CICLO FORMATIVO DE GRADO SUPERIOR
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

PROYECTO FIN DE CICLO

SafeBar

Pablo Sánchez Cabrero, Alejandro García García,
Miguel Moneo Carmona

CURSO 2020-21

TÍTULO: SafeBar

AUTORES: PABLO SÁNCHEZ CABRERO

ALEJANDRO GARCÍA GARCÍA

MIGUEL MONEO CARMONA

TUTOR DEL PROYECTO: Ernesto Ramiro Córdoba

FECHA DE LECTURA: ... Junio de 2021

En Madrid

Ernesto Ramiro Córdoba
Tutor del PFC

RESUMEN

SafeBar es un proyecto que nace de la necesidad de controlar el aforo y la asistencia en los locales pertenecientes al sector de la restauración en tiempos del COVID-19. Debido a la evolución de la pandemia, y a la rápida escalada de contagios que la ha acompañado, se ha hecho patente la necesidad de realizar un control en este tipo de establecimientos con el fin de prevenir que el virus siga propagándose sin control.

La idea central de esta aplicación es simple, una plataforma móvil que permita a cada usuario reservar en el restaurante que elijan, en un fecha y hora de su elección (pudiendo elegir también el número de personas que asistirán a esa reserva). Por otro lado, el restaurante en cuestión podrá usar esta app para controlar su aforo y gestionar sus reservas. De esta manera, la aplicación únicamente dejara reservar al usuario si este no está excediendo el aforo del local para ese momento en concreto. Gracias a esto se solucionan dos problemas importantes. Primero, se ayuda en la prevención y control del virus COVID-19. Segundo, se ayuda al sector de la restauración proporcionándole un entorno seguro para sus clientes, permitiéndole de esta manera continuar con su actividad económica.

Este proyecto ha sido desarrollado para un sistema Android, con las ultimas funcionalidades y una conexión a base datos en tiempo real, lo que permitirá ofrecer tanto a usuarios como a restaurantes, un servicio de la mayor calidad. Se prevé que pueda seguir creciendo en el futuro y que pueda incorporar nuevas funcionalidades para continuar apoyando al sector de la restauración en estos tiempos de pandemia.

ABSTRACT

SafeBar is a project that started with the urgent need of controlling the capacity and the attendance in bars and restaurants due the crisis originated by the COVID-19 pandemic. The evolution of the disease and the rapid spread of infections have created an urgent necessity of control. With this checking, we hope it can help and stop the fast spread of this harmful disease.

The main goal of this app is quite simple, is a mobile platform which allows every user to book on the restaurant they want. The bookings are made by date and hour, it can also be included how many people make up the reservation. On the other hand, the establishment has the ability of managing and confirming their bookings. The app will not allow a user to make a reservation that exceeds the capacity established by the government and the establishment. Thanks to this, the two main problems will be solved. In the first place, it helps to prevent the spread of the virus. Secondly, the sector of the restoration will not suffer the financial consequences originated by the virus so much and they can offer a safe service to its clients.

This project has been made for an Android system, with the latest improvements on tech and with a strong connection in real time to a database. This will allow offering the users and the restaurants a great service. Our predictions say that the app will grow exponentially and will incorporate new functionalities to continue our bond with the restoration sector in these difficult and tragic times.

AGRADECIMIENTOS

Sería un poco egoísta y nada creíble decir que este trabajo ha sido de tres personas, porque detrás de este trabajo hay mucha gente detrás que ha aportado su granito de arena.

Hay que agradecer al centro, centro profesional europeo, el tener este grado superior de gran calidad educativa, buenas instalaciones y números recursos adaptados a las tecnologías más actuales del mercado. Al gozar de dichas herramientas se ha podido sacar el proyecto adelante.

Durante estos dos años ha habido un buen número de docentes que han hecho una labor encomiable en materia del uso de las nuevas herramientas solicitadas por el mercado, extensos conocimientos en todas las áreas relacionadas con la informática y una gran mentalidad para triunfar en el mundo del mañana como profesionales. El profesorado ha estado a la altura con sus conocimientos que han servido para desarrollar el proyecto.

Cabe mencionar el apoyo de la familia durante estos dos años en algunos casos financiando el curso y en otros dado un gran apoyo para conseguir sacar la titulación correspondiente. Debido a este pilar este trabajo al final del curso se ha afrontado con suficiente energía.

CONTENIDO

1. INTRODUCCIÓN

En este punto se introducirá al proyecto, SafeBar, explicando su propósito, motivación y antecedentes. Se pondrá en contexto de cómo surgió la idea, como se decidió ponerla en práctica y el por qué.

1.1. Objetivos

Los objetivos de esta aplicación, llamada SafeBar, son los siguientes enumerados:

- Ayudar al sector de la restauración en estos tiempos de pandemia.
- Ofrecer espacios seguros y limpios para evitar el contagio del COVID-19 a los clientes y a los empleados de los restaurantes.
- Registrar la información de los clientes y restaurantes para mejorar y agilizar la experiencia gastronómica.

Cuando se habla de ayudar a los restaurantes, se tiene en cuenta la difícil situación que atraviesan estos negocios por motivos sanitarios, por medidas impuestas por los distintos organismos gubernamentales y por la crisis económica surgida por la pandemia. La aplicación fomenta el desarrollo de la actividad de estos negocios en estos difíciles momentos.
Al hablar de los espacios seguros, se refiere a un control de aforo y horario para gestionar mejor el número de personas en el local y poder cumplir con las distintas normativas vigentes.
Si se registran los datos de los clientes aparte de una agilidad mayor esto proporciona que los clientes tengan más difícil la poco común pero lamentable práctica de irse sin pagar.

1.2. Motivación

Este trabajo ha sido elegido para intentar ayudar al sector de la restauración que tanto ha sido castigado en nuestro país, aunque siempre hay que respetar las medidas sanitarias propuestas por los médicos y científicos que hacen una gran labor por la salud pública. En este país dicho sector juega un papel fundamental en la vida cotidiana de los ciudadanos españoles. Debido a la pandemia surge la necesidad de este sector a adaptarse a los nuevos tiempos o morir lentamente. Debido a que también forman parte de la vida de los integrantes de este grupo hacen que sea un interés común intentar reflotar la hostelería.

Figura 1: Artículo

1.3. Antecedentes

La pandemia llegó a nuestro país a principios de 2020 y debido a la situación sanitaria el gobierno decidió cerrar nuestro país y que todos sus habitantes permanecieran en su domicilio hasta finales de mayo.

Tras el confinamiento el mundo había cambiado y tuvo que aprender a convivir con este virus. Para evitar los contagios se impusieron una serie de normativas a la hostelería que castigaron mucho a dicho sector.

Para convivir con el virus, pero evitar la quiebra de los negocios surge la necesidad de la hostelería de implementar cambios. Este trabajo puede ayudar un poco a los restaurantes y bares en esta nueva situación mundial. Se respeta el uso de mascarilla, la distancia y la desinfección, pero se implementa un sistema para controlar el aforo y el horario para no paralizar la restauración.

En otros sectores también han surgido aplicaciones para convivir con el virus e intentar frenar la crisis sanitaria y financiera.

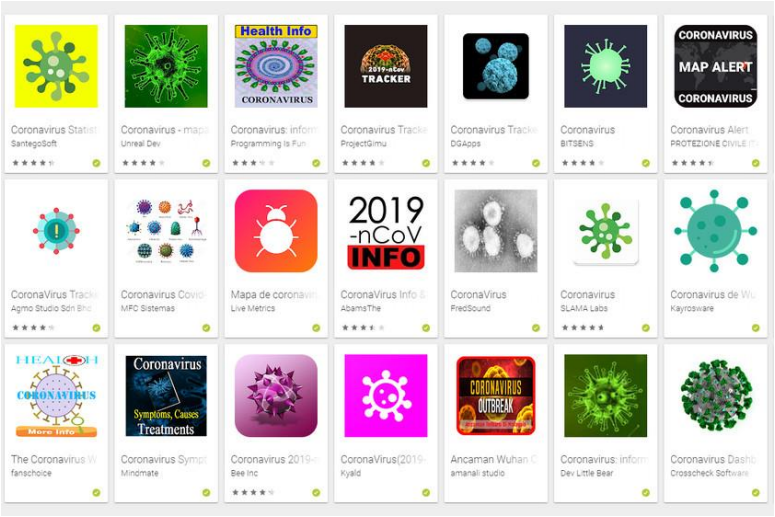

Figura 2: Aplicaciones en el Play Store

2. DESARROLLO DEL PROYECTO

En esta parte se pasará a comentar todo aquello relacionado con el desarrollo y la realización del trabajo. Se explicará el proceso, las herramientas y la metodología que se han usado, y se detallará el funcionamiento de la aplicación, así como sus posibilidades y aspectos a mejorar (futuras actualizaciones).

2.1. Herramientas tecnológicas

A continuación, vamos a exponer las principales tecnologías que se han utilizado en este proyecto. Desde lenguajes de programación, hasta programas de diseño. Se intentará exponer de una manera clara y sencilla, y resaltando los aspectos más importantes de cada uno y mencionando su contribución al proyecto.

- Java: la aplicación está programada en este lenguaje de programación. Un lenguaje compilado, orientado a objetos, escrito en C y que hoy en día sigue siendo de los más utilizados del mundo. Permite crear aplicaciones complejas, está totalmente integrado en los sistemas Android y ofrece muchísimas posibilidades tanto a nivel de desarrollo como de diseño.

Figura 4: Logotipo Java

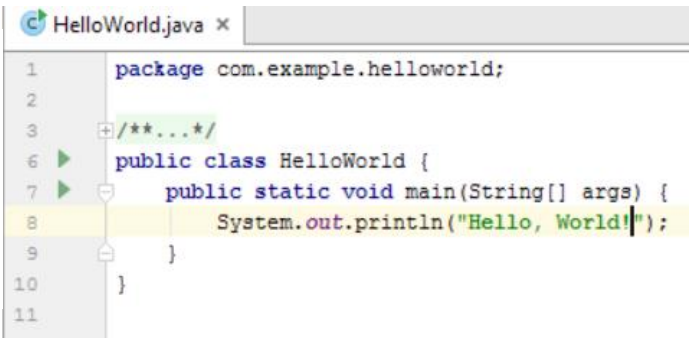

Figura 3: Código Java

- Android Studio: para construir esta aplicación se ha utilizado el programa conocido como Android Studio. Este entorno de desarrollo se utiliza en la construcción de aplicaciones multiplataforma dirigidas a dispositivos que utilicen Android como sistema operativo. Ha sido desarrollado por Google, es gratis para Windows, Mac y Linux, y tiene numerosas posibilidades de implementación, tanto a nivel gráfico y visual, como a nivel de desarrollo puro.

Figura 6: Logotipo Android Studio

Figura 5: Inicio Android Studio

Permite el uso de multitud lenguajes de programación tales como Java, C, C++, o JavaScript. La parte gráfica es llevada a cabo en XML y CSS entre otros. A su vez también permite el uso de frameworks específicos, como Ionic o React Native, que aúnan diversas tecnologías permitiendo a los desarrolladores construir aplicaciones "hibridas" para dispositivos que utilicen distintos sistemas operativos, tales como IOS.

Este IDE es considerado, junto con Xcode, como el estándar de la industria, en lo que se refiere al desarrollo de aplicaciones móviles.

Cuenta con el respaldo de Google, con una cantidad casi infinita de bibliotecas que nutren al programa de multitud de opciones a la hora de programar y desarrollar, con un sistema de base datos en la nube y en tiempo real conocido como Firebase que permite gestionar el almacenamiento de permisos, información, fotos, etc., y con un sistema completamente integrado para compartir tu trabajo con otros desarrolladores a través de la plataforma de repositorios remotos conocida como Github. A continuación, podemos ver un ejemplo de su interfaz:

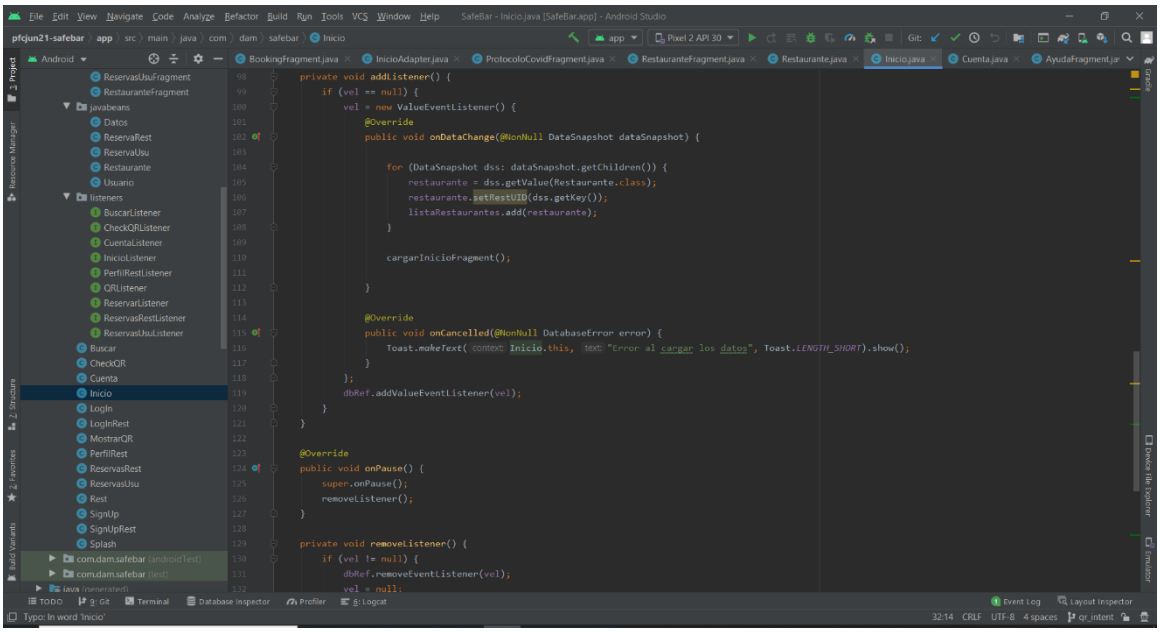

Figura 7: Interfaz Android Studio

Como podemos ver, a nivel visual es sencilla. Sin embargo, cuenta con numerosas opciones en cuanto a la configuración y las herramientas se refiere, dotando al programa de una capa de complejidad que por supuesto se ve recompensada en las posibilidades que ofrece para el desarrollo y el diseño para todo tipo de aplicaciones.

- Firebase: este sistema de gestión de datos en tiempo real desarrollado por Google está directamente implementado en Android Studio, ofreciendo todos sus servicios al desarrollador y permitiéndole realizar multitud de labores de una forma mucho más fácil e intuitiva de lo que se hacía anteriormente.

Figura 8: Logotipo Firebase

Es mucho más que una base de datos y es que entre otras cosas permite al desarrollador:

- Autentificación de usuarios y administradores: permite, de una manera sencilla, registrar y controlar los permisos y accesos a la base de datos en tiempo real.

- RealTime Database: base de datos en tiempo real que permite almacenar información mediante un formato XML, con una estructura de árbol. Su acceso y modificación en cualquier momento unidas a la versatilidad que ofrece a la hora d estructurar el modelo de datos la hacen una herramienta indispensable para cualquier aplicación móvil.

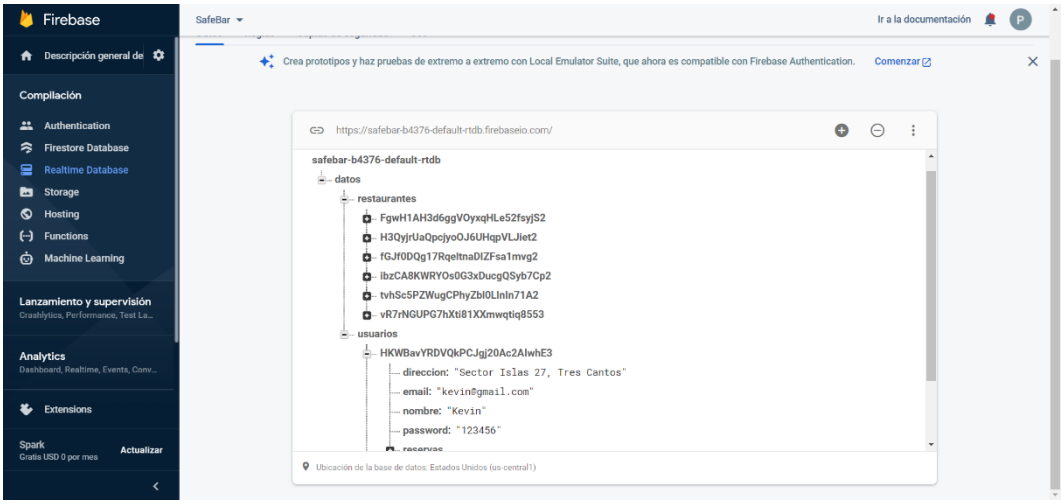

Figura 9: RealTime Database desde consola Firebase

- Storage: este servicio se utiliza para guardar fotos en la nube. Permite el registro, el acceso y el borrado en cualquier situación. Las fotos se pueden clasificar y agrupar en carpetas para después referenciarlas en la parte de RealTime Database y llamarlos desde la aplicación en cualquier momento.

- Github: es un controlador de versiones que permite a los desarrolladores compartir sus trabajos (códigos, diseños, etc.) con quienes ellos deseen. De esta manera permite conectar tu repositorio local con otro remoto en la nube y trabajar de forma simultánea con otros desarrolladores en un mismo proyecto mediante el sistema de ramas.

  Esto permite que el trabajo se divida entre los integrantes de un equipo, de manera que cada uno pueda realizar su parte del trabajo sin comprometer la del resto.

Figura 10: Logotipo GitHub

Gracias a sus diferentes funcionalidades, podemos descargar y subir progresos, también realizar el conocido "merge" que consiste en juntar diversas ramas en una, combinando el trabajo de varias personas y obteniendo como resultado final la "fusión" de ambos códigos.

Github estas completamente integrado en AndroidStudio, lo que ha facilitado su utilización en este proyecto, permitiendo a este equipo realizar un reparto de tareas, y una actualización constante del proyecto de una manera cómoda y sin riesgos.

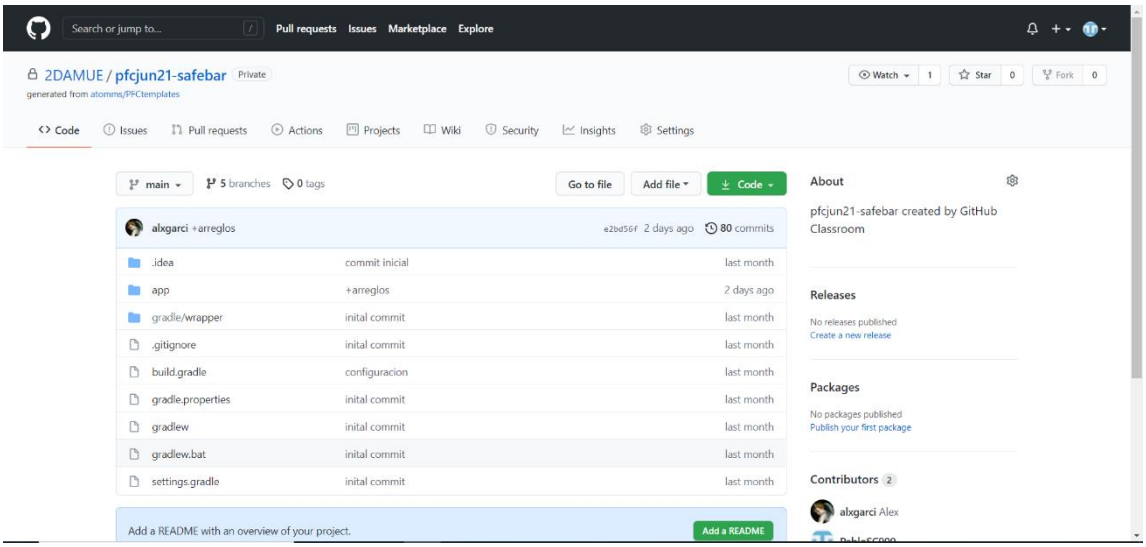

Figura 11: Interfaz de la herramienta GitHub

Por otro lado, la interfaz de la página es sencilla e intuitiva, permitiendo la gestión del espacio de trabajo y de las ramas de una manera fácil y rápida.

- Adobe XD: esta herramienta desarrollada por Adobe permite crear plasmar la idea de prototipo llevando la parte del diseño a un nivel más profesional. Mediante Adobe XD podemos crear una serie de mockups que nos darán un marco en el que trabajar luego, para que, a la hora de realizar el desarrollo a nivel de código, usaremos estos mockups para guiarnos y concentrarnos únicamente en la parte de programación.

Este software nos permitirá concretar nuestras ideas para realizar un boceto de la aplicación donde podremos empezar a plasmar aspectos fundamentales como la estructura, los colores, el número de ventanas o la navegabilidad. A su vez también nos permite compartir nuestro trabajo con otras personas, tanto en formato presentación, como permitiendo a estas la edición de este. Es, sin duda, el programa idóneo para realizar un primer prototipo de cualquier aplicación.

Figura 12: Logotipo Adobe XD

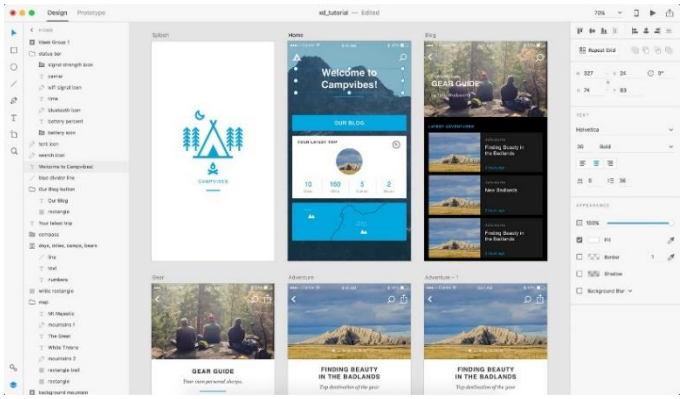

Figura 13: Interfaz Adobe XD

Con esto concluye el listado y exposición de las herramientas tecnológicas más importantes utilizadas en este proyecto. Por supuesto ha habido más que han sido de ayuda en determinados momentos y nos han ayudados con el progreso del proyecto, sin embargo, estas han sido sin duda las que mayor han tenido en el resultado final.

2.2. Planificación

El proyecto ha sido planificado considerando los factores más relevantes, distribuyendo las tareas de la forma más eficiente. Por supuesto, la comunicación entre los miembros del equipo ha sido constante, y para ello se ha hecho uso de numerosas herramientas que detallaremos a continuación.

En primer lugar, hablaremos de la planificación a nivel de fechas, y después del reparto de tareas. Por último, pasaremos a exponer las herramientas utilizadas en dicho proceso.

1. Planificación a nivel de tiempo: el proyecto ha contado con una duración de 9 semanas, por lo que ha sido fundamental la fijación de unos objetivos quincenales y el seguimiento de la evolución y del progreso de estos. Estos *sprints* de dos semanas se han dividido de la siguiente manera:

    - Semanas 1 y 2: elaboración de la idea, diseño del prototipo (colores, fuentes, ventanas), estructura básica de la aplicación y del modelo de datos. Configuración de la base de datos (permisos y conexiones). Ventanas de Splash, LogIn y SignUp.

    - Semanas 3 y 4: programación de la estructura de la aplicación (activitys, fragments, listeners… y comunicación entre ellos). Inclusión de primeras funcionalidades e implementación de la primera capa de formatos y estilos.

    - Semanas 5 y 6: se prueban las funcionalidades básicas con datos de Firebase. Inclusión de nuevas funcionalidades más avanzadas. Implementación segunda capa de estilos y formatos.

    - Semanas 7 y 8: Se prueban funcionalidades más avanzadas con datos de Firebase. Inclusión de ultimas funcionalidades más avanzadas y se testean con Firebase. Implementación tercera capa de estilos y formatos.

    - Semana 9: Arreglos finales a nivel de funcionalidad, del diseño, y de la base de datos. Realización de la memoria.

2. Distribución de tareas: el proyecto se ha dividido teniendo en cuenta las capacidades de cada uno y las áreas en las que podía aportar más. Aunque todos hemos acabado colaborando en prácticamente la totalidad del proyecto, sí que se han definido unos roles centrales para facilitar el

reparto de tareas. Dicho reparto ha sido llevado a cabo como se expone a continuación:

- Alejandro García García: encargado de la parte gráfica, conexión de componentes y la incorporación de elementos novedosos y dinámicos que dotaran a la aplicación de identidad propia.
- Pablo Sánchez Cabrero: encargado de la parte de Firebase (conexión y gestión de datos a través de la app), y de dotar de funcionalidad al código.
- Miguel Moneo Carmona: encargado de montar la estructura de ventanas y de dotar de funcionalidad al código.

Por último, se comentarán las herramientas que se han utilizado para realizar esta planificación y mantener la comunicación entre los miembros del equipo a lo largo de estas semanas:

- Herramientas para la planificación:
  - Hojas de cálculo de Google
  - Trello
  - Microsoft Excel

- Herramientas para la comunicación:
  - Slack
  - Skype
  - WhatsApp
  - Gmail

2.3. Descripción del trabajo realizado

En este apartado se expondrán los aspectos más importantes del proyecto, así como sus principales características y una visión global del trabajo realizado.

En primer lugar, hablaremos de la estructura básica de SafeBar. Para esto haremos referencia a como se ha empaquetado y distribuido el código, los estilos, y a como está estructurada la base de datos en Firebase.

Después comentaremos las principales funcionalidades de la aplicación y hablaremos de su formatos y estilo. Todo aquello que dota de identidad a SafeBar.

Por último, se explicarán los aspectos más importantes relacionados con el comportamiento del código y como la estructura creada maneja la información (proveída por el usuario) a lo largo de toda la experiencia de navegación.

Por tanto, empecemos con el primer punto: la estructura. La estructura de SafeBar podría dividirse en tres partes:

Por un lado, tendríamos la estructura del código Java, por otro lado, el sistema de almacenamiento de recurso gráficos, y por último la estructura del modelo de datos especificado en Firebase. El esquema general quedaría así:

- Estructura Java: compuesto por todas las clases Java e Interfaces.
- Recursos: comprenden todos los Iconos, Imágenes y layouts.
- Modelo de Datos (Firebase): cómo se organiza la información en la base de datos.

- Estructura Java: para esta aplicación se han creado un total de:
  - 14 Activitys.
  - 16 Fragments dinámicos.
  - 9 Interfaces (Listener).
  - 4 clases para representar el modelo de datos.
  - 5 clases destinadas a conexiones y adaptaciones de formatos (adapters y menús).

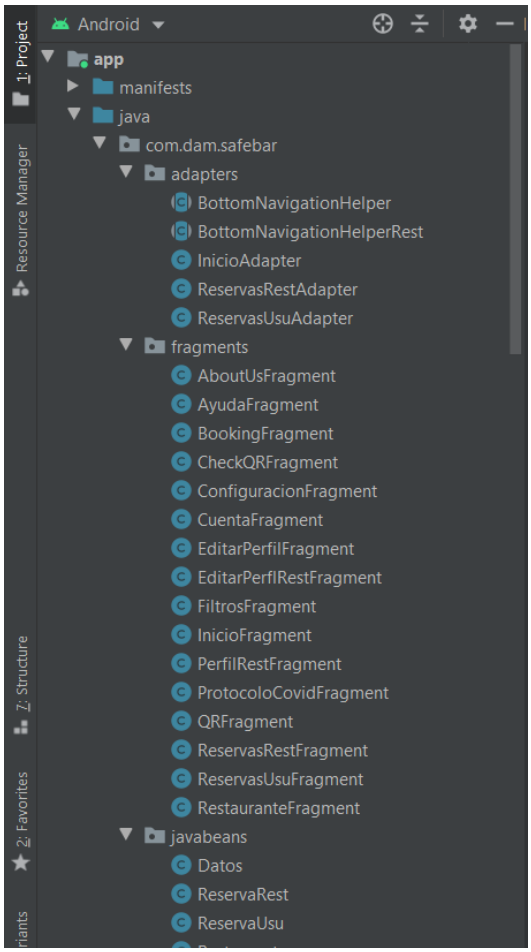

Figura 14: Estructura a nivel de proyecto en Android Studio

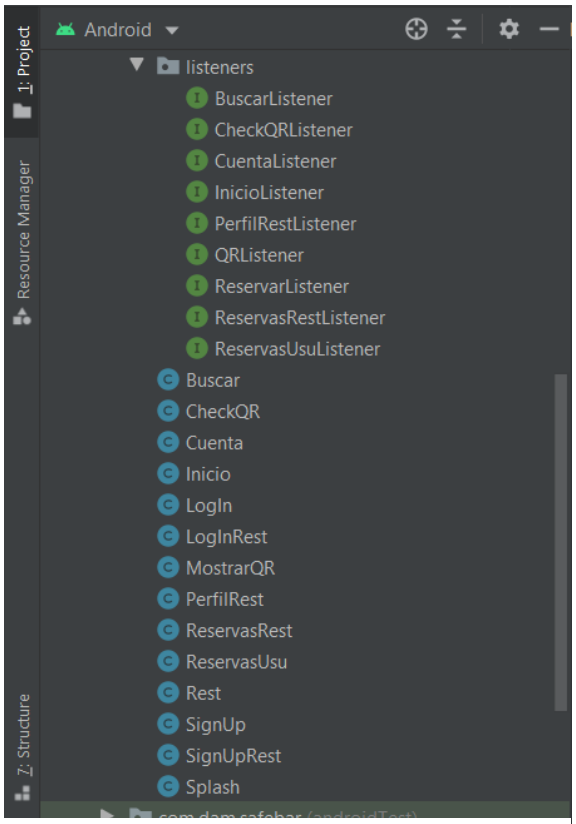

Figura 15: Estructura a nivel de proyecto en Android Studio 2

- Recursos: por otra parte, SafeBar cuenta con números recursos para dotar al código de un formato y estilo propios:
    - 36 Layouts.
    - 3 animaciones.
    - 4 menús.
    - 4 fuentes de letra.
    - Un archivo donde agrupa los diferentes estilos y colores utilizados a lo largo de toda la aplicación.
    - Un Drawable donde almacena todas las imágenes e iconos usados a lo largo del desarrollo de la app.

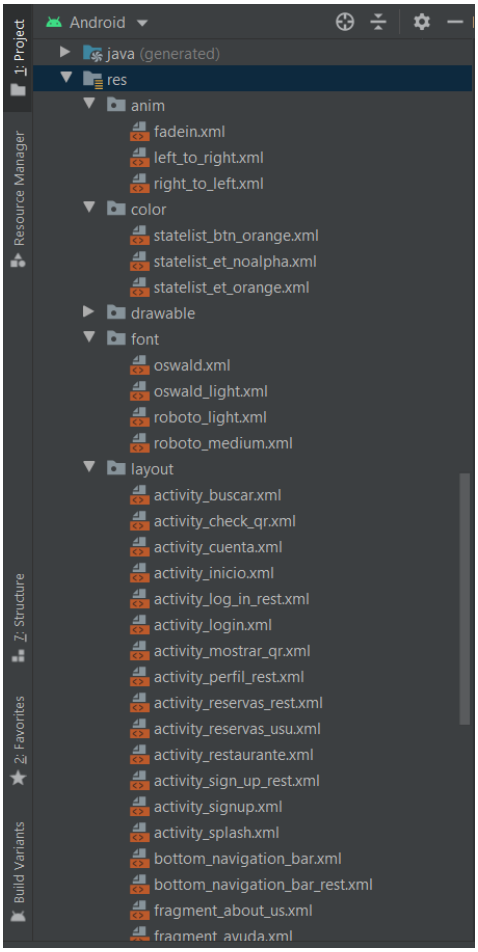

Figura 17: Estructura a nivel de proyecto en Android Studio, Recursos 1

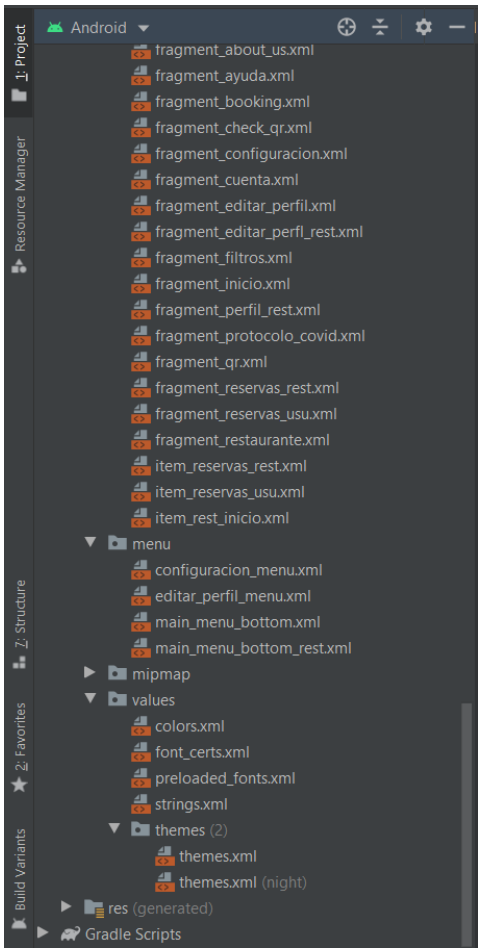

Figura 16: Estructura a nivel de proyecto en Android Studio, Recursos 2

- Modelo de datos (Firebase): El modelo datos está estructurado de la siguiente manera:
  - 2 entidades principales: Usuarios y Restaurantes. Estas contaran con una serie de atributos que representaran los principales datos de los usuarios de la aplicación, tanto desde la perspectiva de cliente (Usuario), como desde la perspectiva del local (Restaurante).

Figura 18: Entidad principal Usuario en Firebase

  - 2 tipos de reservas: ReservasUsu y ReservasRest. Agruparan los datos referentes a cada reserva. Ambas representan la misa reserva, pero una desde el punto de vista de Usuario y otra desde el de Restaurante.

Figura 19: Restaurante, atributos y reservas en Firebase

ReservasUsu hará referencia a las reservas de la parte de Usuarios y ReservasRest hará referencia a las reservas de la parte de Restaurante. Cada una de las dos tendrá ligeras diferencias en cuanto a los datos que encapsula, aunque ambas tendrán el mismo código en común (generado en el momento de creación de la reserva).

- Estructura en forma de árbol de Firebase: los datos estarán divididos en dos tablas principales "usuarios" y "restaurantes". Cada tabla contendrá los distintos usuarios y restaurantes clasificados por un código único generado durante su registro.

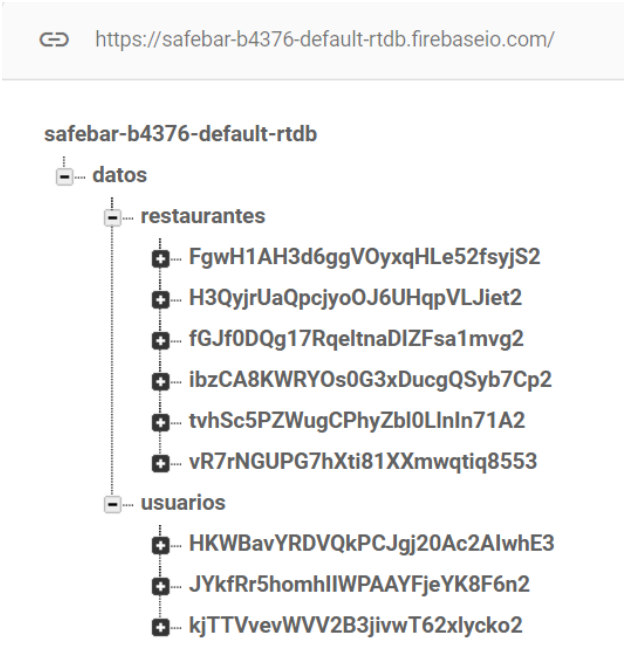

Figura 20: Entidades Restaurantes y Usuarios con sus códigos únicos generados en Firebase

Dentro de cada elemento podremos encontrar sus datos principales tales como nombre, email, dirección, etc. Además, cada uno tendrá su propio elemento de "reservas" donde se encontrarán almacenadas los datos de cada reserva que tengan (tanto en restaurantes como en usuarios). La ruta de reservas de usuarios presenta alguna diferencia respecto a la de restaurantes, aunque es bastante similar.

Una vez definida la estructura básica pasemos ahora a hablar de las principales funcionalidades de la aplicación, así como de formato y estilo.

- Principales funcionalidades: esta aplicación cuenta con una serie de funcionalidades a menudo relacionadas entre sí y que, en conjunto, definen la identidad de SafeBar.

  En primer lugar, cuenta con la posibilidad de iniciar sesión o de registrarse, a través de las ventanas de LogIn y SignUp (a las que se accede de forma automática después de que aparezca el Splash).

Figura 23: Splash de SafeBar

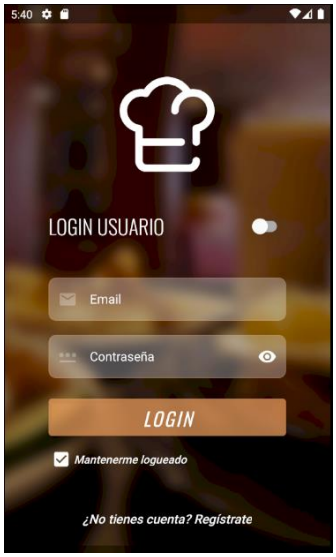

Figura 22: Inicio de sesión para usuarios de Safebar

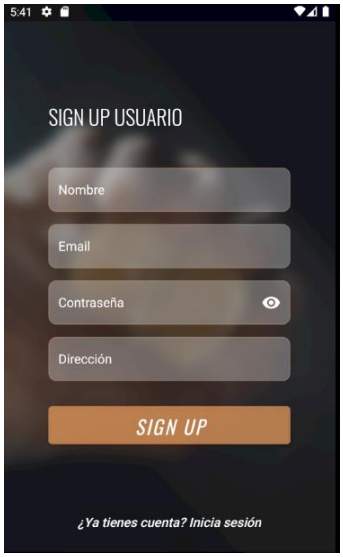

Figura 21: Registro para usuarios de SafeBar

Por su puesto estas dos opciones estarán disponibles tanto para un Usuario, como para un Restaurante. Para cambiar entre uno y otro se hará uso del switch situado en la parte superior de la ventana de LogIn.

Una vez que iniciemos sesión, o que nos registremos, accederemos a la parte principal de la aplicación. En esta habría dos perspectivas dependiendo si hemos accedido como un Usuario o como un Restaurante. Empecemos por la parte de Usuario.

En esta parte encontraremos que la aplicación está dividida en 4 secciones principales: Inicio, Buscar, Reservas y Cuenta. Podremos acceder a cada una de ella con un BottomBar que nos muestra cuatro iconos que representan cada una de estas secciones.

A parte de estas cuatro mencionadas anteriormente podemos considerar que hay otra sección más, a la que solo se puede acceder desde la sección de Inicio o de Buscar. Esta sección sería la de Restaurante, que permite entre otras cosas, que el usuario realice la reserva.

Pasemos a hablar ahora de cada una de estos apartados de forma más concreta:

- Inicio: esta ventana tiene la función de cargar los restaurantes registrados y almacenados en Firebase. Mostrará un Recycler View con una lista de restaurantes, mientras expone de manera visual sus principales características (foto, nombre, dirección, aforo, precio medio, calificación).

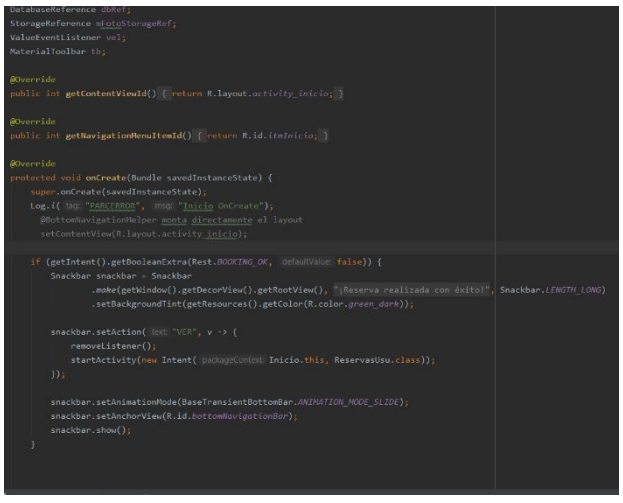

Figura 24: Código de OnCreate inicio con el Snackbar de 'Reserva Realizada'
(ANEXO A.2)

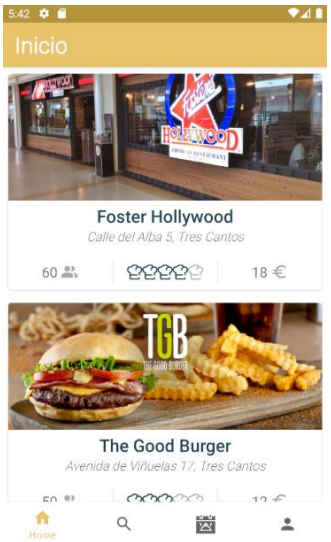

Figura 25: Inicio de SafeBar

Al pulsar en uno de los elementos accederemos a la sección de Restaurante. En dicha sección cargaremos los datos (directamente de la base de datos) del restaurante que hemos pulsado en la sección de Inicio.

- Buscar: esta sección cuenta con un Edit Text y un botón para buscar restaurantes mediante un filtrado de nombre. Por poner un ejemplo, si yo introduzco "ti", mostrará una lista (con el mismo formato que Inicio) de todos los restaurantes guardados en base de dato que empiecen por "ti" (ignorando diferencias entre mayúsculas y minúsculas).

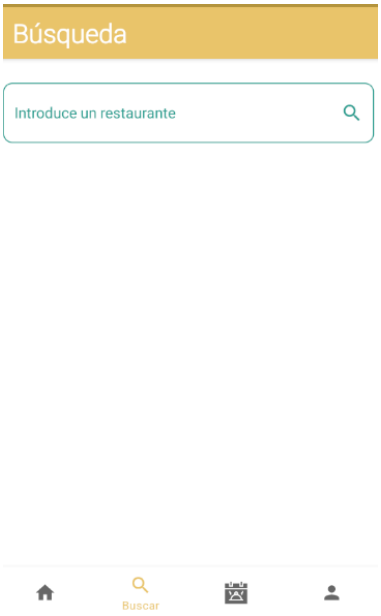

Figura 27: Búsqueda de SafeBar

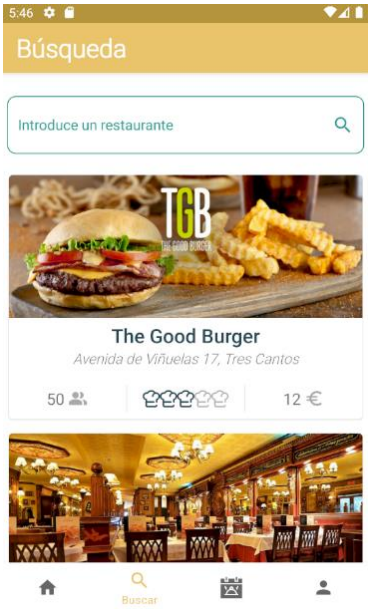

Figura 26: Búsqueda de SafeBar 2

De nuevo, al pulsar en un elemento de la lista, se accederá a la sección de Restaurante, donde se cargarán los datos desde Firebase, del elemento seleccionado.

- Restaurante: esta sección mostrar los datos de un restaurante en específico seleccionado en las secciones de Inicio o Buscar. Únicamente se puede acceder desde ellas.

Figura 29: Vista del restaurante antes de reservar en SafeBar

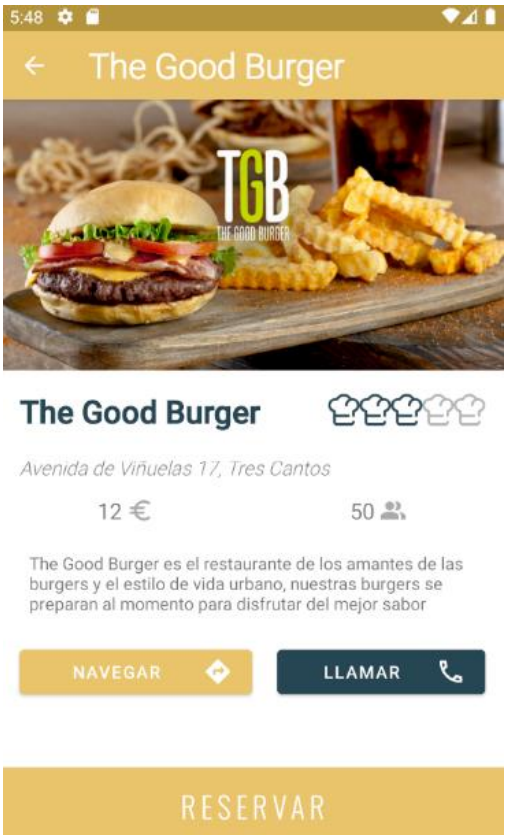

Figura 28: Vista de restaurante antes de reservar en SafeBar 2

A parte de mostrar los datos del restaurante de una forma clara y visual, nos permite tres funcionalidades mediante tres botones: navegar, llamar y reservar. El botón de "navegar" nos permite consultar de forma automática la localización del restaurante en Google Maps. El botón de "llamar" nos permite llamar al restaurante en cuestión.

El botón de “reservar”, por último, nos permite acceder a la siguiente ventana:

Figura 30: Código para la carga de la reserva ***(ANEXO A.3)***

Figura 32: Vista de Reserva en SafeBar

Figura 31: Código para la carga de la reserva 2 ***(ANEXO A.3)***

En esta ventana seleccionaremos una fecha y una hora (mediante un Date picker y un Time picker), y un número de personas. Después, al pulsar el botón de “reservar”, la aplicación comprobará si en la base de datos existen reservas para ese restaurante, en ese día y en esa hora que, sumando su número de personas con el número de personas de tu reserva puedan exceder el aforo del mismo.

En caso de que dicho aforo se exceda, el sistema te lo notificará y te impedirá realizar la reserva con esos parámetros. En caso de que el aforo no exceda la reserva se realizara generando un código único y guardándola en la base de datos con una ruta específica, según los parámetros que hayamos introducido, tanto en la parte del

usuario que la ha hecho, como en la parte del restaurante donde se ha hecho.

- Reservas: en esta sección se mostrarán todas las reservas que el usuario tiene, ordenadas por fecha. A parte de mostrar los datos principales de cada reserva, permitirá al usuario acceder a cada una para mostrar su código QR (código generado a partir del código único generado durante la reserva).

Figura 34: Visualización de reservas en SafeBar

Figura 33: Visualización de QR de Reserva en SafeBar

Este código QR podrá ser escaneado desde la parte de Restaurante, por el restaurante en cuestión, para así validar la reserva.

- Cuenta: esta sección se dividirá en cuatro subsecciones. Estas serán Configuración, Ayuda, About US y Normativa COVID.

Figura 36: Vista de Cuenta en SafeBar

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_cuenta, container, attachToRoot: false);
    Button btnConfiguracion = view.findViewById(R.id.btnCuentaConfiguracion);
    Button btnAboutUs = view.findViewById(R.id.btnCuentaAboutUs);
    Button btnAyuda = view.findViewById(R.id.btnCuentaAyuda);
    Button btnProtocoloCovid = view.findViewById(R.id.btnCuentaProtocoloCovid);
    tvEmail = view.findViewById(R.id.tvCuentaEmail);

    dbRef = FirebaseDatabase.getInstance().getReference( path: "datos/usuarios");
    fba = FirebaseAuth.getInstance();
    user = fba.getCurrentUser();

    addListener();

    btnConfiguracion.setOnClickListener(v -> listener.abrirConfiguracion());

    btnAboutUs.setOnClickListener(v -> listener.abrirAboutUs());

    btnAyuda.setOnClickListener(v -> listener.abrirAyuda());

    btnProtocoloCovid.setOnClickListener(v -> listener.abrirProtocoloCovid());

    return view;
}
```

Figura 35: Código de botones de Cuenta con sus Listener ***(ANEXO A.1)***

La más relevante será la de Configuración, donde el usuario podrá ver y editar su perfil, así como cerrar sesión.

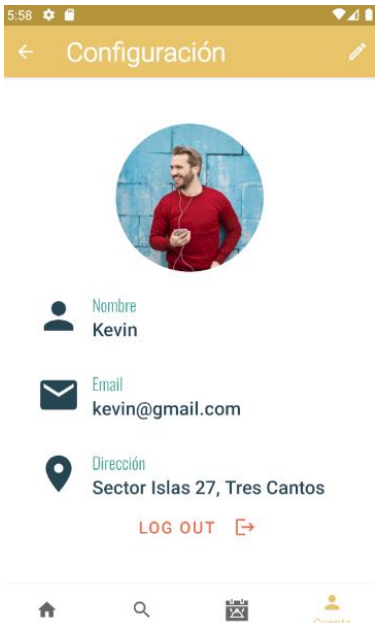

Figura 38: Vista de Configuración en SafeBar

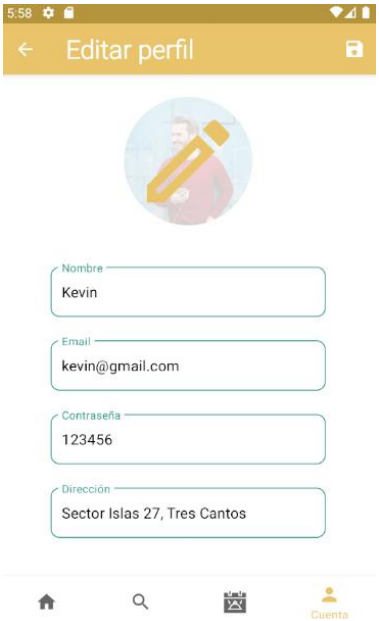

Figura 37: Vista de Editar Perfil en SafeBar

About Us y Normativa COVID proveerán información importante tanto de los desarrolladores como del COVID-19.

Figura 40: Vista de About Us en SafeBar

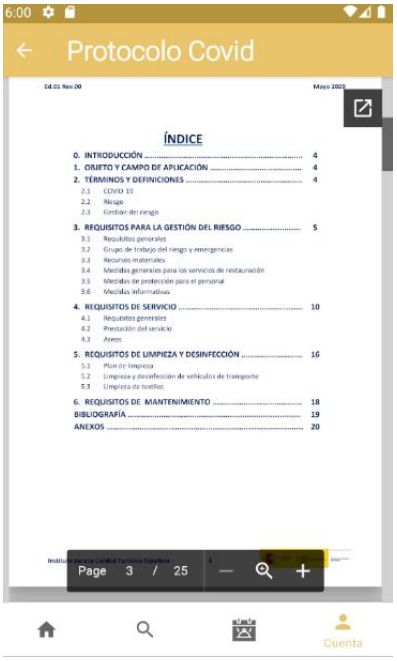

Figura 39: Vista de Protocolo Covid en SafeBar

Ayuda permitirá al usuario contactar con SafeBar a través de llamada telefónica o de correo electrónico.

Figura 41: Vista de Ayuda en SafeBar

Hablemos ahora de la parte de Restaurante. Esta parte es más sencilla, pero absolutamente necesaria para funcionamiento de SafeBar. Está compuesta por dos secciones: Reservas y Perfil. Hablemos de ellas:

- Reservas: esta parte es bastante similar a su homóloga de la parte de Usuario. Básicamente muestra los datos (aunque estos cambian un poco) de las reservas que tiene cada restaurante.

Figura 43: Vista de Reservas en la parte del restaurante en SafeBar

Figura 42: Vista de Verificar QR en la parte del restaurante en SafeBar

La diferencia es que al pulsar en una reserva no se abrirá ningún QR. Esta vez lo que se abrirá será una nueva ventana donde podremos escanear un QR a través de la cámara del teléfono, y validar si el código escaneado coincide con el código de la reserva que hemos seleccionado anteriormente.

Figura 45: Código de la Reserva en Restaurante ***(ANEXO A.4)***

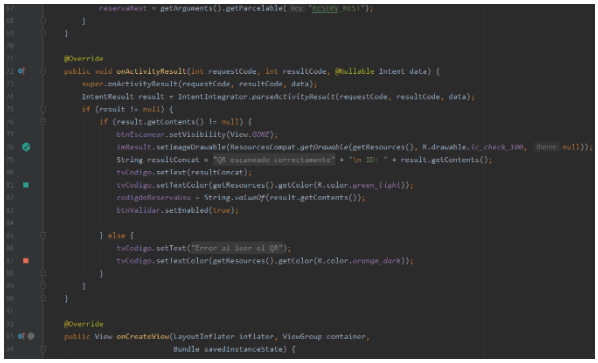

Figura 44: Código para abrir la interfaz de escaneo del QR ***(ANEXO A.5)***

Si no coincide, la aplicación nos lo dirá, pero en caso de que sí que coincida, la aplicación nos permitirá "validar" la reserva. Esta validación notificara al restaurante que toda esta correcto y borrara la reserva en cuestión, tanto para el usuario, como para el restaurante.

- Perfil: aquí el restaurante en cuestión podrá ver sus datos de perfil y modificarlos, así como cerrar sesión.

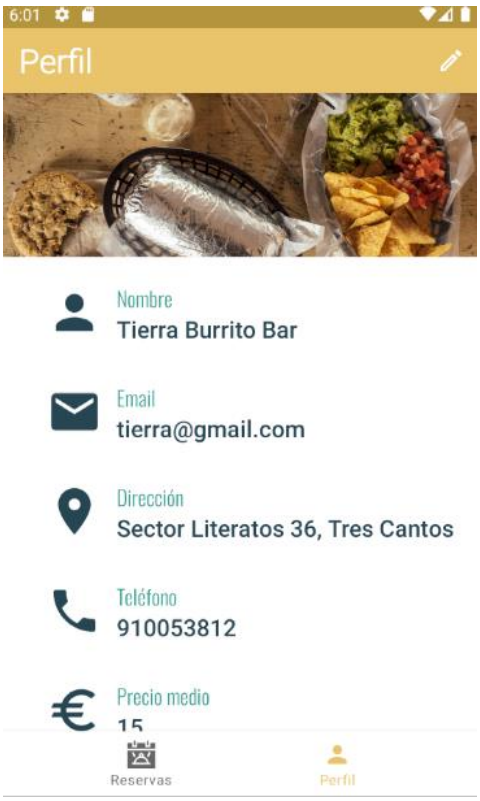

Figura 47: Vista de Perfil del restaurante en SafeBar

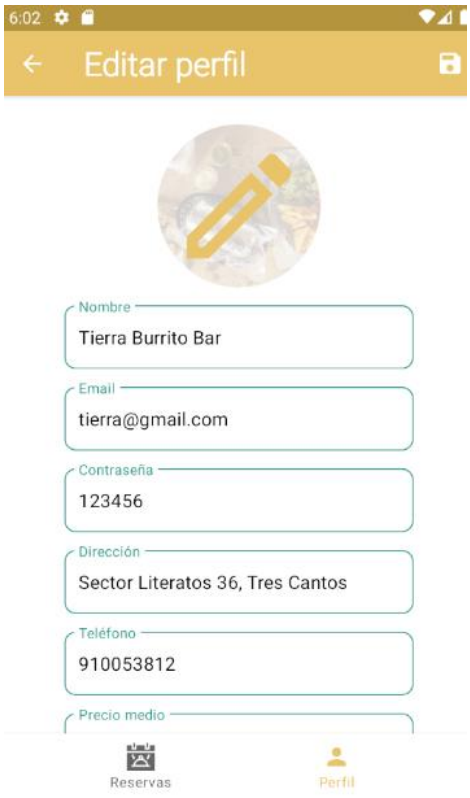

Figura 46: Vista de Editar Perfil del restaurante en SafeBar

Para finalizar la exposición acerca de las funcionalidades de la aplicación, añadir que durante toda la navegación tendremos la opción de volver atrás pulsando la flecha situada en el App Bar (en la parte superior izquierda de la ventana), así como de permanecer logeado hasta el momento en que cerremos sesión. (ANEXO)

Hablemos ahora acerca de los formatos y estilos implementados en SafeBar. Todos los elementos incorporados en la aplicación están pensados para dotarla de identidad y personalidad. Se ha querido ofrecer un estilo dinámico y que concuerde con la temática de la aplicación.

La disposición de los botones, la navegabilidad y la organización de las ventanas ha sido pensada para facilitar al usuario una experiencia de uso fácil y satisfactoria.
Analicemos pues los aspectos más importantes del estilo tan personal que posee SafeBar:

- Recursos multimedia: como hemos visto anteriormente, en el LogIn y SignUp se han implementado una serie "gifs" de fondo que cambian en función de en qué parte estemos (Usuario o Restaurante), y que dotan a estas ventanas de un mayor atractivo.

- Logo: como podemos observar el logo de SafeBar es un Gorro de chef. De esta manera hacemos referencia directa a la identidad y la temática de nuestra aplicación. Añadir también que este logo es usado a lo largo de toda la aplicación, desde el Splash, hasta las ventanas de Inicio y Restaurante (donde se utiliza para indicar la calificación que tiene el restaurante en cuestión).

Figura 49: Logotipo de SafeBar

Figura 48: RatingBar con el logotipo de SafeBar

- Colores: el color predominante a lo largo de toda la aplicación es el naranja, dividiéndose en light y dark, para ofrecer una mayor versatilidad. El color secundario sería el verde que de nuevo se divide en más tonalidades para intentar a acompañar y complementar al naranja, que en esta ocasión no tiene un sentimiento tan eléctrico, y que opta por una versión más apagada o incluso pastel.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="green_dark">#264653</color>
    <color name="green_light">#2A9D8F</color>
    <color name="yellow">#E9C46A</color>
    <color name="yellow_dark">#b4943c</color>
    <color name="orange_dark">#E76F51</color>
    <color name="orange_light">#F4A261</color>
    <color name="white_light_alpha">#CCFFFFFF</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="white_alpha">#4DFFFFFF</color>

</resources>
```

Figura 50: Colores usados
(ANEXO B.1, B.2)

- Fuentes: las fuentes utilizas son Roboto y Oswald. Ambas ofrecen más de una versión, ayudando a la aplicación a pulir un estilo sencillo y moderno, que huye de los recargos excesivos.

- Iconos: los iconos usados a lo largo de todo el proyecto han seguido una misma línea, basada en la sencillez y la sobriedad. Permitiendo al usuario saber en todo momento su situación dentro de la navegación, así como dejando claras las funciones de cada elemento.

- Estilos y menús: gracias a la organización y disposición de los elementos y los componentes a lo largo de toda la secuencia de ventanas, se ha conseguido crear una experiencia de usuario sencilla y satisfactoria.

Figura 51: Barra de Navegación de SafeBar
(ANEXO B.3)

Para acabar esta exposición acerca del estilo, añadir que se ha intentado en todo momento dotar a SafeBar de una identidad propia, intentando usar todas las herramientas disponibles para que la aplicación resulte visualmente sencilla y agradable para el usuario.

Por último, se hablará a del sistema de comunicación que se ha desarrollado para realizar una comunicación eficiente entre las ventanas, al mismo tiempo que se gestiona la información obtenida desde Firebase.

La aplicación se ha diseñado siguiendo una estructura en la que cada sección principal se corresponde con un Activity y en la que se cargaran de 1 a 3 Fragments (dependiendo de la sección).

La comunicación entre fragments se realizarán mediante una serie de Listener, que permitirán la navegabilidad entre secciones, así como facilitarán los datos necesarios a la siguiente sección para que esta puedo extraer la información referente a esa ventana desde Firebase.

En definitiva, la aplicación goza de una arquitectura solvente y eficiente que la permite llevar a cabo todas sus interesantes funcionalidades, así como de un estilo y formatos atractivos, que la dota de personalidad e identidad propias.

2.4. Resultados y validación

Considerando el objetivo inicial de este proyecto, creemos que el resultado final cumple, sin lugar a dudas con los requerimientos que se acordaron en un principio. La aplicación funciona correctamente, es eficiente y no tiene bugs importantes. Obviamente hay elementos que, con más tiempo, se podrían haber mejorado (tiempos de carga, animaciones), e incluso se podrían haber implementado una serie de nuevas funcionalidades.

No obstante, creemos teniendo en cuenta el tiempo dado y los objetivos marcados al inicio, se ha realizado un gran trabajo, con un resultado correcto y funcional, en el que se ofrece un servicio concreto que se solventa de manera lógica y sin errores.

El resultado del proyecto se ha validado en numerosas simulaciones, y el equipo está convencido de que es sólido y está lo suficiente testado para decir que un éxito.

Por supuesto hay un gran margen de mejora y funcionalidades que nos hubiera gustado pulir, sin embargo, creemos que el resultado final es tanto útil como visual y que soluciona un problema real en los tiempos que corremos actualmente.

3. CONCLUSIONES

Desde comienzo del proyecto se ha realizado un trabajo de una duración total de 9 semanas. En ese tiempo se ha elaborado la idea, diseñado el prototipo, desarrollado la aplicación, testeado los resultados y elaborado la documentación.

El trabajo resultante es una aplicación totalmente funcional que, si bien tiene algunos aspectos por pulir (como algunas animaciones, o los tiempos de carga), consideramos que, no solo se ajusta a los objetivos iniciales, si no que en muchos aspectos sobrepasa la valoración inicial que se hizo sobre este proyecto a principios del mes Abril.

Se ha desarrollado una aplicación que soluciona un problema real de una manera solvente, con una interfaz atractiva y mucho potencial a futuro.
SafeBar permite a un usuario reservar en sus restaurantes favoritos de una manera simple, y a estos restaurantes les permite gestionar sus numerosas reservas. Pero lo más importante es que esta aplicación ejerce como un control de seguridad para asegurarnos de que para la fecha y hora de mi reserva, no se va a exceder el aforo en ese local. SafeBar se asegura de crear un entorno seguro para los clientes y para el propio restaurante.

Por supuesto se han hecho múltiples ajustes desde la idea inicial de este proyecto, como por ejemplo la validación mediante escaneo de código QR. Esta funcionalidad ha pasado de ser una posibilidad en nuestra hoja de ruta, a ser una de los elementos más importantes y reseñables de nuestra aplicación. Este sistema, junto con el control de aforo, son las dos funcionalidades más representativas de Firebase, y dotan a este proyecto de una diferencia cualitativa tanto frente a la idea inicial del mismo, como frente a otras aplicaciones similares que actualmente ya están en el mercado.

A lo largo de estos dos meses, nos hemos encontrado con números problemas, la mayoría de ellos relacionados con las partes que realizaban interacción con Firebase, siendo esta la parte angular de nuestras preocupaciones.

También ha habido otras causas de retraso, como pueden ser los elementos más novedosos y complejos introducidos: código QR y componentes gráficos avanzados. No obstante, consideramos que han podido resolverse de la manera adecuada y que nos han permitido llevar el trabajo a un nivel más avanzado y con el que, finalmente, nos sentimos satisfechos.
Por supuesto, el tiempo ha sido un factor muy importante. Debido al corto calendario nos hemos visto obligados a "recortar" en algunos campos. Algunas funcionalidades finalmente no han podido estar y otras no son todo lo eficientes que nos gustaría. Sin embargo, el resulta final es sólido y no cuenta con errores graves ni bugs importantes.

En definitiva, consideramos que el trabajo realizado en este proyecto cumple (e incluso supera) las expectativas y la idea inicial que se planteó en el mes de abril. SafeBar es un aplicación útil, funcional, atractiva y testada. Soluciona una necesidad real de los tiempos actuales, y lo más importante, gracias a la estructura y el enfoque dado por este equipo creemos que puede tener mucho potencial de crecimiento, ya que cuenta con una idea clara y bien definida sobre la que se podrán incorporar nuevas mejoras, únicamente limitadas por el tiempo y la imaginación.

3.1. Innovación

Si algo hace SafeBar es innovar. Sin duda alguna el objetivo principal de esta aplicación es la de modernizar la relación entre los clientes y el sector de la restauración, incorporando las nuevas tecnologías y utilizándolas para estandarizar un servicio que, en plena pandemia, se antoja urgente, pero que, al mismo tiempo, también representa el futuro de este sector, facilitando la gestión a ambas partes y solucionando problemas tanto sanitarios como económicos.

Por último, hay que considerar también que, si bien existen otras aplicaciones que incorporan funcionalidades relacionadas con la reserva de restaurantes, distan mucho del servicio ofrecido por SafeBar. La mayor parte de las otras aplicaciones relacionadas con este sector ofrecen algunas funcionalidades importantes (aunque la mayoría de ellas tiene que ver con descuentos), pero ninguna aúna los dos aspectos más importantes con los que cuenta SafeBar: control sanitario (aforo) y gestión de reservas.

3.2. Trabajo futuro

En lo que se refiere al futuro de este proyecto, solamente añadir que SafeBar cuenta con un gran potencial de crecimiento y mucho margen de mejora. Aún faltan muchas funcionalidades por implementar y numerosas posibilidades por explorar.

Creemos firmemente que este proyecto tiene un camino más allá del puramente académico, y que en el futuro podría seguir desarrollándose para convertirse en una herramienta básica para multitud de usuarios, y un referente en este sector. Estamos seguro de que, con tiempo y dedicación, SafeBar podría desarrollarse en numerosos ámbitos llegando a abarcar, incluso, un amplio abanico de sectores.

4. BIBLIOGRAFÍA Y WEBGRAFÍA

Las páginas o proyectos que se han utilizado a modo de referencia para ayudarnos en la realización y optimización de tanto código como interacción entre componentes en SafeBar han sido las siguientes:

- Bumptech. (2021) github.com. Fecha de consulta: Abril 2021 *github.com/bumptech/glide*. Como referencia en la carga de imágenes desde la base de datos y ayuda con el caché de almacenamiento de estas dentro de la aplicación.
- Wasabeef. (2021) github.com. Fecha de consulta: Mayo 2021 *github.com/wasabeef/glide-transformations*. Como referencia en la modificación de imágenes y su optimización sin necesidad de almacenarlas.
- JourneyApps & ZXing. (2021) github.com. Fecha de consulta: Mayo 2021 *github.com/journeyapps/zxing-android-embedded*. A modo de ayuda para implementar la correcta lectura de Códigos QR y su generación.
- Wdullaer. (2020) github.com. Fecha de consulta: Mayo 2021 *github.com/wdullaer/MaterialDateTimePicker*. Implementada en la aplicación como librería con más funcionalidad que el TimePicker original de Google.
- GOOGLE. (2021) firebase.google.com. Fecha de consulta: Abril – Mayo 2021 *firebase.google.com/docs/guides*. Como referencia a la hora de implementar Firebase en la aplicación.
- GOOGLE MATERIAL. (2021) material.io. Fecha de consulta: Abril – Mayo 2021 *material.io/components*. De los recursos más usados durante el proyecto para poder seguir los estándares de Google a la hora de implementar la interfaz.

- MEDIUM. (2021) medium.com. Fecha de consulta: Abril – Mayo 2021 *kennay-kermani.medium.com/android-material-date-picker-and-material-time-picker-in-practice-11c01582e52e*. Usado como referencia a la hora de resolver dudas y problemas que se iban presentando.
- DbDev. (2012) stackoverflow.com. Fecha de Consulta: Mayo 2021 *stackoverflow.com/questions/9370293/add-a-remember-me-checkbox*. En general se trata de un foro de desarrollo que nos ha ayudado bastante, al igual que Medium, a la hora de resolver dudas o buscar información para implementar distintos componentes, como el 'Remember Me' en este caso.
- Maxime Jallu. (2017) stackoverflow.com. Fecha de Consulta: Abril 2021 *stackoverflow.com/questions/5448653/how-to-implement-onbackpressed-in-fragments*. Ayuda para la correcta implementación de la 'Flecha' para volver atrás.
- Nhp. (2017) stackoverflow.com. Fecha de Consulta: Mayo 2021 *stackoverflow.com/questions/45334369/how-to-open-and-directing-google-maps-using-intent*. Método para poder abrir Google Maps en base a una dirección, la dirección del restaurante.
- GOOGLE Developers. (2021) developer.android.com. Fecha de Consulta: Abril – Mayo 2021. *developer.android.com/guide*. Comprende una serie de guías y explicaciones del propio Google sobre métodos de Android y su correcta y óptima implementación. Se han usado como documentación a la hora de implementar los métodos de una forma eficiente.

5. ANEXOS

A. Código Java

1. *onCreateView* en línea 63 de *CuentaFragment.java*.

```
@Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_cuenta, container, false);
        Button btnConfiguracion = view.findViewById(R.id.btnCuentaConfiguracion);
        Button btnAboutUs = view.findViewById(R.id.btnCuentaAboutUs);
        Button btnAyuda = view.findViewById(R.id.btnCuentaAyuda);
        Button btnProtocoloCovid = view.findViewById(R.id.btnCuentaProtocoloCovid);
        tvEmail = view.findViewById(R.id.tvCuentaEmail);

        dbRef = FirebaseDatabase.getInstance().getReference("datos/usuarios");
        fba = FirebaseAuth.getInstance();
        user = fba.getCurrentUser();

        addListener();

        btnConfiguracion.setOnClickListener(v -> listener.abrirConfiguracion());

        btnAboutUs.setOnClickListener(v -> listener.abrirAboutUs());

        btnAyuda.setOnClickListener(v -> listener.abrirAyuda());

        btnProtocoloCovid.setOnClickListener(v -> listener.abrirProtocoloCovid());

        return view;

    }
```

2. *onCreate* y métodos del *BottomNavigationBar* en línea 42 de *Inicio.java*.

```
@Override
    public int getContentViewId() {
        return R.layout.activity_inicio;
    }

    @Override
    public int getNavigationMenuItemId() {
        return R.id.itmInicio;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.i("PARCERROR", "Inicio OnCreate");
//        @BottomNavigationHelper monta directamente el layout
//        setContentView(R.layout.activity_inicio);

        if (getIntent().getBooleanExtra(Rest.BOOKING_OK, false)) {
            Snackbar snackbar = Snackbar
                    .make(getWindow().getDecorView().getRootView(),
R.string.reserva_realizada, Snackbar.LENGTH_LONG)
                    .setBackgroundTint(getResources().getColor(R.color.green_dark));

            snackbar.setAction("VER", v -> {
                removeListener();
                startActivity(new Intent(Inicio.this, ReservasUsu.class));
            });

            snackbar.setAnimationMode(BaseTransientBottomBar.ANIMATION_MODE_SLIDE);
            snackbar.setAnchorView(R.id.bottomNavigationBar);
            snackbar.show();
        }
```

3. *addListener1* y *addListener2* en línea 272 de *BookingFragment.java*

```
 private void addListener1() {
        removeListener();
        if (vel == null) {
            vel = new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                    contAforo = 0;

                    for (DataSnapshot dss : dataSnapshot.getChildren()) {
                        reservaRestCheck = dss.getValue(ReservaRest.class);
                        contAforo = contAforo + reservaRestCheck.getNumPersonas();
                    }

                    contAforo = contAforo + numPersonas;

                    addListener2();

                }

                @Override
                public void onCancelled(@NonNull DatabaseError error) {
                    Snackbar snackbar = Snackbar

.make(getActivity().getWindow().getDecorView().getRootView(),
R.string.error_carga_datos, Snackbar.LENGTH_LONG)

.setBackgroundTint(getResources().getColor(R.color.orange_dark));

snackbar.setAnimationMode(BaseTransientBottomBar.ANIMATION_MODE_SLIDE);
                    snackbar.setAnchorView(R.id.btnBookingFragReservar);
                    snackbar.show();
                }
            };

dbRef.child("restaurantes").child(restUID).child("reservas").child(fecha).child(hora).
addValueEventListener(vel);
        }


    }
```

```
private void addListener2() {

        removeListener();

        if (vel == null) {
            vel = new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                    restaurante = dataSnapshot.getValue(Restaurante.class);

                    if (contAforo > restaurante.getAforo()) {

                        Snackbar snackbar = Snackbar

.make(getActivity().getWindow().getDecorView().getRootView(), R.string.aforo_completo,
Snackbar.LENGTH_LONG)

                    .setBackgroundTint(getResources().getColor(R.color.orange_dark));

             snackbar.setAnimationMode(BaseTransientBottomBar.ANIMATION_MODE_SLIDE);
                        snackbar.setAnchorView(R.id.btnBookingFragReservar);
                        snackbar.show();

                    } else {

                        addListener3();

                    }

                }

                @Override
                public void onCancelled(@NonNull DatabaseError error) {
                    Snackbar snackbar = Snackbar

                    .make(getActivity().getWindow().getDecorView().getRootView(),
                    R.string.error_carga_datos, Snackbar.LENGTH_LONG)

                    .setBackgroundTint(getResources().getColor(R.color.orange_dark));

             snackbar.setAnimationMode(BaseTransientBottomBar.ANIMATION_MODE_SLIDE);
                        snackbar.setAnchorView(R.id.btnBookingFragReservar);
                        snackbar.show();
                    }
                };
                dbRef.child("restaurantes").child(restUID).addValueEventListener(vel);
        }

    }
```

4. Interior de *addListener3* en línea 217 de *BookingFragment.java*.

```
            //Log.i("ERROR3 FECHA", fechaArray);
               Log.i("ERROR2 HORA", horaArray + " " + fechaArray);

               for (DataSnapshot dss: dataSnapshot.getChildren()) {

                   //reservaRest = dss.getValue(ReservaRest.class);

                   reservaRest = null;
                   reservaRest = new ReservaRest();


               reservaRest.setFecha(dss.getValue(ReservaRest.class).getFecha());
                   Log.i("ERROR", reservaRest.getFecha());

               reservaRest.setHora(dss.getValue(ReservaRest.class).getHora());
                   Log.i("ERROR", reservaRest.getHora());

            reservaRest.setUserUID(dss.getValue(ReservaRest.class).getUserUID());
                   Log.i("ERROR", reservaRest.getUserUID());

            reservaRest.setNomUsu(dss.getValue(ReservaRest.class).getNomUsu());
                   Log.i("ERROR", reservaRest.getNomUsu());

        reservaRest.setNumPersonas(dss.getValue(ReservaRest.class).getNumPersonas());
                   Log.i("ERROR", String.valueOf(reservaRest.getNumPersonas()));
                   reservaRest.setCodigo(dss.getKey());
                   Log.i("ERROR", reservaRest.getCodigo());

                   listaReservas.add(reservaRest);
                   Log.i("ERROR LISTA", String.valueOf(listaReservas.size()));

               }

               removeListener();
```

5. *onActivityResult* en línea 71 de *CheckQRFragment.java*.

```
@Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
              IntentResult result = IntentIntegrator.parseActivityResult(requestCode,
      resultCode, data);
        if (result != null) {
            if (result.getContents() != null) {
                btnEscanear.setVisibility(View.GONE);
                imResult.setImageDrawable(ResourcesCompat.getDrawable(getResources(),
R.drawable.ic_check_100, null));
                String resultConcat =
getResources().getString(R.string.reserva_verificada) + "\n ID: " +
result.getContents();
                tvCodigo.setText(resultConcat);
                tvCodigo.setTextColor(getResources().getColor(R.color.green_light));
                codigdoReservaUsu = String.valueOf(result.getContents());
                btnValidar.setEnabled(true);

            } else {
                tvCodigo.setText(getResources().getString(R.string.error_lectura_qr));
                tvCodigo.setTextColor(getResources().getColor(R.color.orange_dark));
            }
        }
    }
```

B. Código XML

1. Archivo *colors.xml*.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="green_dark">#264653</color>
    <color name="green_light">#2A9D8F</color>
    <color name="yellow">#E9C46A</color>
    <color name="yellow_dark">#b4943c</color>
    <color name="orange_dark">#E76F51</color>
    <color name="orange_light">#F4A261</color>
    <color name="white_light_alpha">#CCFFFFFF</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
    <color name="white_alpha">#4DFFFFFF</color>

</resources>
```

2. Tema principal de la aplicación en línea 23 de *themes.xml*

```
<style name="Theme.SafeBar.NoActionBar"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/yellow</item>
        <item name="colorPrimaryVariant">@color/orange_dark</item>
        <item name="colorOnPrimary">@color/white</item>
        <item name="colorPrimaryDark">@color/yellow_dark</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/green_light</item>
        <item name="colorSecondaryVariant">@color/green_dark</item>
        <item name="colorOnSecondary">@color/white</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryDark</item>
        <!-- Customize your theme here. -->
    </style>
```

3. Archivo *bottom_navigation_bar.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.material.bottomnavigation.BottomNavigationView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/bottomNavigationBar"
    android:background="?android:attr/windowBackground"
    app:menu="@menu/main_menu_bottom">

</com.google.android.material.bottomnavigation.BottomNavigationView>
```