente en el fichero 'index.html' como se indica a continuación.

```
<script type="text/javascript"
        src="http://maps.google.com/maps/api/js?key=[YOUR_API_KEY]&libraries=places">
</script>
```

Figura 6-13: Configuración de Google Places API for Web Service

Una vez realizada la solicitud de arranque ya podemos comenzar a realizar la solicitud de búsqueda. Para ello debemos crear un mapa que le pasaremos al servicio de PlacesService antes de llamar al método que nos devolverá la búsqueda de sitios cercanos 'NearbySearch'.

En la llamada al método debemos indicar de manera obligatoria los parámetros 'location' y 'radius', en nuestro caso también pasamos el parámetro 'types' donde se especifica el tipo de establecimientos que buscamos. Además, como paso previo a este, calculamos la ubicación actual, es decir, latitud y longitud de la posición actual que se establece como punto de origen de la búsqueda aplicando el radio seleccionado, este como máximo debe ser 50.000 metros. Además de estos parámetros, el método admite otros muchos más como 'minPriceLevel', 'maxPriceLevel', 'name', 'openNow'...

```
var latLng= new google.maps.LatLng(latitude,longitude);

map = new google.maps.Map(document.getElementById('map'), {
  center: latLng,
  zoom: 15
 });

var request = {
 location: latLng,
 radius: '1000',
 types: ['bar','restaurant']
};

service = new google.maps.places.PlacesService(map);
        return new Promise((resolve,reject)=>{
                service.nearbySearch(request,function(results,status){
                        if(status === google.maps.places.PlacesServiceStatus.OK)
                        {
                                resolve(results);
                        }else
                        {
                                reject(status);
                        }
                });
        });
}
```

Figura 6-14: Petición de establecimientos cercanos

En caso de querer realizar la petición directamente mediante un método get se crearía de la siguiente manera.

```
this.http.get('https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=[LAT,LNG]&type=[TYPE]
&radius=[RADIUS]&key=[YOUR_API_KEY]', {}, {})
.then(data =>{
        result = data.data;
})
.catch(error=>{
        console.log(error.status);
});
```

Figura 6-15: Petición de establecimientos cercanos por URL

El objeto de respuesta de PlacesServiceStatus contiene el estado de solicitud que se debe tratar en consecuencia como 'OK', 'ERROR', 'REQUEST_DENIED'… Los resultados de la búsqueda devuelven un subconjunto de campos tales como nombre, dirección, tipos, identificador único del sitio, coordenadas, valoración media… A continuación, se muestra un ejemplo donde podemos ver cómo se resuelve la petición de la búsqueda.


```
{
"html_attributions": [],
"next_page_token": "CvQD4QEAAFMoLvxBgX...aRWvkbxIAGRgdYea",
"results": [
{
        "geometry": {
                "location": {
                        "lat": 40.96371109999999,
                        "lng": -5.665922899999999
                },
                "viewport": {
                        "northeast": {
                                "lat": 40.9650037802915,
                                "lng": -5.664501569708498
                        },
                        "southwest": {
                                "lat": 40.9623058197085,
                                "lng": -5.667199530291502
                        }
                }
        },
        "icon": "https://maps.gstatic.com/mapfiles/place_api/icons/restaurant-71.png",
        "id": "ba2b9d386ca72f97d5dca209c2e4a3466c81104d",
        "name": "Restaurante Café Corrillo",
        "opening_hours": {
                "open_now": true,
                "weekday_text": []
        },
        "photos": [
                {
                "height": 2448,
                "html_attributions": [
                        "<a href=\"https://maps.google.com/maps/contrib/103917321569137532531/photos\">Arantza Rotaetxe</a>"
                ],
                "photo_reference": "CmRaAAAAakkBzDjPtm2z...SbN-BJqw",
                "width": 3264
                }
        ],
        "place_id": "ChIJAyvHnhUmPw0Ra3D_EdSS-sI",
        "rating": 3.8,
        "reference": "CmRSAAAALm-LT2...VJ5N6GzDlQ",
        "scope": "GOOGLE",
        "types": [
                "cafe",
                "night_club",
                "bar",
                "restaurant",
                "food",
                "point_of_interest",
                "establishment"
        ],
        "vicinity": "Calle Meléndez, 18, Salamanca"
        },
```


Figura 6-16: Respuesta JSON de los establecimientos cercanos

Esta petición nos facilita información general sobre los establecimientos, para obtener detalles más específicos sobre ellos debemos realizar una llamada específica indicando el identificador único del establecimiento. En nuestro caso y para evitar saturar la aplicación tras realizar la petición al servicio de búsqueda solo realizamos la petición de detalles si el usuario desea registrar el establecimiento en nuestra aplicación guardando la información en nuestra propia base de datos.

El método que se encarga de realizar esta solicitud es 'getDetails' al cual debemos pasar el parámetro 'placeId' con el identificador único del establecimiento. En caso de no querer usar el método, podemos realizar una petición usando el método get de HTTP pasando la url completa. A continuación, se muestran ambas opciones:

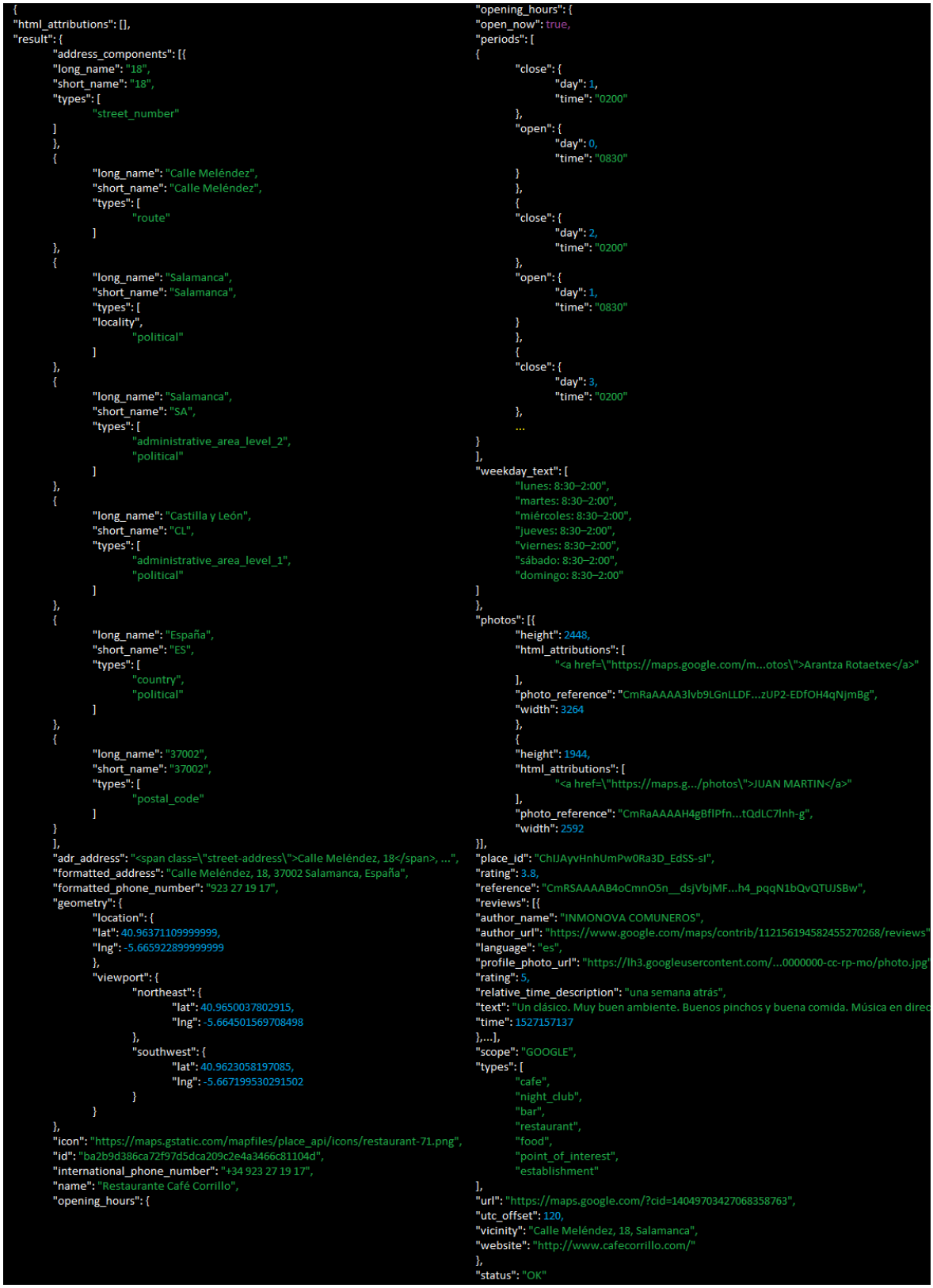
```
var request = {
  placeId: 'ChIJN1t_tDeuEmsRUsoyG83frY4'
};

service = new google.maps.places.PlacesService(map);
        return new Promise((resolve,reject)=>{
        service.getDetails(request, function(place, status) {
                if (status == google.maps.places.PlacesServiceStatus.OK) {
                        resolve(results);
                }else{
                        reject(status);
                });
        });
```

Figura 6-17: Petición de detalles de un establecimiento

```
this.http.get('https://maps.googleapis.com/maps/api/place/details/json?placeid=[place_Id]&key=[YOUR_API_KEY]', {}, {})
.then(data =>{
        result = data.data;
})
.catch(error=>{
        console.log(error.status);
});
```

Figura 6-18: Petición de detalles de un establecimiento por URL

Como hemos indicado anteriormente es necesario realizar esta petición para obtener información específica del establecimiento. En este caso obtenemos un gran nivel de detalle, además de la información recibida con la petición 'nearbySearch' tenemos los campos 'photos' con las fotografías compartidas por los usuarios, 'price_level' que nos indica el rango de precio del establecimiento variando entre 1-5, 'rating' con la valoración media ofrecida por los usuarios, 'reviews' este campo nos muestra una lista de valoraciones, nombre y comentarios dados por los usuarios… A continuación, se muestra la respuesta específica de un local concreto.



```
{
"html_attributions": [],
"result": {
        "address_components": [{
        "long_name": "18",
        "short_name": "18",
        "types": [
                "street_number"
        ]
        },
        {
                "long_name": "Calle Meléndez",
                "short_name": "Calle Meléndez",
                "types": [
                        "route"
                ]
        },
        {
                "long_name": "Salamanca",
                "short_name": "Salamanca",
                "types": [
                "locality",
                        "political"
                ]
        },
        {
                "long_name": "Salamanca",
                "short_name": "SA",
                "types": [
                        "administrative_area_level_2",
                        "political"
                ]
        },
        {
                "long_name": "Castilla y León",
                "short_name": "CL",
                "types": [
                        "administrative_area_level_1",
                        "political"
                ]
        },
        {
                "long_name": "España",
                "short_name": "ES",
                "types": [
                        "country",
                        "political"
                ]
        },
        {
                "long_name": "37002",
                "short_name": "37002",
                "types": [
                        "postal_code"
                ]
        }
        ],
        "adr_address": "<span class=\"street-address\">Calle Meléndez, 18</span>, ...",
        "formatted_address": "Calle Meléndez, 18, 37002 Salamanca, España",
        "formatted_phone_number": "923 27 19 17",
        "geometry": {
                "location": {
                "lat": 40.96371109999999,
                "lng": -5.665922899999999
                },
                "viewport": {
                        "northeast": {
                                "lat": 40.9650037802915,
                                "lng": -5.664501569708498
                        },
                        "southwest": {
                                "lat": 40.9623058197085,
                                "lng": -5.667199530291502
                        }
                }
        },
        "icon": "https://maps.gstatic.com/mapfiles/place_api/icons/restaurant-71.png",
        "id": "ba2b9d386ca72f97d5dca209c2e4a3466c81104d",
        "international_phone_number": "+34 923 27 19 17",
        "name": "Restaurante Café Corrillo",
        "opening_hours": {
        "open_now": true,
        "periods": [
        {
                "close": {
                        "day": 1,
                        "time": "0200"
                },
                "open": {
                        "day": 0,
                        "time": "0830"
                }
                },
                {
                "close": {
                        "day": 2,
                        "time": "0200"
                },
                "open": {
                        "day": 1,
                        "time": "0830"
                }
                },
                {
                "close": {
                        "day": 3,
                        "time": "0200"
                },
                ...
        }
        ],
        "weekday_text": [
                "lunes: 8:30–2:00",
                "martes: 8:30–2:00",
                "miércoles: 8:30–2:00",
                "jueves: 8:30–2:00",
                "viernes: 8:30–2:00",
                "sábado: 8:30–2:00",
                "domingo: 8:30–2:00"
        ]
        },
        "photos": [{
                "height": 2448,
                "html_attributions": [
                        "<a href=\"https://maps.google.com/m...otos\">Arantza Rotaetxe</a>"
                ],
                "photo_reference": "CmRaAAAA3lvb9LGnLLDF...zUP2-EDfOH4qNjmBg",
                "width": 3264
                },
                {
                "height": 1944,
                "html_attributions": [
                        "<a href=\"https://maps.g.../photos\">JUAN MARTIN</a>"
                ],
                "photo_reference": "CmRaAAAAH4gBfIPfn...tQdLC7Inh-g",
                "width": 2592
        }],
        "place_id": "ChIJAyvHnhUmPw0Ra3D_EdSS-sI",
        "rating": 3.8,
        "reference": "CmRSAAAAB4oCmnO5n__dsjVbjMF...h4_pqqN1bQvQTUJSBw",
        "reviews": [{
        "author_name": "INMONOVA COMUNEROS",
        "author_url": "https://www.google.com/maps/contrib/112156194582455270268/reviews"
        "language": "es",
        "profile_photo_url": "https://lh3.googleusercontent.com/...0000000-cc-rp-mo/photo.jpg"
        "rating": 5,
        "relative_time_description": "una semana atrás",
        "text": "Un clásico. Muy buen ambiente. Buenos pinchos y buena comida. Música en direc
        "time": 1527157137
        },...],
        "scope": "GOOGLE",
        "types": [
                "cafe",
                "night_club",
                "bar",
                "restaurant",
                "food",
                "point_of_interest",
                "establishment"
        ],
        "url": "https://maps.google.com/?cid=14049703427068358763",
        "utc_offset": 120,
        "vicinity": "Calle Meléndez, 18, Salamanca",
        "website": "http://www.cafecorrillo.com/"
        },
        "status": "OK"
```


Figura 6-19: Respuesta JSON a petición de detalles

7 Pruebas y resultados

7.1 Pruebas

Las pruebas realizadas en la aplicación se pueden dividir en 3:

- **Pruebas unitarias**, estas pruebas se han realizado por el equipo de desarrollo y se realizan sobre todos los módulos y tareas. Tienen como objetivo encontrar defectos en la aplicación y solventarlos, repitiéndose hasta verificar que cualquier corrección previa no genera nuevos defectos, verificando que todos los elementos funcionan correctamente de manera individual.

- **Pruebas del sistema**, estas pruebas se han realizado por el equipo de desarrollo y tiene como objetivo las siguientes funcionalidades:
    - Verificar que el sistema es utilizable y funciona de manera correcta bajo todas las condiciones normales y anormales.
    - Hay que asegurar que se entregan los componentes y correcciones de alta calidad.
    - Identificar y devolver componentes defectuosos al equipo de desarrollo.

- **Pruebas de cliente**, estas son las últimas pruebas y son realizadas por el cliente, en este caso por familiares y amigos, estas pruebas se realizan de manera independiente y se encargan de verificar que el producto sea lo que se ha indicado en el contrato.

Durante todo el desarrollo se han realizado pruebas unitarias y del sistema por parte del equipo de desarrollo, en este caso un solo miembro, y en última instancia se han realizado pruebas por parte del cliente para verificar el correcto funcionamiento del sistema.

7.2 Resultados

Los resultados finales que se han obtenido en la realización de las pruebas son altamente satisfactorios, a nivel de funcionalidad cumple con las expectativas planteadas a lo largo del desarrollo. Aunque se ha detectado que en áreas con un acceso limitado a internet los tiempos generales de respuesta son más altos que se áreas con buena conectividad, lo que nos lleva a replantearnos un mejor manejo de las llamadas realizadas a los servidores para manejar mejor esos tiempos de respuesta. Por otra parte, se añade un pequeño documento de plan de pruebas funcionales donde se reflejan las principales características de la aplicación.

8 Conclusiones y trabajo futuro

8.1 Conclusiones

Durante el desarrollo de este proyecto se ha implementado una APP multiplataforma para Android e IOS que nos permite obtener datos en tiempo real mediante el uso de la API de Google y la conexión a nuestra base de datos buscando un resultado satisfactorio para el usuario final.

Para alcanzar este objetivo, en primer lugar, se realizó un análisis inicial de requisitos, objetivos, herramientas y tecnologías. Gracias a este análisis, se ha visto claramente la importancia de tomar una buena decisión a la hora de elegir las herramientas y tecnologías que necesitamos para el desarrollo de nuestras aplicaciones.

Durante esta primera etapa del proyecto, se ha podido comprobar que existe una mayor afinidad entre distintas tecnologías, herramientas y lenguajes y el ser capaz de comprender esto ayuda a crear aplicaciones de manera más sencilla y eficiente. Además de la afinidad también es muy importante el nivel de documentación que ofrecen las mismas ya que los lenguajes y herramientas más usados por la comunidad ofrecen muchísima mejor documentación, y al trabajar un mayor número de personas con ellas, en los foros se puede encontrar a muchísimas más personas que han compartido soluciones a algunos de esos mismos problemas.

Además, en la actualidad no es necesario tener un perfil *Full-stack* para desarrollar de manera individual la propia aplicación, ya que, al menos a nivel del *Backend*, existen muchas herramientas facilitadas en las propias bases de datos para gestionar todo el tránsito de esos datos y el nivel de seguridad. Un claro ejemplo de esto son las funcionalidades que ofrece la base de datos Firebase Cloud Firestore ya que esta nos permite administrar la seguridad, crear reglas para los datos e incluso añadir métodos para la gestión de información desde la propia interfaz de la base de datos sin necesidad de tener grandes conocimientos sobre el tema, con el apoyo en la completa documentación que ofrece.

Por otra parte, el haber realizado un buen análisis de requisitos y objetivos antes de comenzar el desarrollo de la aplicación ha permitido tener una clara idea de la división de los servicios para evitar, en la medida de lo posible, duplicar información y permitiendo así que cualquier modificación solo afecte a un único fichero, facilitando cualquier tipo de modificación futura tanto al desarrollador original como a cualquier otro desarrollador.

También se ha aprendido la importancia de realizar aplicaciones que puedan migrar de manera sencilla entre distintos sistemas operativos o que puedan generarse para varios de esos sistemas operativos directamente, pudiendo así llegar a un mayor número de usuarios. En nuestro caso se ha desarrollado la aplicación para dos sistemas operativos, pero, al aparecer cada día nuevas herramientas como es el caso de ‘Capacitor’, que permite evitar usar la funcionalidad nativa del dispositivo y eso conlleva una mayor facilidad para crear aplicaciones PWA ya que usa sus propios plugins. Es importante tener en cuenta que debemos ajustarnos a nuestras necesidades ya que el uso de elementos nativos del dispositivo ofrece una mejor y más rápida respuesta.

Otro punto a tener en cuenta es destacar que en la actualidad existen muchas APIs que podemos usar en nuestros desarrollos como es el caso de 'Google Places API for Web'. Estas APIs nos facilitan información ya validada, lo que nos permite liberar de la mayor carga de trabajo posible al usuario, como ocurre en nuestro caso, y también se simplifica la labor de verificación de datos durante el desarrollo. Además, debemos mencionar que el registro es mucho más sencillo usando los sistemas de terceros como son Google, Facebook, Twitter… De esta manera sabemos que, gracias a sus propias validaciones, los usuarios no se registrarán con mails de spam ya que se requiere verificación mediante el número del teléfono móvil.

Por último, se debe tener en cuenta que en el desarrollo de aplicaciones además de ofrecer los servicios solicitados por los usuarios se debe contemplar también el realizar un seguimiento del uso por parte de los usuarios viendo así que funcionalidades son las que más se usan para mejorar el éxito de la aplicación. Como ejemplo también podemos, en el caso de incluir anuncios en la aplicación, contabilizar el número de entradas realizadas a esos anuncios para ver el éxito de las distintas campañas publicitarias que se pueden usar para financiar nuestra aplicación.

Como resumen a todo lo anterior, debo decir que ha sido una experiencia muy enriquecedora el analizar y el trabajar con todas las posibilidades que los servicios de terceros, la web y la comunidad ofrecen. Todos esos recursos están a nuestro alcance y, en muchas ocasiones, nos olvidamos de que existen y de que están para facilitarnos la labor diaria.

8.2 Trabajo futuro

Las líneas de trabajo futuro que se han propuesto para el proyecto se pueden esperar en tres áreas: implementar mejoras de usabilidad y funcionalidad, mejora del sistema de pruebas y migración de la aplicación a PWA.

Sobre la implementación de las mejoras de usabilidad y funcionalidad queda pendiente estudiar la viabilidad e implementación de caché de datos, ofrecer la posibilidad de acceder a la aplicación sin necesidad de estar conectado a internet, mejorar los tiempos de respuesta, implementación de un apartado de ferias y eventos de interés añadiendo notificaciones push para avisar de estos eventos. Además, mejorar el sistema de registro de datos manual, realizando un registro automático sobre la API de Google. En nuestro caso no deberían existir problemas con el límite de peticiones diarias que se pueden realizar de manera gratuita a la API (1000 peticiones), ya que la mayor parte de la carga de las llamadas a la API se realizaría en el primer periodo de vida de la APP, posteriormente usaríamos nuestros propios recursos almacenados en nuestra base de datos. También se realizará el desarrollo de una página web, servicios backoffice para facilitar la administración por parte de los administradores y la implementación de los servicios en el servidor QA.

Aunque se han usado plugins compatibles para Android e IOS las mejoras del sistema de pruebas vienen dadas por el uso de emuladores apropiados para IOS verificando la funcionalidad mediante el uso de condiciones reales.

El último punto para tener en cuenta es la migración de la APP a un proyecto PWA modificando distintas funcionalidades dadas por los plugins nativos ofrecidos por Apache Cordova con el uso de funcionalidad web o el uso de Capacitor, un nuevo puente nativo para APPs webs que nos proporciona sus propios plugins, aún en fase Alpha.

Referencias

[1] Google/TNS, Google Consumer Barometer, 2017.
Recuperado de https://www.thinkwithgoogle.com/intl/es-es/canales-de-publicidad/movil/estudio-de-google-consumer-barometer-2017-el-%C3%B1o-de-los-m%C3%B3viles/ - Último acceso: 10/06/2018

[2] Desarrollo de aplicaciones móviles
Recuperado de http://estudiowam.com/desarrollo-de-aplicaciones-moviles/ - Último acceso: 10/06/2018

[3] Android Studio
Recuperado de https://developer.android.com/studio/index.html?hl=es-419 - Último acceso: 10/06/2018

[4] Notepad++
Recuperado de https://notepad-plus-plus.org/download/v7.5.6.html - Último acceso: 10/06/2018

[5]Angular
Recuperado de https://angular.io/ - Último acceso: 10/06/2018

[6] Ionic
Recuperado de https://openwebinars.net/blog/ionic-framework-que-es/ - Último acceso: 10/06/2018

[7] Consola de desarrolladores de google
Recuperado de https://developers.google.com/maps/?hl=es-419 - Último acceso: 10/06/2018

[8] MongoDB Atlas
Recuperado de https://docs.atlas.mongodb.com/ - Último acceso: 10/06/2018

[9] Como elegir tu base de datos: Cloud Firestore o Realtime Database
Recuperado de https://firebase.google.com/docs/firestore/rtdb-vs-firestore?hl=es-419 - Último acceso: 10/06/2018

[10] ¿Qué es una metodología ágil?
Recuperado de https://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software - Último acceso: 10/06/2018

[11] ¿Cuáles son las metodologías ágiles más usadas?
Recuperado de https://blog.conectart.com/metodologias-agiles/ - Último acceso: 10/06/2018

[12] Comparativa de patrones de arquitectura
Recuperado de https://ingsoftwarei2014.wordpress.com/category/comparacion-de-los-patrones-de-arquitectura-mvc-mv-vm-mvp/ - Último acceso: 10/06/2018

[13] Facebook developers
Recuperado de https://developers.facebook.com/ - Último acceso: 10/06/2018

Glosario

API	Application Programming Interface
PWA	Progressive Web Apps
DOM	Document Object Model
XML	eXtensible Markup Language
IDE	Entorno de Desarrollo Integrado
MVC	Model View Controler
JSON	JavaScript Object Notation
SASS	Syntatically Awesome Styleheets
HTML	HyperText Markup Language

Anexos

A. Manual de usuario

Tapeoapp

Junio 2018
Version 1.0

Manual de Usuario

1. INTRODUCCIÓN

1.1 ¿Qué es Tapeoapp?

Tapeoapp es aplicación multiplataforma que nos dará información en tiempo real, trabajando directamente con la nube, sobre lugares donde podremos interaccionar socialmente conociendo de antemano información variada. La información que nos proporcionará de cada sitio mostrará los datos de ubicación del local, una galería de fotos subidas por los usuarios, los comentarios de estos y una puntuación general.

1.2 ¿Como acceder a Tapeoapp?

Para accede a Tapeoapp necesitas:

- Un smartphone o tablet.
  - Android XX.
  - IOS XX.

- Acceso a través del icono de la APP.

USUARIO Y CONTRASEÑA
El Sistema reconoce tus credenciales de Google y Facebook para logarse en la APP, no necesita crear un perfil nuevo con usuario/contraseña.

ACCESO
Tapeoapp no requiere estar registrado en la aplicación para poder disfrutar de ella. El estar dado de alta ofrece mayor funcionalidad.

2. Empezando

2.1 Sistema de Módulos

Tapeoapp tiene diferentes módulos. Dependiendo de si el usuario se encuentra logado se le mostrarán los módulos disponibles.

- **Locales**: desde este módulo el usuario tendrá disponible:
  - Ver todos los locales registrados filtrados previamente.
  - Registrar un nuevo local.
  - Buscar locales por localidad y nombre dentro de esa localidad.

- **Mapa:** desde este módulo el usuario tendrá disponible:
  - Ver todos los locales registrados en la aplicación cercanos a la localidad en la que se encuentre el usuario.

- **Favoritos:** desde este módulo el usuario tendrá disponible:
  - Ver todos los locales registrados como favoritos por parte del usuario previamente registrado en la APP.

- **Idiomas:** desde este módulo el usuario tendrá disponible:
  - Cambiar el idioma en el que se muestran los mensajes por parte de la aplicación, se han implementados los paquetes de idiomas en inglés y castellano.

2.2 Seguridad

Básicamente Tapeoapp se divide en 3 roles:

Rol	Permisos
Usuario del sistema	• Registrar nuevos locales. • Ver todos los establecimientos registrados filtrados previamente por localidad. • Consultar el mapa. • Cambiar idioma.
Usuario registrado	• Todos los permisos de usuario del Sistema. • Ver locales favoritos. • Añadir comentario y valoración de los locales.
Administrador	• Acceso completo a la APP. • Administración desde la base de datos.

2.3 El proceso

El proceso de negocio que cubre el Sistema Tapeoapp se puede resumir en los siguientes pasos:

- Un nuevo local se registra en la aplicación. El usuario puede añadir valoración y comentarios desde el registro de local en caso de estar registrado en la APP.

- Dependiendo de si el usuario está dado de alta en la APP podrá modificar su comentario y valoración en caso de ya estar registrada la información o añadir esos campos.

3. Menú lateral

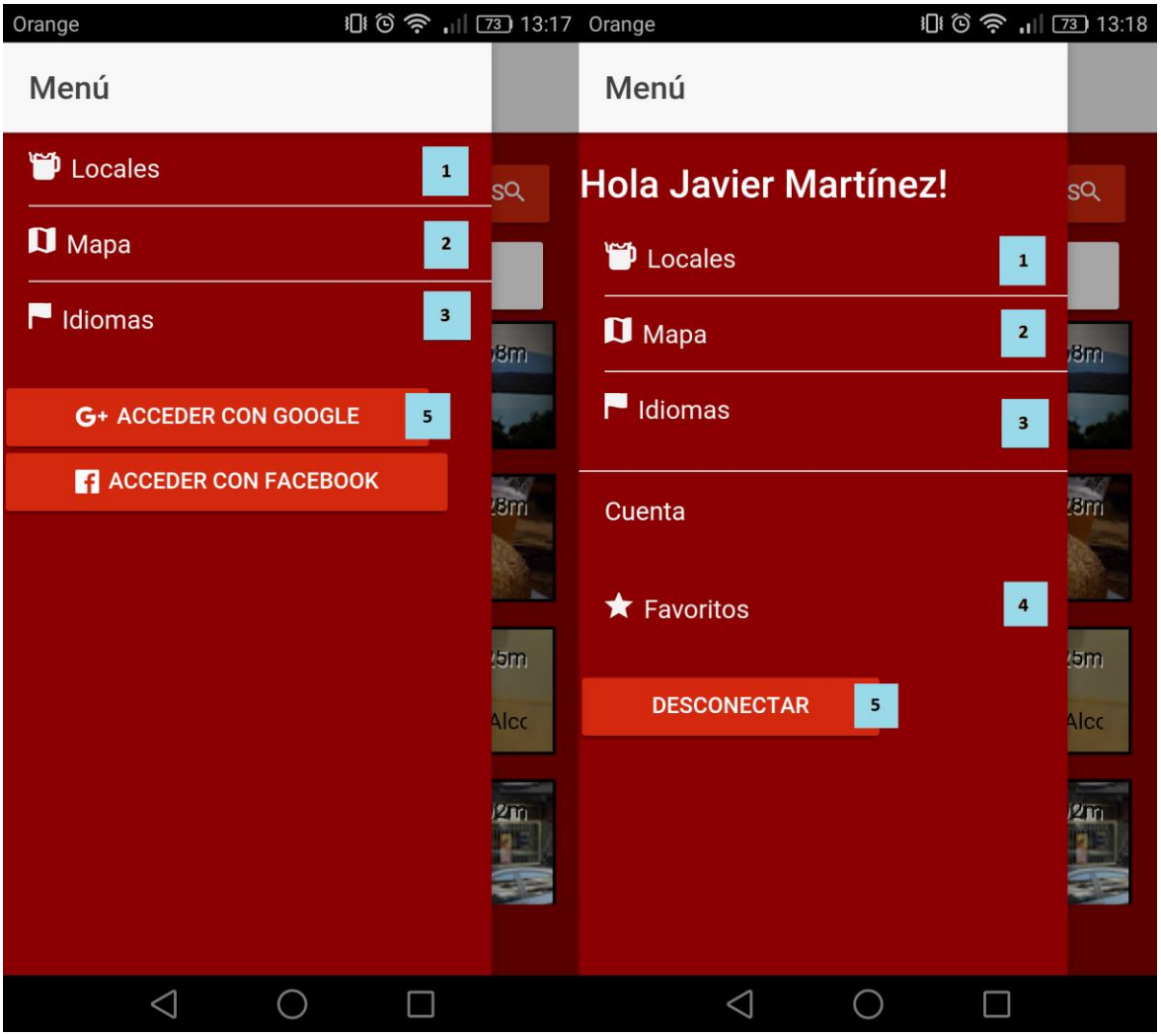


Figura Anexo A 3-1: Menú lateral

1	**Locales** Esta opción nos lleva a la vista de locales.
2	**Mapa** Esta opción nos lleva a la vista de mapa.
3	**Idiomas** Esta opción nos lleva a la vista de Idiomas.
4	**Favoritos** Esta opción nos lleva a la vista de favoritos.
5	**Acceso/Desconexión** Esta opción nos permite acceder al sistema usando datos de registro.

4. Locales

4.1 Búsqueda de locales

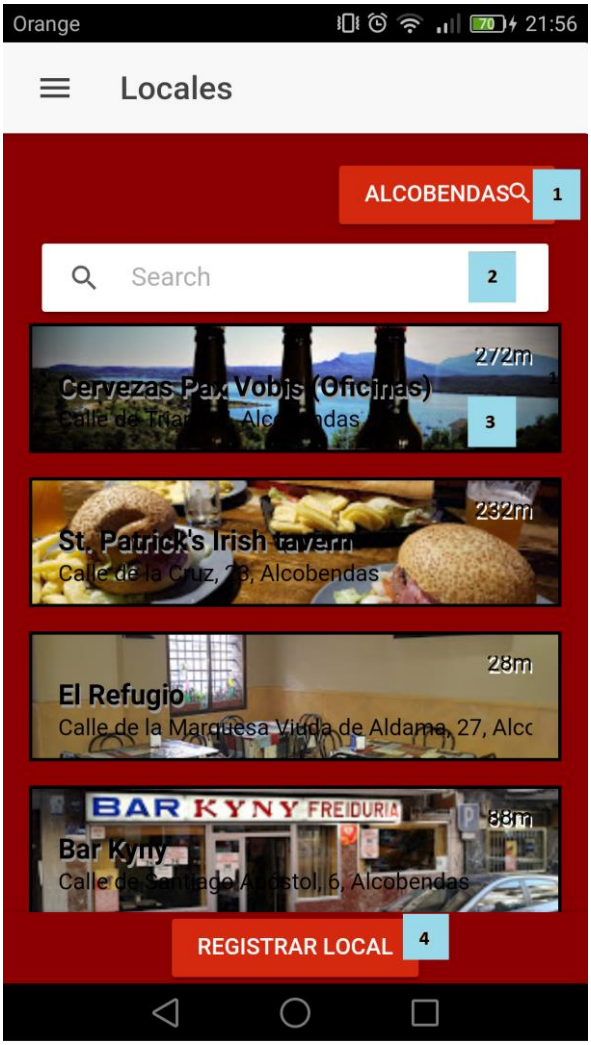


Figura Anexo A 4-1: Pantalla locales

1	**Búsqueda por localidad** Esta opción nos permite buscar locales por su localidad ajustando la información a la vista actual.
2	**Búsqueda por nombre** Esta opción nos permite buscar locales por su nombre sobre la vista actual.
3	**Locales** El Sistema nos ofrece un resumen de todos los locales registrados en la APP de la localidad seleccionada. Si seleccionamos alguno de los locales nos muestra la pantalla de información de este.

4	**Nuevo local** Esta opción lanza un nuevo listado de locales cercanos para registrar en la APP.
5	**Provincias** Esta opción nos permite seleccionar sobre qué provincia queremos mostrar las localidades.
6	**Localidades** Esta opción nos permite seleccionar sobre qué localidad queremos mostrar los locales.

Para cada local el sistema nos mostrará la siguiente información:

Figura Anexo A 4-2: Botón de local

- Distancia en metros.
- Nombre.
- Dirección.

Para filtrar por la localidad se deben seguir los siguientes pasos:

- Seleccionar una provincia del listado.
- Seleccionar una localidad del listado.
- Solo aparecerán provincias y localidades de locales registrados en nuestra aplicación.

4.2 Nuevo local

Para registrar un nuevo local se deben seguir los siguientes pasos:

- Ir a la ventana 'Locales'.
- Hacer clic en **REGISTRAR LOCAL**. El Sistema cargará un nuevo listado de locales no registrados en la aplicación a una distancia de 1000 metros desde la ubicación actual. Tras seleccionar una de las opciones el sistema cargará una nueva vista con información detallada del local.

Campos a rellenar:

- Horario: en caso de no cargarse el horario de manera automática durante el registro los usuarios podrán rellenar los campos.
- Valoración: solo disponible para usuarios registrados en la APP.
- Comentario: solo disponible para usuarios registrados en la APP.

Tanto el campo valoración como el campo comentario se registran de manera conjunto, ambos deben contener información para ser almacenados en el sistema.

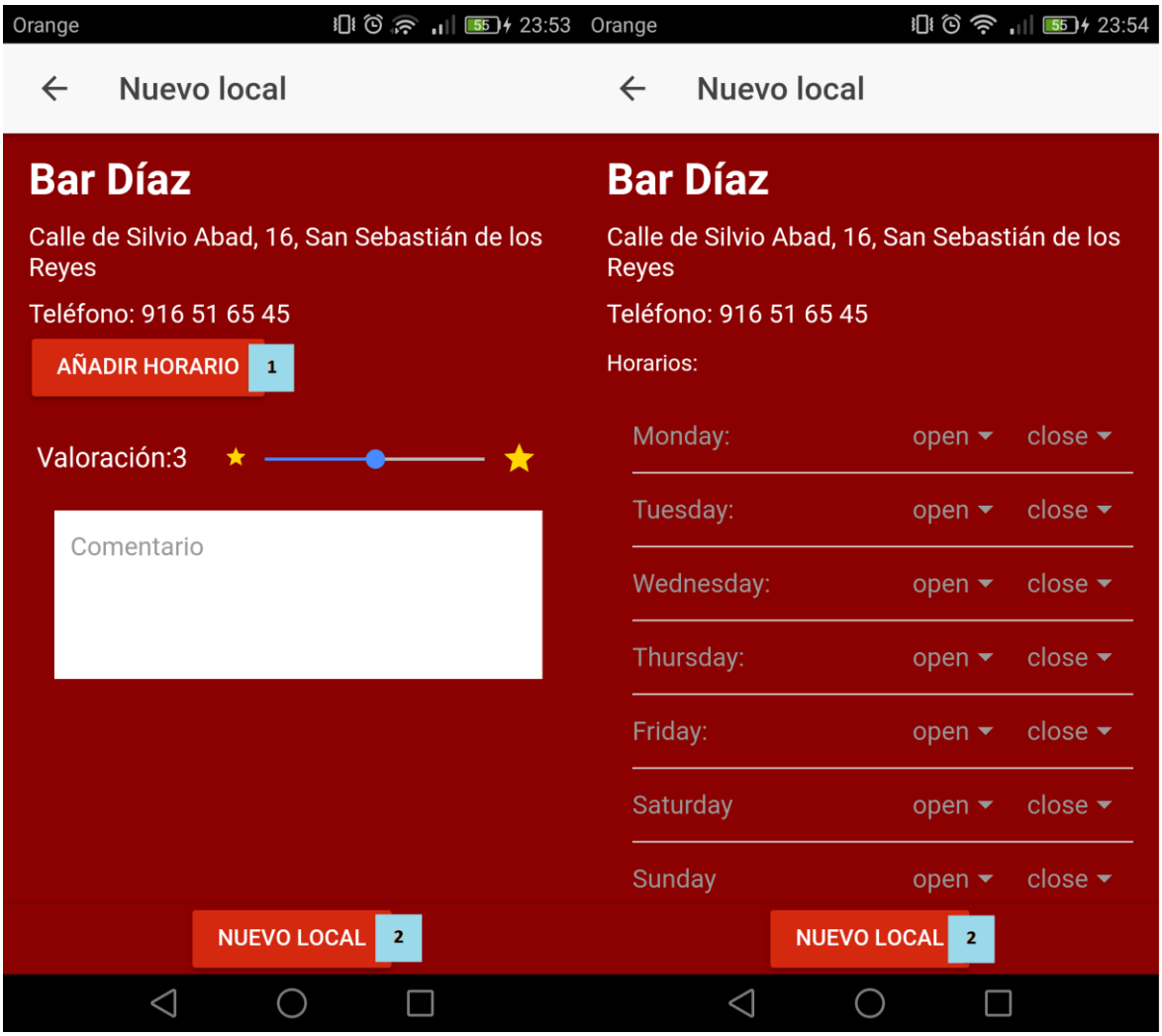


Figura Anexo A 4-3: Registro de nuevo local sin horario

1	**Añadir nuevo horario** Esta opción nos permite añadir un horario de apertura y cierre del local en caso de no estar añadido.
2	**Nuevo local** El sistema completará el registro de los datos.

Si el horario ya se encuentra registrado en la aplicación se muestra de la siguiente manera:

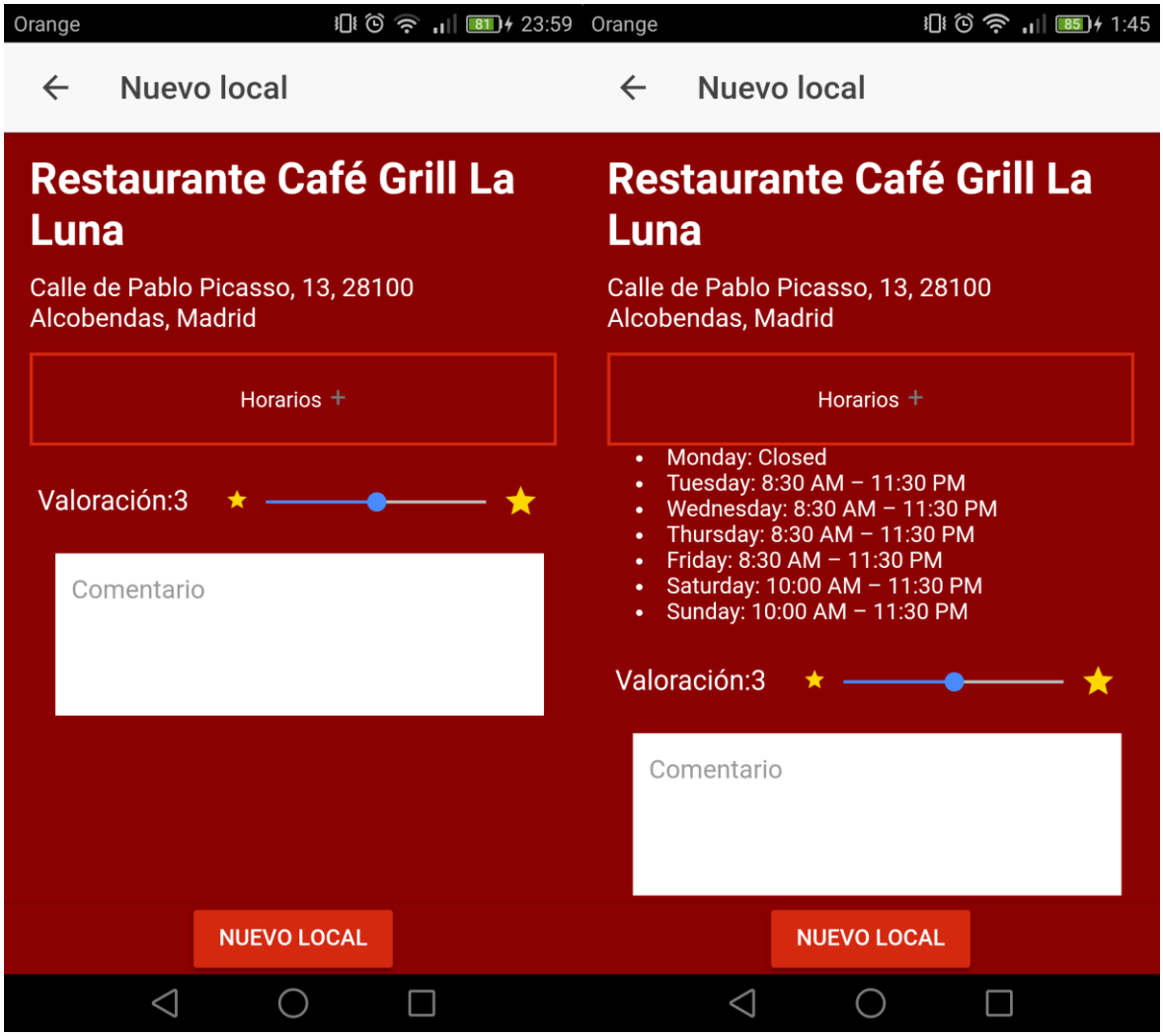


Figura Anexo A 4-4: Registro de nuevo local con horario

En este punto se completará el registro del local en el sistema con la información facilitada.

4.3 Detalles del local

Una vez registrado el local el Sistema genera una vista de detalle.

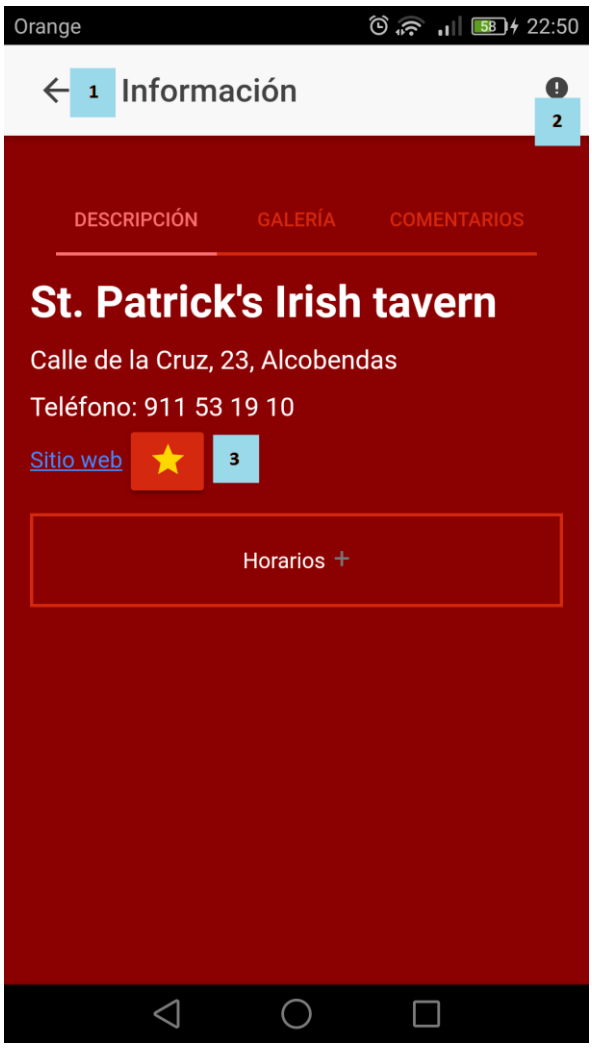


Figura Anexo A 4-5: Pantalla de detalles de un local

1	**Volver a locales** Esta opción nos permite volver a la lista de locales.
2	**Reportar incidencia** El sistema permite registrar una incidencia sobre el local.
3	**Sitio web y favorito** Desde el sistema se podrá acceder a la web del local y marcar el mismo como favorito

En esta vista podemos ver los siguientes campos:

Figura Anexo A 4-6: Opciones de la pantalla detalles

- Descripción: nos muestra los datos del local e información general.
- Galería: nos muestra las imágenes compartidas por los usuarios.
- Comentarios: nos muestra todos los comentarios y valoraciones sobre el local registrados en la aplicación, además permite editar nuestro propio comentario y valoración.

4.3.1 Reporte de errores

Figura Anexo A 4-7: Vista de reporte de errores

El usuario podrá reportar errores sobre el contenido de los locales, deberá seleccionar uno de los motivos que ofrece el sistema, tales como información incorrecta, imágenes inapropiadas… Además, podrá añadir un comentario sobre la incidencia notificada.

4.3.2 Descripción

En la sección descripción el Sistema nos muestra los siguientes campos:

- Nombre.
- Dirección.
- Valoración media.
- Botón de favorito marcado con una estrella.
- Horario: en caso de no estar este registrado permitirá añadirlo.

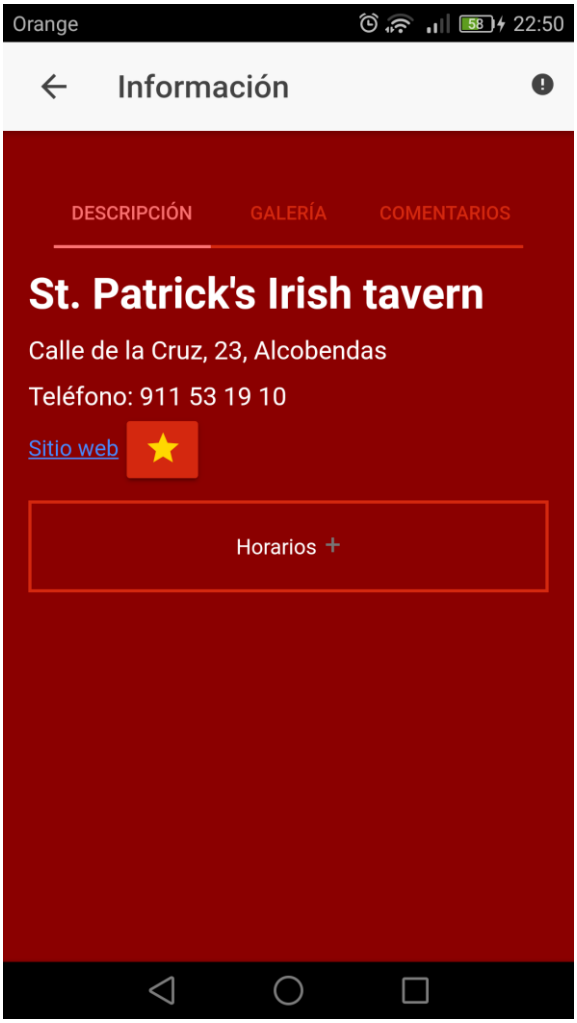


Figura Anexo A 4-8: Vista de descripción de detalles

4.3.3 Galería

En la sección galería cualquier usuario del sistema puede ver las imágenes subidas por el resto de los usuarios. Únicamente los usuarios dados de alta en el sistema pueden subir nuevas imágenes, siendo estas cargadas desde la galería del dispositivo o la propia cámara.

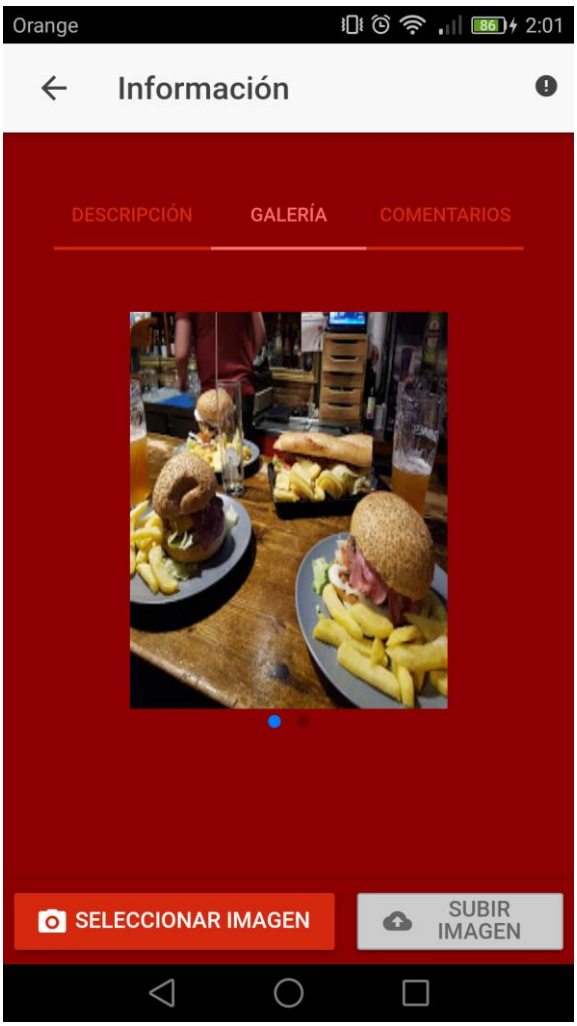


Figura Anexo A 4-9: Vista de galería en detalles

4.3.4 Comentarios

En la sección comentarios cualquier usuario del sistema puede leer los comentarios y valoraciones del resto de usuarios. Solo los que se encuentren dados de alta en el sistema pueden añadir su comentario y valoración y editar estos campos.

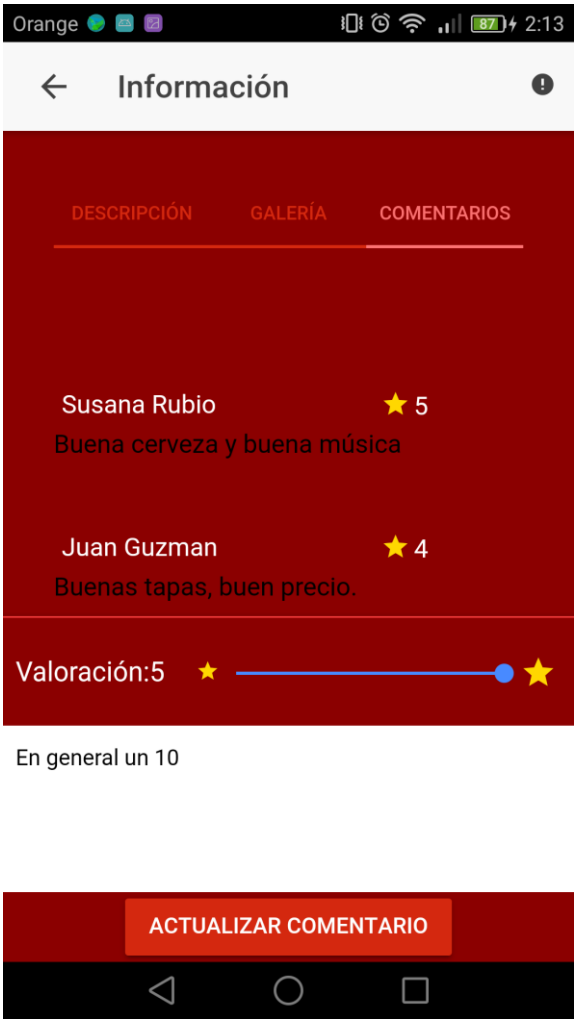


Figura Anexo A 4-10: Vista de comentarios en detalles

Para añadir o modificar un comentario:

- Ir a comentarios.
- Añadir tu valoración.
- Añadir tu comentario.
- Hacer clic **AÑADIR COMENTARIO** o **ACTUALIZAR COMENTARIO**.

5. Mapa

Desde MAPA el usuario puede ver sobre un mapa todos los locales registrados en el sistema, la lista de locales que se muestran es de la localidad en la cual se encuentra el usuario.

Se distinguen tres tipos de iconos en el mapa, el primero marca la ubicación actual y se muestra como una flecha de color verde. En segundo lugar, se muestra como una estrella los locales registrados como favoritos y por último se marcan el resto de los locales registrados con un icono de color rojo. Si hacemos clic sobre los iconos nos muestra el nombre del local.

Figura Anexo A 5-1: Pantalla mapa

6. Idiomas

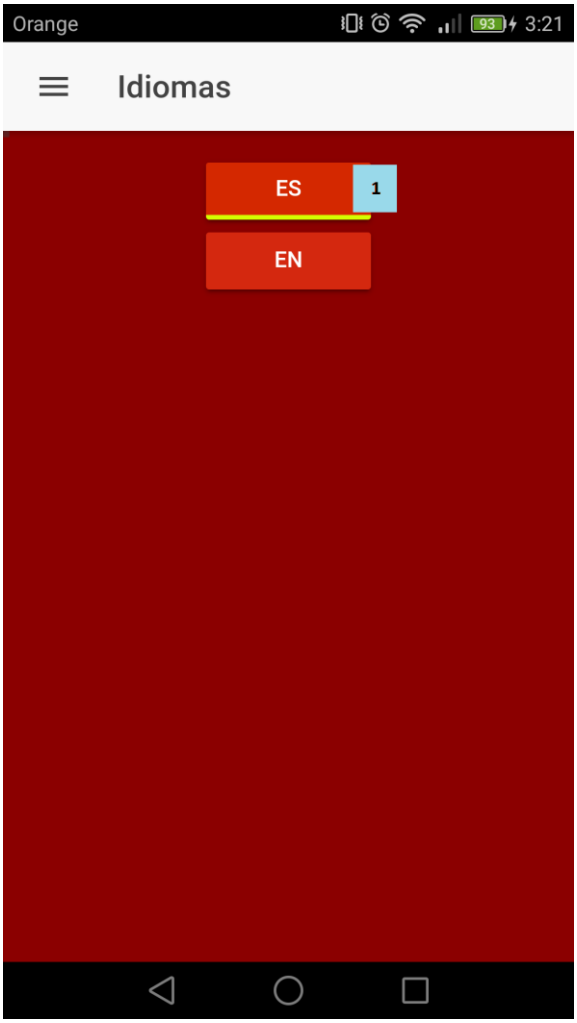


Figura Anexo A 6-1: Pantalla idiomas

1	**Selección de idioma** Esta opción nos permite cambiar el idioma de la aplicación entre las opciones que se muestran en la pantalla.

7. Favoritos

Desde FAVORITOS el usuario puede ver un listado con todos los locales que ha seleccionado previamente como favoritos. Desde cada uno podrá acceder a la vista de información.

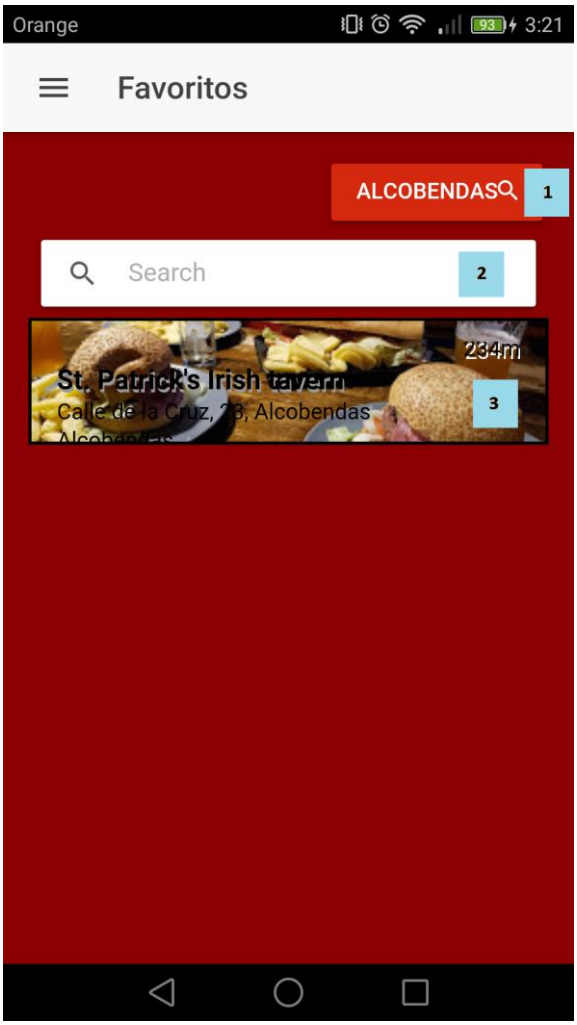


Figura Anexo A 7-1: Pantalla favoritos

1	**Búsqueda por localidad** Esta opción nos permite buscar locales por su localidad ajustando la información a la vista actual.

2	**Búsqueda por nombre** Esta opción nos permite buscar locales por su nombre sobre la vista actual.
3	**Locales** El Sistema nos ofrece un resumen de todos los locales registrados en la APP de la localidad seleccionada que hemos solicitado previamente como favoritos. Si seleccionamos alguno de los locales nos muestra la pantalla de información de este.

B. Plan de pruebas funcionales

Tapeoapp

Planes de Pruebas Funcionales

Versión: 0100

Fecha: 08/06/2018

HOJA DE CONTROL

Proyecto	Tapeoapp		
Entregable	Planes de Pruebas Funcionales		
Versión/Edición	0100	**Fecha Versión**	08/06/2018
Aprobado por	Javier Martínez Hernández	**Fecha Aprobación**	13/06/2018
		Nº Total de Páginas	10

REGISTRO DE CAMBIOS

Versión doc	Causa del Cambio	Responsable del Cambio	Fecha del Cambio
0100	Versión inicial	Javier Martínez Hernández	08/06/2018

CONTROL DE DISTRIBUCIÓN

Nombre y Apellidos
Javier Martínez Hernández

1 INTRODUCCIÓN

1.1 Objeto

El objetivo de este documento es recoger los casos de pruebas que verifican que el sistema satisface los requisitos especificados. Deberá contener la definición de los casos de prueba, la matriz de trazabilidad entre casos de pruebas y requisitos, y la estrategia a seguir en la ejecución de las pruebas.

1.2 Alcance

Este documento va dirigido a validar el buen funcionamiento de las distintas funcionalidades que se deben realizar en la aplicación mediante la realización de pruebas unitarias de cada uno de los módulos.

Estas pruebas unitarias se han realizado durante el desarrollo de cada módulo, añadiéndose a la aplicación tras realizar su correspondiente validación.

2 TRAZABILIDAD DE CASOS DE PRUEBAS – REQUISITOS

En este apartado se cumplimenta una matriz donde se indica la correspondencia entre los casos de prueba definidos y los requisitos funcionales especificados en la sección de requisitos. Las filas representan cada uno de los casos de pruebas definidos y las columnas los requisitos funcionales.

	RF-001	RF-002	RF-003	RF-004	RF-005	RF-006	RF-007	RF-008	RF-009	RF-010	RF-011
<CP-1>					X						X
<CP-2>	X	X									X
<CP-3>	X	X									X
<CP-4>		X		X							X
<CP-5>		X	X	X	X						X
<CP-6>	X	X	X	X							X
<CP-7>											X
<CP-8>	X	X	X	X			X				X
<CP-9>	X	X	X	X				X			X
<CP-10>	X	X	X	X		X					
<CP-11>	X	X	X	X						X	X
<CP-12>									X		

3 DEFINICIÓN DE LOS CASOS DE PRUEBAS

Cargar local usando la API de Google	**<CP-1>**	
	¿Prueba de despliegue?	**Si/No**
Descripción: En este caso de prueba se valida la correcta petición y respuesta de datos realizada a la API Google Places.		
Prerrequisitos • El usuario puede o no estar dado de alta en la aplicación. • Se debe tener conexión a internet. • Debe estar habilitado el geoposicionamiento del dispositivo.		
Pasos: 1. Cargar la aplicación. 2. Entrar en la vista locales. 3. Hacer click en el botón Registrar local. 4. Hacer click en cualquiera de los locales mostrados en el listado.		
Resultado esperado: • Listado de locales con información general. • Detalles del local seleccionado.		
Resultado obtenido: • Listado de locales con información general. • Detalles del local seleccionado.		

Registro y validación en el sistema con Google	**<CP-2>**	
	¿Prueba de despliegue?	**Si/No**
Descripción: En este caso de prueba se valida la correcta conexión a la aplicación mediante el uso de una cuenta de mail de Google.		
Prerrequisitos • El usuario no estar dado de alta en la aplicación. • Se debe tener conexión a internet.		
Pasos: 1. Cargar la aplicación. 2. Entrar en el menú lateral. 3. Hacer click en el botón 'Acceso con Google'. 4. Introducir credenciales de Google en la nueva ventana generada. 5. Completar validación		
Resultado esperado: • Cargar nueva ventana para introducir las credenciales de Google, mail y contraseña. • Conexión correcta del sistema, mostrar nombre del usuario y habilitar botón de desconexión.		
Resultado obtenido: • Cargar nueva ventana para introducir las credenciales de Google, mail y contraseña. • Conexión correcta del sistema, mostrar nombre del usuario y habilitar botón de desconexión.		

Registro y validación en el sistema con Facebook	**<CP-3>**	

	¿Prueba de despliegue?	Si/No
Descripción: En este caso de prueba se valida la correcta conexión a la aplicación mediante el uso de una cuenta Facebook.		
Prerrequisitos • El usuario no estar dado de alta en la aplicación. • Se debe tener conexión a internet.		
Pasos: 1. Cargar la aplicación. 2. Entrar en el menú lateral. 3. Hacer click en el botón 'Acceso con Facebook. 4. Introducir credenciales de Facebook en la nueva ventana generada. 5. Completar validación		
Resultado esperado: • Cargar nueva ventana para introducir las credenciales de Facebook, mail y contraseña. • Conexión correcta del sistema, mostrar nombre del usuario y habilitar botón de desconexión.		
Resultado obtenido: • Cargar nueva ventana para introducir las credenciales de Facebook, mail y contraseña. • Conexión correcta del sistema, mostrar nombre del usuario y habilitar botón de desconexión.		

Conexión a la base de datos (Lectura)	**<CP-4>**	
	¿Prueba de despliegue?	**Si/No**
Descripción: En este caso de prueba se valida la correcta conexión a la base de datos capturando información en la misma.		
Prerrequisitos • El usuario puede o no estar dado de alta en la aplicación. • Se debe tener conexión a internet. • Debe estar habilitado el geoposicionamiento del dispositivo.		
Pasos: 1. Cargar la aplicación. 2. Entrar en el menú locales. 3. Hacer click en el cualquiera de los locales del listado.		
Resultado esperado: • Obtener detalles del local seleccionado.		
Resultado obtenido: • Detalles del local seleccionado.		

Conexión a la base de datos (Escritura/lectura)	**<CP-5>**	
	¿Prueba de despliegue?	**Si/No**
Descripción: En este caso de prueba se valida la correcta conexión a la base de datos capturando e insertando información en la misma.		
Prerrequisitos • El usuario debe estar dado de alta en la aplicación. • Se debe tener conexión a internet.		

- Debe estar habilitado el geoposicionamiento del dispositivo.

Pasos:
1. Cargar la aplicación.
2. Entrar en el menú locales.
3. Hacer click en el botón Registrar local.
4. Hacer click en cualquiera de los locales mostrados en el listado.
5. Insertar información extra como horario, valoración y comentario.

Resultado esperado:
- Obtener nuevo listado de locales en la vista local con el nuevo registro.

Resultado obtenido:
- Obtener nuevo listado de locales en la vista local con el nuevo registro.

Registro manual de locales	**<CP-6>**	
	¿Prueba de despliegue?	**Si/No**

Descripción:
En este caso de prueba se valida la correcta conexión a la base de datos insertando información de manera manual en la misma por usuarios dados de alta y no estar disponible hasta la validación de esta por parte de los administradores.

Prerrequisitos
- El usuario debe estar dado de alta en la aplicación.
- Se debe tener conexión a internet.
- Debe estar habilitado el geoposicionamiento del dispositivo.

Pasos:
1. Cargar la aplicación.
2. Entrar en el menú locales.
3. Hacer click en el botón Registrar local.
4. Hacer click en el botón 'Registro manual'.
5. Añadir información solicitada en la nueva vista.
6. Completar registro de información.

Resultado esperado:
- Obtener nuevo un nuevo acceso en la base de datos no visible desde la vista locales.

Resultado obtenido:
- Obtener nuevo un nuevo acceso en la base de datos no visible desde la vista locales.

Conexión sin internet	**<CP-7>**	
	¿Prueba de despliegue?	**Si/No**

Descripción:
En este caso de prueba se valida la notificación en caso de no estar conectado a internet al inicializar la aplicación.

Prerrequisitos
- El usuario debe o no estar dado de alta en la aplicación.
- No se debe tener conexión a internet.
- Debe estar habilitado el geoposicionamiento del dispositivo.

Pasos:
1. Cargar la aplicación.

2. Entrar en el menú locales.

Resultado esperado:
- Notificación de falta de conectividad.

Resultado obtenido:
- Notificación de falta de conectividad.

Registro de comentarios	**<CP-8>**	
	¿Prueba de despliegue?	**Si/No**

Descripción:
En este caso de prueba se valida la correcta conexión a la base de datos insertando un nuevo comentario o actualización de uno ya existente además de incluir una valoración.

Prerrequisitos
- El usuario debe estar dado de alta en la aplicación.
- Se debe tener conexión a internet.
- Debe estar habilitado el geoposicionamiento del dispositivo.

Pasos:
1. Cargar la aplicación.
2. Entrar en el menú locales.
3. Hacer click en el cualquiera de los locales del listado.
4. Obtener detalles del establecimiento.
5. Acceder a la pestaña comentarios.
6. Añadir información solicitada en la nueva vista.
7. Completar registro de información.

Resultado esperado:
- Nuevo comentario o actualización añadido al local seleccionado.

Resultado obtenido:
- Nuevo comentario o actualización añadido al local seleccionado.

Registro en galería	**<CP-9>**	
	¿Prueba de despliegue?	**Si/No**

Descripción:
En este caso de prueba se valida la correcta conexión a la base de datos capturando e insertando nuevas imágenes en la misma.

Prerrequisitos
- El usuario debe estar dado de alta en la aplicación.
- Se debe tener conexión a internet.
- Debe estar habilitado el geoposicionamiento del dispositivo.

Pasos:
1. Cargar la aplicación.
2. Entrar en el menú locales.
3. Hacer click en el cualquiera de los locales del listado.
4. Obtener detalles del establecimiento.
5. Acceder a la pestaña galería.
6. Hacer click en seleccionar imagen y cargar imagen de la cámara o galería.
7. Completar registro haciendo click en subir imagen.

Resultado esperado:

- Actualizar la galería de imágenes y añadir nuevo registro en la base de datos.

Resultado obtenido:
- Actualizar la galería de imágenes y añadir nuevo registro en la base de datos.

Registro de favorito	**<CP-10>**	
	¿Prueba de despliegue?	**Si/No**

Descripción:
En este caso de prueba se valida la correcta conexión a la base de datos registrando un local como favorito en la misma.

Prerrequisitos
- El usuario debe estar dado de alta en la aplicación.
- Se debe tener conexión a internet.
- Debe estar habilitado el geoposicionamiento del dispositivo.

Pasos:
1. Cargar la aplicación.
2. Entrar en el menú locales.
3. Hacer click en el cualquiera de los locales del listado.
4. Obtener detalles del establecimiento.
5. Acceder a la pestaña descripción.
6. Hacer click en botón favoritos.

Resultado esperado:
- Actualizar la lista de locales favoritos y añadir registro de favorito en el local seleccionado por el usuario.

Resultado obtenido:
- Actualizar la lista de locales favoritos y añadir registro de favorito en el local seleccionado por el usuario.

Reporte de errores	**<CP-11>**	
	¿Prueba de despliegue?	**Si/No**

Descripción:
En este caso de prueba se valida la correcta conexión a la base de datos registrando un reporte de error en la misma.

Prerrequisitos
- El usuario debe estar dado de alta en la aplicación.
- Se debe tener conexión a internet.
- Debe estar habilitado el geoposicionamiento del dispositivo.

Pasos:
1. Cargar la aplicación.
2. Entrar en el menú locales.
3. Hacer click en el cualquiera de los locales del listado.
4. Obtener detalles del establecimiento.
5. Hacer click en el botón de reporte.
6. Introducir el tipo de incidencia y comentario.
7. Guardar reporte.

Resultado esperado:
- Creación de nuevo registro en la base de datos en errores.

Resultado obtenido:

- Creación de nuevo registro en la base de datos en errores.

Compatibilidad con Android	**<CP-12>**	
	¿Prueba de despliegue?	**Si/No**
Descripción: En este caso de prueba se valida el correcto funcionamiento de la aplicación en Android.		
Prerrequisitos • El usuario debe estar dado de alta en la aplicación. • Se debe tener conexión a internet. • Debe estar habilitado el geoposicionamiento del dispositivo.		
Pasos: 1. Cargar la aplicación. 2. Entrar en el menú locales. 3. Validar todos los casos de prueba anteriores.		
Resultado esperado: • Resultado esperado de los casos de prueba indicados.		
Resultado obtenido: • Resultado esperado de los casos de prueba indicados.		

4 ESTRATEGIA DE EJECUCIÓN DE PRUEBAS

En este apartado se cumplimenta una matriz donde se indica la correspondencia entre los casos de prueba definidos y los ciclos en los cuales se implementan. Las filas representan cada uno de los casos de pruebas definidos y las columnas los ciclos en los cuales se han ejecutados las pruebas.

	<Ciclo 1>	**<Ciclo 2>**	**<Ciclo 3>**	**<Ciclo 4>**	**<Ciclo 5>**	**<Ciclo 6>**
<CP-1>	**X**	**X**	**X**	**X**	**X**	
<CP-2>	**X**	**X**	**X**	**X**	**X**	
<CP-3>	**X**	**X**	**X**	**X**	**X**	
<CP-4>		**X**	**X**	**X**	**X**	
<CP-5>		**X**				
<CP-6>				**X**		
<CP-7>						**X**
<CP-8>			**X**			
<CP-9>			**X**			
<CP-10>					**X**	
<CP-11>					**X**	**X**
<CP-12>	**X**	**X**	**X**	**X**	**X**	**X**

C. Presupuesto

A continuación, se muestra es presupuesto calculado en función de la estimación en jornadas de cada una de las partes en las que se divide el desarrollo del sistema.

	Actividad	Responsable	Jornada
1.- ANÁLISIS Y DISEÑO			
1.1	Creación BBDD	Desarrollo	2
1.2	Módelo de datos	Diseño	2
1.3	Creación de Línea Gráfica Final	Diseño	2
2.- DESARROLLO APP			
2.1	Desarrollo Esqueleto de aplicación	Tecnología	2
2.2	Navegación General incluyendo creación Detalle	Desarrollo	3
2.3	Seguridad Google & FB	Desarrollo	3
2.4	Buscador de Locales	Desarrollo	1
2.5	Nuevos locales	Desarrollo	5
2.6	Detalle Local - Galería	Desarrollo	2
2.7	Detalle Local - Descripción	Desarrollo	1
2.8	Detalle Local - Comentarios	Desarrollo	1
2.9	Favoritos	Desarrollo	1
2.10	Mapa	Desarrollo	1
2.11	Idiomas	Desarrollo	1
2.12	Integración API Google places	Desarrollo	2
2.13	Integración Diseño Gráfico	Diseño	7
3.- DESARROLLO WEB			
3.1	Web Pública	Desarrollo	1
3.2	Back Office	Desarrollo	5
3.3	Seguridad	Desarrollo	2
3.4	Integración Diseño Gráfico	Diseño	3
4.- SERVICIOS DE INTERCONEXIÓN			
4.1	Tracking de Usuarios	Desarrollo	4
4.2	Seguridad	Desarrollo	2
4.3	Perfil	Desarrollo	1
5.- PRUEBAS			
5.1	Pruebas de integración y carga	Desarrollo	3
5.2	Configuración	Desarrollo	2
5.3	Instalación en entorno	Desarrollo	1
5.4	QA	Desarrollo	14
5.5	Pruebas y Resolución de incidencias	Diseño	5
6.- PASO A PRODUCCIÓN			
6.1	Instalación en entorno QA	Desarrollo	1
6.2	Instalación en entorno Producción	Desarrollo	1
7.- DOCUMENTACIÓN			
7.1	Manual de Usuario	Desarrollo	1
7.2	Manual de Instalación	Desarrollo	2
7.3	Documentación General	Desarrollo	5

Figura Anexo C 4-1: Desglose en jornadas del proyecto

El desglose indicado anteriormente incluye toda la realización del sistema actual y partes de trabajo futuro, estando orientado a un presupuesto real. En función de los costes del personal implicado en el desarrollo del proyecto que se desglosan a continuación se ha

realizado el coste total del presupuesto. Estos costes de personal y coste medio de la jornada se han obtenido de la empresa I-NERCYA INTELLIGENT SOFTWARE.

Puesto	Precio Hora	Precio Jornada
Jefe de Proyecto	50	400,00 €
Analista Funcional	43	344,00 €
Programador	25	200,00 €
Diseñador Gráfico	25	200,00 €

Total por Jornada
250,00 €

Figura Anexo C 4-2: Costes de personal

Actividad	Jornadas	Coste
Dirección de Proyecto		- €
Requerimientos de usuario	2	500,00 €
Analisis y Diseño de la solución	6	1.500,00 €
Desarrollo de Prototipo	3	750,00 €
Desarrollo APP	30	7.500,00 €
Desarrollo WEB	11	2.750,00 €
Servicios de Interconexión	7	1.750,00 €
Pruebas de integración, carga, configuración, etc	25	6.250,00 €
Despliegue en entornos	2	500,00 €
Documentación	8	2.000,00 €
TOTAL	94	23.500,00 €

Figura Anexo C 4-3: Presupuesto

Como observamos el tiempo estimado para el desarrollo completo del proyecto son 95 jornadas, comprendidas en días laborables, aproximadamente cinco meses y un coste de 23.500€