

PROYECTO INTEGRADO:

ERP

PARA HOSTELERÍA

CURSO 2021-2022

2º DAM

LIDIA FERNÁNDEZ MARTÍNEZ



ÍNDICE

INFORMACIÓN.....	3
Resumen y descripción del proyecto	3
Motivación del proyecto	3
Herramientas utilizadas	4
• Hardware	4
• Software	4
Módulos empleados	5
• Primero	5
• Segundo.....	5
Investigación	6
• Maven	6
• Drag and drop.....	6
• Conexión y acceso a Firebase a través de una aplicación de escritorio.....	7
• Gestión de roles de usuarios.....	7
• Generar un archivo .exe	7
• Bases de datos no relacionales, RealTime DataBase	7
• Múltiples tablas y control de relaciones mediante código	7
• Añadir elementos extra a un recycler.....	8
• Generar un PDF.....	8
ANÁLISIS	8
Usuarios potenciales	8
Requisitos funcionales	8
PLANIFICACIÓN	9
Planificación del tiempo de desarrollo de la aplicación	9
ESTRUCTURA GENERAL DEL PROGRAMA	9
Estructura de paquetes.....	9
Diseño	10
• Pantallas.....	10
• Iconos	22
• Normas de usabilidad aplicadas.....	25
Gradle,Manifest y Strings	26
• Gradle.....	26
• Manifest	27
• Strings.....	27

BASE DE DATOS	27
Estructura de la base de datos.....	27
Diagrama de la base de datos	28
CONTROL DE ERRORES Y EXCEPCIONES	29
APLICACIÓN DE GESTIÓN DE USUARIOS	29
Diseño	30
Normas de usabilidad aplicadas	33
PROBLEMAS ENCONTRADOS Y SOLUCIONES APORTADAS	33
FUTURAS MEJORAS	33
BIBLIOGRAFÍA	34

INFORMACIÓN

Resumen y descripción del proyecto

AlmeriRest es una aplicación móvil sobre gestión de servicios de hostelería, como: restaurantes, bares, etc.

Orientada para venderse directamente a los comercios a la que va destinada, se puede utilizar como recurso para la automatización de pedidos y comandas y del servicio realizado.

Un ERP orientado a la hostelería es un software para llevar de forma integrada un restaurante, cafeterías y bares.

AlmeriRest consta de cinco partes diferentes:

- **Una gestión de familias** donde se agruparán los distintos productos ofrecidos, a los cuales se puede realizar un CRUD.
- **Una gestión de artículos** donde se realizará el CRUD de los diferentes artículos asociados a una familia existente.
- **Una gestión de salones** donde se realizará el CRUD de las diferentes mesas.
- **Un TPV** donde se controlan las comandas y las consumiciones y tickets de las diferentes mesas.
- **Una gestión de pedidos** donde se podrá acceder a las comandas realizadas en cada mesa en la TPV.
- **Una gestión de usuarios** que se desarrollará en una aplicación secundaria complementaria a la aplicación principal en la que se controlarán los roles y los permisos de los usuarios.

Motivación del proyecto

Este proyecto supone la finalización del Ciclo Superior en Desarrollo de Aplicaciones Multiplataforma (DAM), donde se pondrá en práctica en un proyecto real todo lo aprendido durante los dos años anteriores.

Para ello he decidido crear un ERP enfocado en el sector hostelero aprovechando una idea propuesta en la empresa en la que estamos realizando los estudios en formación DUAL y las prácticas FCT.

Lo que más me ha llevado a decantarme por esta opción sobre otras analizadas, es que es un proyecto que supone varios retos, es una aplicación muy útil en el sector hostelero y que en un futuro puede ser comercializado.

Debido a esto último se ha realizado un estudio y se han buscado los TPV más utilizados en el momento:

- **Cuiner:** integra una amplia gama de soluciones a través de sus diferentes módulos. El módulo de gestión de reservas que funciona 24 horas, sincroniza las reservas realizadas desde la web, TPV, tablet o Smartphone. Cuiner dispone de facturación con datos fiscales.
- **Sysme:** se pueden controlar las solicitudes abiertas, asignarlas a una mesa o sector de la barra. El entorno visual es muy amigable y permite identificar solicitudes, añadir más productos a la cuenta y cobrar
- **Glop:** está pensado para pantallas táctiles. Tiene una interfaz muy visual y cómoda para cualquier usuario. Se estructura a través de menús, fácilmente identificables y operables. El apartado utilidades, permite gestionar en la misma pantalla venta y cobro de un mismo artículo en diferentes porciones o sabores.
- **OfiBarman:** es un software para bares flexible y adaptado al tamaño y perfil de cada establecimiento. Es una herramienta pensada para el ritmo que este tipo de negocios impone. Los productos, agrupados en familias, se visualizan e identifican rápidamente para seleccionarlos en la pantalla táctil.

Herramientas utilizadas

Como herramientas principales empleadas para el desarrollo del proyecto se encuentran, el IDE AndroidStudio y el lenguaje de programación Kotlin, lenguaje de programación de código abierto creado por JetBrains y actualmente es el más empleado por los programadores para el desarrollo y programación de aplicaciones Android. También, se hace uso de una conexión inalámbrica a Internet para hacer conexión con la base de datos de Firebase.

Por otro lado, se utiliza el lenguaje de programación Java y el IDE Netbeans para crear la interfaz gráfica y dar funcionalidad a la aplicación de escritorio de la gestión de los usuarios.

- **Hardware**

- Ordenador marca DELL con un procesador Intel® core™ i5-4210U, una memoria RAM de 8 y sistema de 64 bits.
- Móvil marca Huawei y modelo P30 Lite.

- **Software**

- Windows 8.1
- Android 11.01.11
- Conexión a Internet mediante Chrome.
- En Firebase apartado de “Authentication” con correo electrónico/ contraseña y Google y “RealTime DataBase”.

- NetBeans 13.

Módulos empleados

- **Primero**

Sistemas informáticos:

- Instalación, configuración y uso del SO Windows 10 pro.

Bases de Datos:

- Creación y uso de una base de datos no relacional.
- Creación acceso y modificación de los datos de la base de datos.
- Diagramas de base de datos.

Entornos de desarrollo:

- Documentación interna del código.
- Instalación, configuración y uso de AndroidStudio 11.01.11.
- Instalación, configuración y uso de NetBeans 13.

Lenguaje de marcas y sistemas de gestión de información:

- Empleo del lenguaje XML.

Programación:

- Lenguaje de Java.

- **Segundo**

Programación de servicios y procesos:

- Creación y uso de hilos y procesos.
- Uso de Java Mail.

Desarrollo de interfaces:

- Uso de conceptos de usabilidad en el diseño de la interfaz gráfica.
- Diseño de una interfaz gráfica.
- Documentación interna y externa del proyecto.
- Instalación y configuración de Maven.
- Uso de Java swing para el diseño de la interfaz gráfica.

Programación multimedia y dispositivos móviles:

- Empleo de AndroidStudio.
- Uso del lenguaje Kotlin.
- Uso de la base de datos Relatime Database.

Sistemas de gestión empresarial:

- Creación y desarrollo de un ERP.
- Documentación interna del código.

Investigación

A continuación, se explicarán las distintas partes de investigación que se han realizado:

- **Maven**

Para la aplicación de gestión de usuarios, se ha investigado como crear la integración en un proyecto Maven de Firebase y todas sus librerías y JavaFX. Se buscaba acceder a la base de datos que se utiliza en la aplicación principal y ofrecer la mayor posibilidad que ofrece Java con respecto a la creación y diseño de interfaces gráficas. Para esto último, también se intentó conectar el proyecto al software SceneBuilder. Pero desafortunadamente, después de varias semanas intentando que funcionara y preguntando a la profesora de Desarrollo de Interfaces, se vio que no era posible la combinación de todos estos elementos, y al final, se optó por hacer un proyecto Maven con Java swing, que si permite dicha combinación de Maven y Firebase.

- **Drag and drop**

Consiste en un conjunto de métodos que permiten interactuar las imágenes que se encuentren en pantalla desplazándolas a otra posición siendo esta una concreta. Por falta de tiempo, no pudo llegar a ser implementado en la aplicación. Se tenía pensado utilizarlo en el apartado de montar salones de forma gráfica.

- **Conexión y acceso a Firebase a través de una aplicación de escritorio**

Desarrollo y creación de una conexión a Firebase desde una aplicación de escritorio y, para ello, se emplea el lenguaje de Java.

- **Gestión de roles de usuarios**

Se realizará una segunda aplicación en la que se controlará de forma segura la validación de usuarios, ya que estos, solo podrán acceder a la aplicación principal si son validados por el administrador. Por otro lado, dependiendo de rol que tenga el usuario, se les dará acceso a distintas partes de la aplicación principal.

- **Generar un archivo .exe**

Haciendo uso del software Launch4j y a partir del .jar de la aplicación secundaria, se genera un archivo .exe de esta aplicación.

- **Bases de datos no relacionales, RealTime DataBase**

Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Todos los usuarios comparten una instancia de Realtime Database y reciben actualizaciones automáticamente con los datos más recientes.

- **Múltiples tablas y control de relaciones mediante código**

Con esta base de datos se hará uso de múltiples tablas como: salones, familias, artículos...

RealTime DataBase es NoSQL, por lo que el CRUD que se realiza en la base de datos, debe ser controlado a través de código, además de controlar también las relaciones, consistencia y ausencia de redundancia.

- Añadir elementos extra a un recycler

Añadir a los recycler ya establecidos de la aplicación principal nuevos elementos extra y dar funcionalidad a estos, como, por ejemplo, spinners o botones. Como surgieron algunos problemas con estos, no se llegaron a agregar.

- Generar un PDF

Se genera un PDF en la aplicación secundaria con la lista de los usuarios existentes en la base de datos.

Se tenía pensado poder generar un PDF con la factura del pedido en la aplicación principal, pero debido a que no se llegaron a solucionar errores y no llegaba a funcionar se descartó añadirlo a la aplicación.

ANÁLISIS

Usuarios potenciales

Los usuarios a los que van destinados este conjunto de aplicaciones son las empresas del sector hostelero, ya que, les permiten automatizar el proceso de pedidos de los clientes y la posterior gestión del ticket con lo consumido. Además de poder controlar a los distintos empleados que utilizan el ERP.

Requisitos funcionales

El usuario puede registrarse si es la primera vez que entra en la aplicación, y si ya posee una cuenta puede entrar identificándose.

En la pantalla principal una vez que ha sido autorizado por el administrador se encuentran las opciones de la app a las que se tiene acceso según el rol que se le haya asignado al usuario, estas se muestran de forma sencilla e intuitiva.

Tiene varias opciones a explorar según el rol:

- Rol inicial: El usuario visualiza una pantalla con un mensaje en el que se indica que debe esperar a que el administrador le asigne el rol.
- Rol camarero: El usuario solo tendrá acceso a la gestión mediante la TPV, podrá hacer comandas y obtener el coste de estas.
- Rol total: El usuario tendrá acceso total a toda la app incluye además de la TPV la gestión de las familias, artículos y salones.

En cada una de las pantallas de la app se podrá acceder a las diferentes opciones y pantallas de forma sencilla, bajo el empleo de las normas de usabilidad.

PLANIFICACIÓN

Planificación del tiempo de desarrollo de la aplicación

Se comienza analizando la idea propuesta y el posible desarrollo de esta. Después se decide donde se almacenará la base de datos que, posteriormente, se ha pensado y diseñado. Se utilizará finalmente Firebase RealTime DataBase para almacenar los datos en cloud.

Lo primero que se empieza en el desarrollo propio de la aplicación inicial, es el diseño e interfaz visual. Seguidamente, se realiza la parte de la lógica relacionada con la base de datos.

En el siguiente diagrama de Gantt, se muestra de forma gráfica como se ha ido elaborando ambas aplicaciones:

Nº Semanas / Tareas	1	2	3	4	5	6	7	8	9	10	11
Idea del proyecto											
Investigación											
Elaboración pantalla login											
Creación pantalla inicial											
Creación familias											
Creación artículos											
Creación salones											
Creación TPV											
Creación 2ª aplicación gestión roles											

ESTRUCTURA GENERAL DEL PROGRAMA

Estructura de paquetes

Dentro de la raíz principal tengo los archivos Kotlin MainActivity(pantalla principal) y SplashActivity (pantalla de carga). A continuación, tengo estructurado el proyecto en los siguientes paquetes:

- Artículos: se encuentra la lógica y funcionalidad de los artículos.
- Clases auxiliares: archivos Kotlin que son de apoyo y de elementos auxiliares al resto de clases.
- Email: se encuentra la lógica para enviar un email desde la aplicación.
- Familias: se encuentra la lógica y funcionalidad de las familias.
- Login: se encuentra la lógica para registrarse a la aplicación, o acceder en caso de ya estar registrado.

- Pedido: se encuentra la lógica relacionada con los pedidos que se han realizado en la TPV.
- Recycler_articulos: archivo adapter y dataClass de los artículos.
- Recycler_articulos_tpv: archivo adapter y dataClass de los artículos en la TPV.
- Recycler_familias: archivo adapter y dataClass de las familias.
- Recycler_pedidos: archivo adapter y dataClass de los pedidos realizados.
- Recycler_salon: archivo adapter y dataClass de los salones.
- Recycler_user: archivo adapter y dataClass de los usuarios.
- Salon: se encuentra la lógica y funcionalidad de la gestión de salones.
- Tpv: se encuentra la lógica y funcionalidad de la Tpv.

Diseño

- **Pantallas**

Fondo de pantalla y logo del SplashActivity, que aparece mientras se carga la aplicación. Se utiliza el logo de la empresa en la que se estaba realizando la formación DUAL y la FCT.



Interfaz del LoginActivity donde debemos introducir un email y la contraseña que elijamos. Pulsaremos el botón registrarse si no hemos accedido nunca a la aplicación, el botón acceder si ya estamos registrados y el botón limpiar si queremos borrar lo escrito en los campos.



Interfaz del MainActivity que aparece si el usuario posee el rol -1. Deberá esperar a que se le den permisos con la aplicación secundaria. El botón salir permite cerrar la sesión y salir de la aplicación.



Interfaz del MainActivity que aparece si el usuario posee el rol 0. El usuario solo tendrá acceso a la TPV y a los pedidos que se hagan en esta.



Interfaz del MainActivity que aparece si el usuario posee el rol 1. Es el mayor rol disponible y se tiene acceso a todas las partes de la aplicación:

- El botón Familias nos permitirá acceder a la gestión de las familias. Estas son la manera de clasificar por grupos los diferentes artículos que se añadan.
- El botón Artículos nos permitirá acceder a la gestión de los artículos que se dispongan.
- EL botón Gest.Salones permite registrar mesas a la aplicación.
- El botón TPV permite tomar comandas de las mesas disponibles.
- El botón Pedidos muestra las comandas realizadas desde la TPV por mesas.



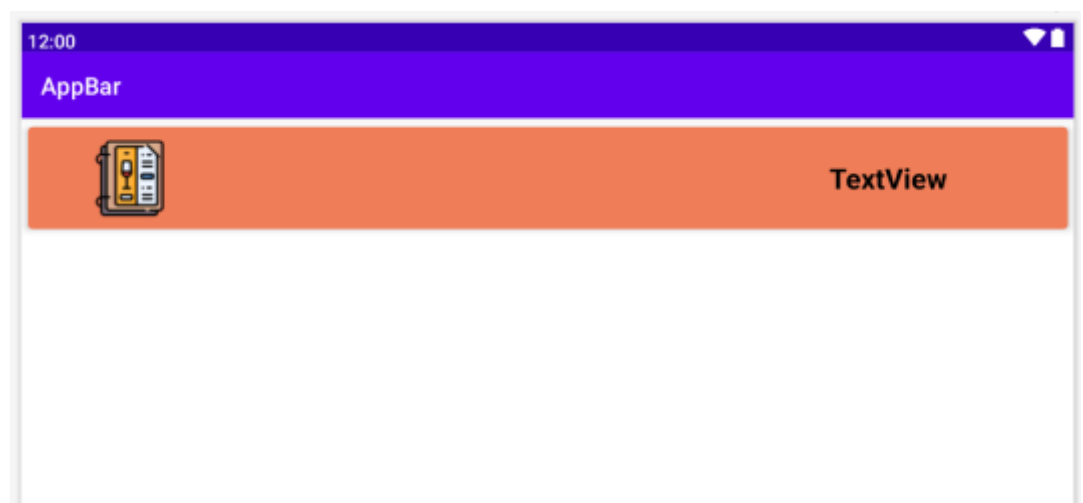
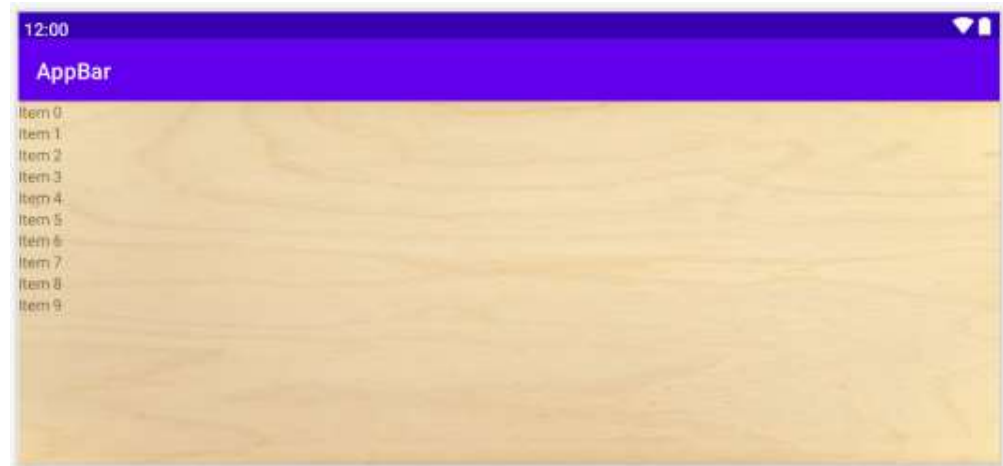
Interfaz del menú principal del apartado familias. Donde se podrán crear familias o acceder a las ya existentes. El botón salir nos permite volver al menú principal (MainActivity).



Interfaz del activity donde se crean las familias. Se debe introducir el nombre de la familia y en el spinner elegir la imagen que queremos darle a esa familia. El botón guardar familia guardará la familia, el botón limpiar borrará lo escrito en el campo de texto y el botón volver nos llevará al menú de las familias.



Interfaz del recycler donde se muestran las familias que se han creado y el Cardview donde se muestran.



Interfaz donde se modifica una familia. En el campo de texto aparecerá el nombre de la familia seleccionada, y en el spinner la imagen que posee. El botón modificar familia cambiará los datos con los nuevos introducidos y el botón volver nos llevará a la lista de familias existentes.



Interfaz del menú principal de artículos. Donde se podrán crear artículos o acceder a los ya existentes. El botón salir nos permite volver al menú principal (MainActivity).



Interfaz del activity donde se crea un nuevo artículo. Se debe introducir el nombre del artículo y su precio. Además, en el spinner hay que elegir la familia en la que queremos introducir el artículo. Solo aparecerán las familias que hayan sido creadas e introducidas manualmente. El botón guardar artículo guardará el artículo, el botón limpiar borrará lo escrito en los campos de texto y el botón volver nos llevará al menú de los artículos.

12:00

AppBar

Indica el nombre del artículo

Indica el precio del artículo

Seleccione una familia a la pertenece el artículo

GUARDAR ARTÍCULO

LIMPIAR

VOLVER

Interfaz del recycler donde se muestran los artículos y el Cardview donde se muestran.

12:00

AppBar

Item 0

Item 1

Item 2

Item 3

Item 4

Item 5

Item 6

Item 7

Item 8

Item 9

12:00

AppBar

ImageView

TextView

TextView

Interfaz donde se modifica un artículo. En los campos de texto aparecerá el nombre del artículo seleccionado y su precio, y en el spinner la familia a la que pertenece el artículo. El botón modificar artículo cambiará los datos

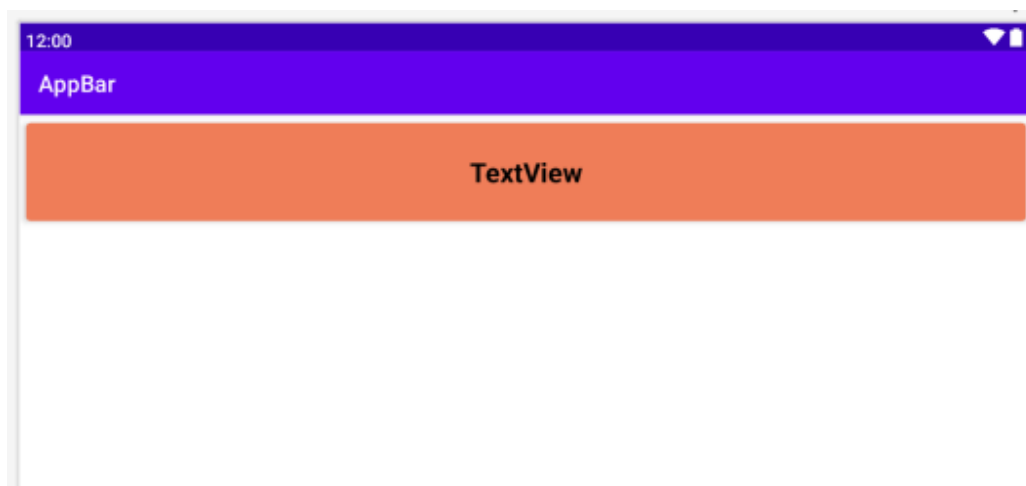
con los nuevos introducidos y el botón volver nos llevará a la lista de artículos existentes.

Interfaz del menú principal de salones. Donde se podrán crear mesas o acceder a las ya existentes. El botón salir nos permite volver al menú principal (MainActivity).

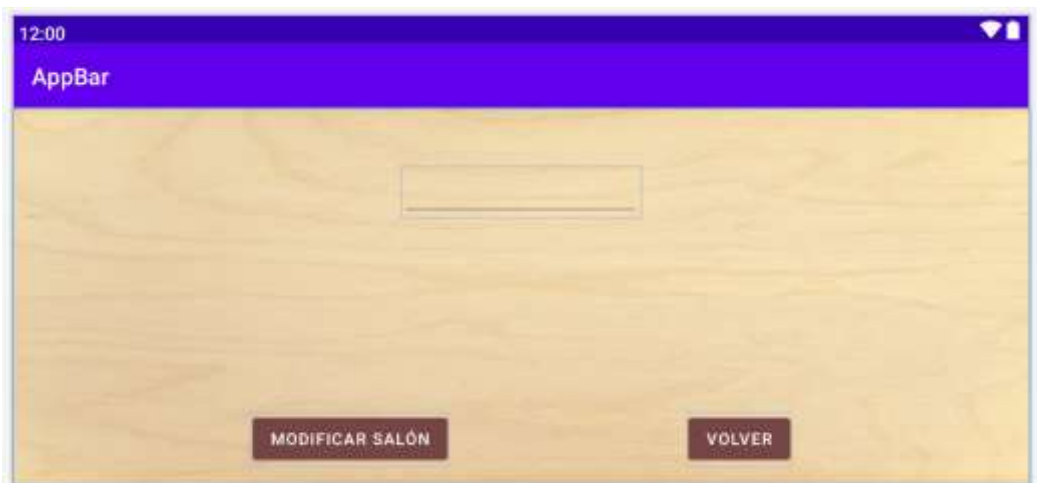
Interfaz del activity donde se crea un nuevo salón. Se debe introducir el nombre de la mesa. El botón guardar salon guardará la mesa, el botón limpiar borrará lo escrito en el campo de texto y el botón volver nos llevará al menú de los salones.



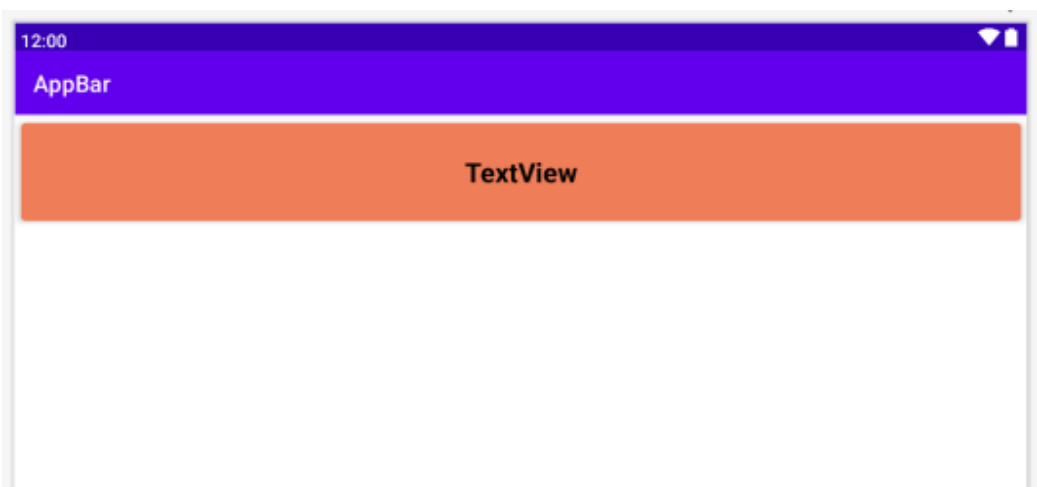
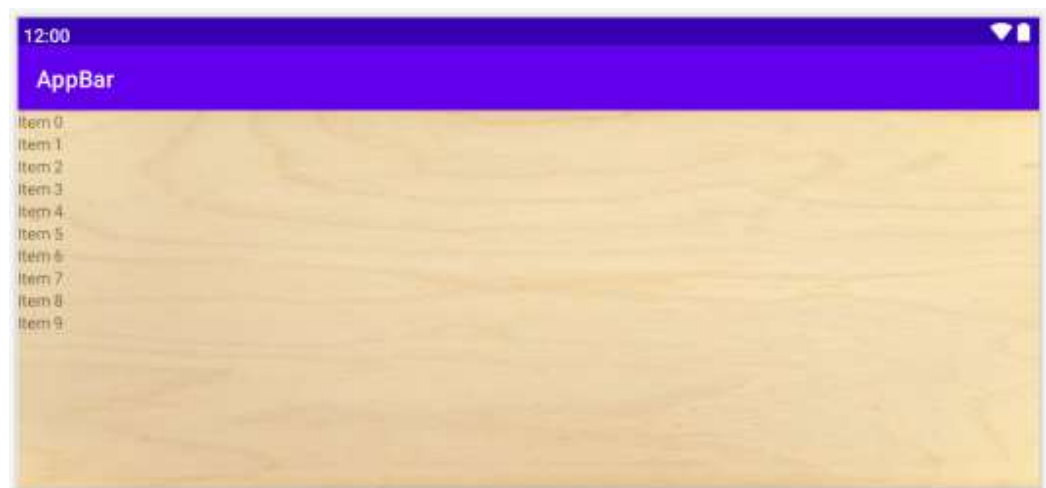
Interfaz del recycler donde se muestran las mesas y el Cardview donde se muestran.



Interfaz donde se modifica un salón. En el campo de texto aparecerá el nombre de la mesa seleccionada. El botón modificar salón cambiará los datos con los nuevos introducidos y el botón volver nos llevará a la lista de mesas existentes.



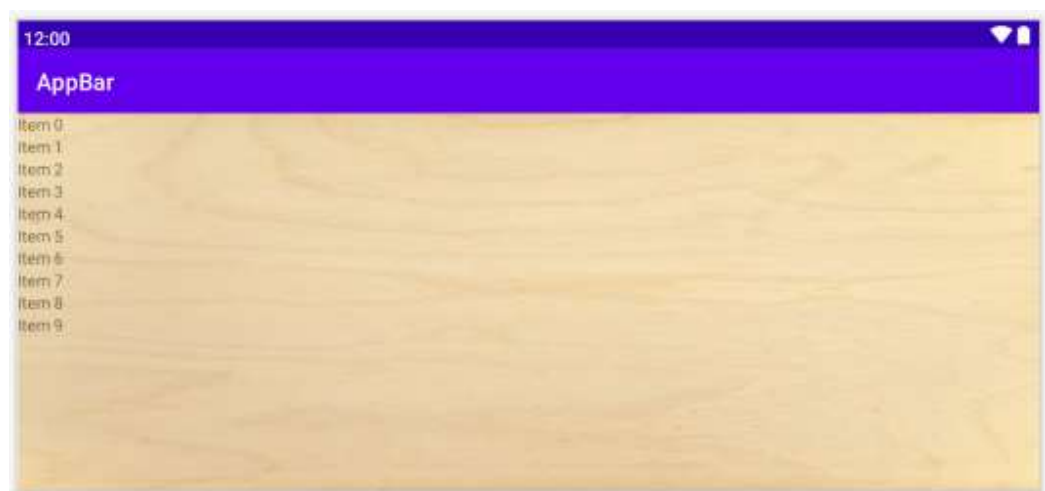
Interfaz inicial de la TPV (recycler de salones).

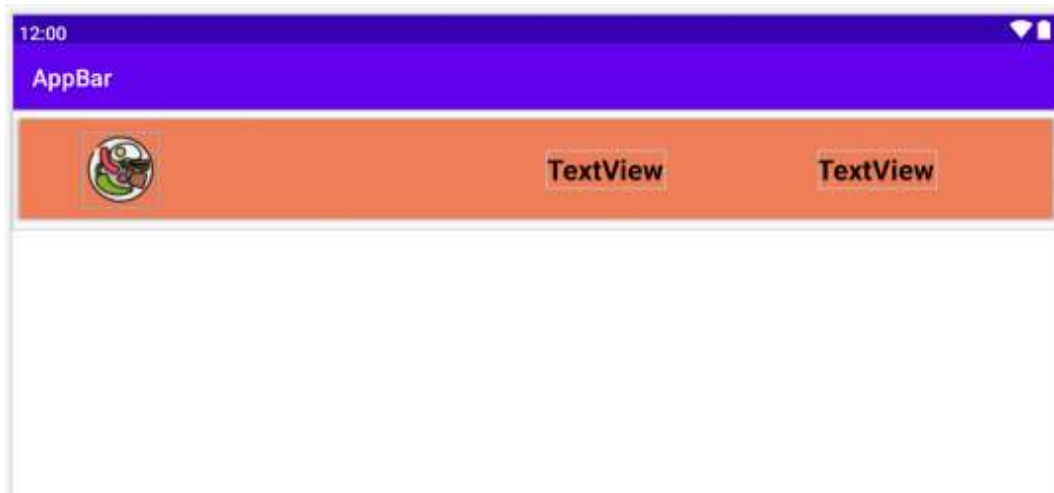


Segunda interfaz de la TPV (recycler familias).



Tercera interfaz de la TPV (recycler artículos).



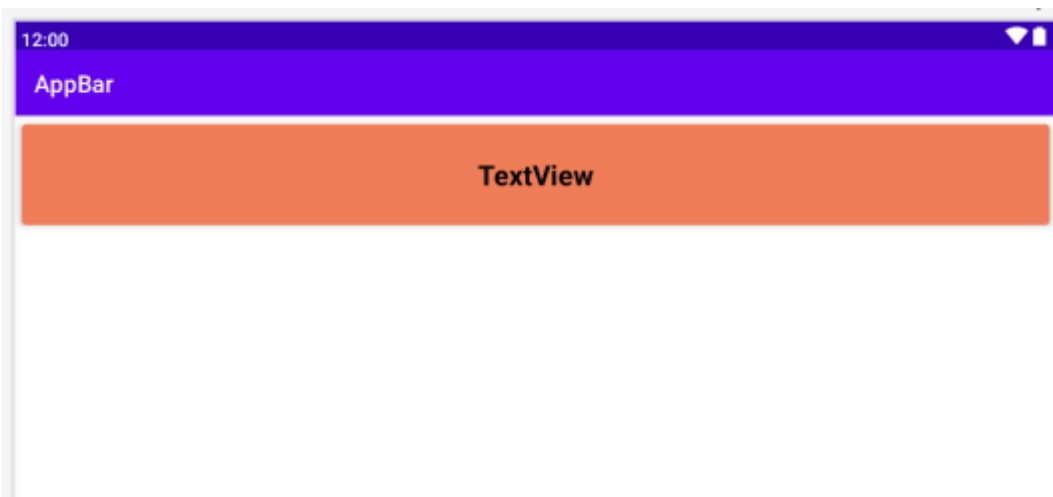


Cuarta interfaz de la TPV al seleccionar un artículo para guardar la línea de pedido. Aparecerán los datos del artículo seleccionado. Se deberá elegir la cantidad del artículo para que nos aparezca el botón guardar comanda. El botón guardar comanda guardará el pedido y el botón volver nos llevará a la primera interfaz de la TPV (recycler de salones).

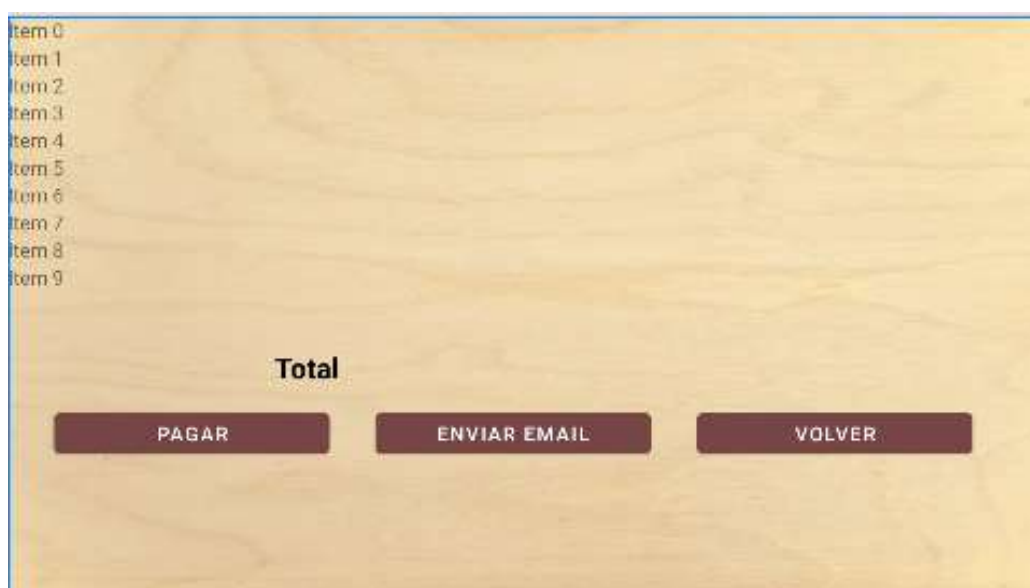


Interfaz inicial del apartado pedidos (recycler salones).

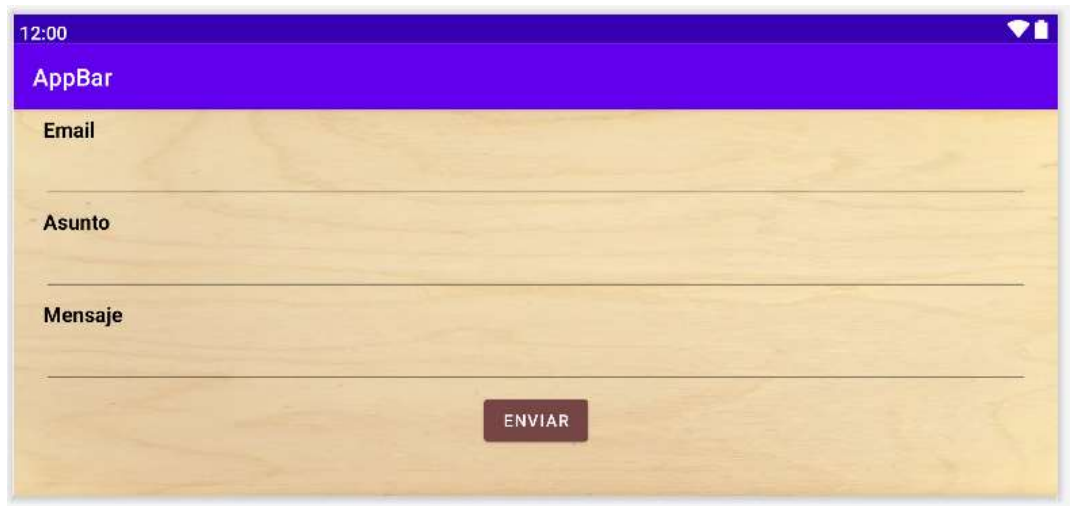




Segunda interfaz de los pedidos dentro de cada mesa. Aparecerán las CardView con los artículos que se han pedido en esa mesa, además del precio total de la cuenta, un botón para pagar, un botón para enviar un email con la factura y un botón para volver a las mesas.



Interfaz para enviar un email. Envía un email con la factura. Debido a que Google desactivó la opción de poder enviar correos desde terceros no se puede realizar esta funcionalidad.



The screenshot shows a mobile application interface for sending an email. At the top, there is a purple header bar with the text 'AppBar'. Below this, the interface has a light wood-grain background. It contains three text input fields labeled 'Email', 'Asunto', and 'Mensaje'. At the bottom center, there is a dark red button with the text 'ENVIAR'.

- Iconos

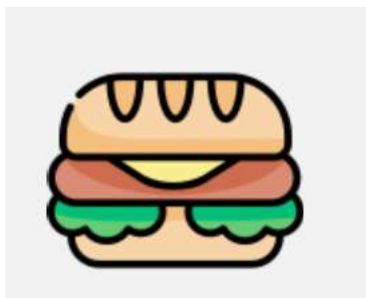
Familias



Familia sin categoría



Familia de bocadillos



Familia de cafés



Familia de refrescos



Familia de postres



Artículos



Salones



TPV



Pedidos



Fondo general de la aplicación



- **Normas de usabilidad aplicadas**

- Se utilizan colores y un fondo de pantalla que no dificultan el uso ni la lectura de los textos y amenizan el uso de esta.
- Uso de iconos descriptivos y que se relacionan con la acción que está realizando el usuario.
- El nombre de la aplicación es descriptivo para que el usuario sepa de qué se trata la aplicación.
- Uso de formas sencilla y amigables para el usuario además de un lenguaje amigable y cotidiano.
- Al usuario se le indica en todo momento si ha cometido algún error o si es necesario que cambie algo para poder seguir avanzando.
- El usuario sabe en todo momento lo que está pasando gracias a mensajes de advertencia y mensajes informativos.

A continuación, explico las reglas de usabilidad de Jacob Nielsen que se han aplicado al proyecto:

1. **Visibilidad:** Explica al usuario cuál es el estado del sistema en cada momento, y lo mantiene informado de lo que está pasando.
2. **Relación con la realidad:** Se utiliza un lenguaje familiar y cotidiano para los usuarios que utilizarán la aplicación. Además, se organiza la información con un orden natural y lógico.
3. **Control y libertad:** Ofrece funciones de rehacer y deshacer que permitan al usuario tener total libertad a la hora de interactuar con la aplicación.
4. **Consistencia y estándares:** Establecer unas convenciones lógicas y mantenerlas siempre (mismo lenguaje, mismo flujo de navegación).
5. **Prevención de errores:** Ayuda a los usuarios a evitar que se equivoquen antes de que cometan el error.
6. **Reconocimiento:** Se hace visible todo lo que sea posible, no se espera que los usuarios recuerden o memoricen información. Esta se muestra en todo momento.
7. **Flexibilidad:** Permite que el sistema pueda adaptarse a los usuarios frecuentes.
8. **Estética y minimalismo:** Solo se muestra lo necesario y relevante en cada acción, no se añade ninguna información extra o innecesaria.
9. **Recuperarse de los errores:** Ayuda a los usuarios a reconocer y corregir sus errores, indica siempre el problema concreto que está ocurriendo y sugiere soluciones constructivas.
10. **Ayuda y documentación:** La información de ayuda debe ser breve, concisa, fácil de buscar y enfocada a las tareas del usuario.

Gradle,Manifest y Strings

- **Gradle**

A continuación, se muestra el contenido del gradle que corresponde a la parte que indica las librerías que se han utilizado en la aplicación:

```
//Para firebase la BD
implementation platform('com.google.firebase:firebase-bom:29.2.1')
implementation 'com.google.firebase:firebase-analytics-ktx'
implementation 'com.google.firebase:firebase-database-ktx:20.0.3'
```

```
implementation 'com.google.firebase:firebase-database:20.0.3'  
implementation 'com.google.android.gms:play-services-auth:20.1.0'
```

```
//Para cargar imagenes  
implementation 'com.squareup.picasso:picasso:2.71828'  
implementation 'com.google.firebase:firebase-auth-ktx:21.0.3'
```

```
//Para usar Java mail  
implementation files('libs\\activation.jar')  
implementation files('libs\\additionnal.jar')  
implementation files('libs\\mail.jar')
```

- **Manifest**

Fichero xml en el que se encuentran las rutas de los paquetes y clases de la aplicación. Aquí se ha indicado que pantalla será la que se lance primero y que todos los activities tengan una orientación horizontal.

Además, se han otorgado a la aplicación permisos para acceder a Internet y para leer y escribir en ficheros.

- **Strings**

Fichero xml en el que se han ido escribiendo todas las cadenas de texto que se han utilizado en la aplicación asignándolas a elementos como textView, recycler...

También se ha añadido un fichero String.xml para poder traducir todo este texto a un idioma distinto (en este caso el inglés) para que cuando el usuario cambie el idioma de su móvil también o haga la aplicación.

BASE DE DATOS

Estructura de la base de datos

La base de datos se compondrá de 5 tablas:

- **Familias:** En esta tabla se almacenarán las familias principales. Estará formada por los siguientes campos:
 - Id (Int).
 - NombreFamilia (String).
 - UrlImagenFamilia (Long).

- **Artículos:** En esta tabla se almacenarán todos los artículos que se vayan a vender. Estará formada por los siguientes campos:
 - Id (Int).
 - NombreArticulo(String).
 - Precio(Double).
 - IdFamilia (Int).
 - UrlImagenFamilia (Long).

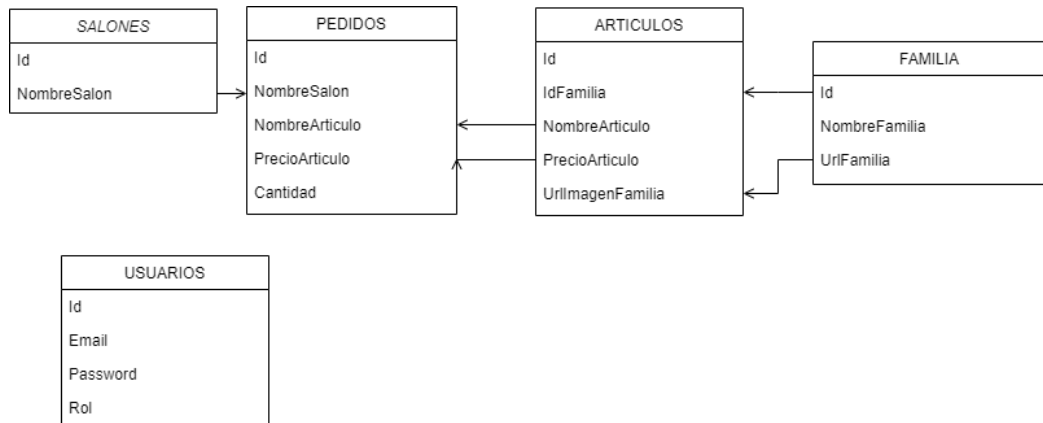
- **Salones:** En esta tabla se almacenarán los distintos salones que existan en el establecimiento. Estará formada por los siguientes campos:
 - Id (Int).
 - NombreSalon (String).

- **Pedidos:** En esta tabla se almacenarán los distintos pedidos que realicen los clientes. Estará formada por los siguientes campos:
 - Id (Int).
 - IdSalon (Int).
 - IdArticulo (Int).
 - PrecioArticulo (Double).
 - CantidadArticulo (Int).

- **Usuarios:** En esta tabla se almacenarán los distintos usuarios que se registren en la aplicación. Estará formada por los siguientes campos:
 - Id (Int).
 - IdSalon (Int).
 - IdArticulo (Int).
 - PrecioArticulo (Double).
 - CantidadArticulo (Int).

Diagrama de la base de datos

A continuación, se muestra la estructura de la base de datos explicada anteriormente en forma visual a través de un diagrama. Aunque se utiliza una base de datos no relacional, se representan las relaciones que existen (que se controlan mediante el código), en forma gráfica, utilizando flechas.



CONTROL DE ERRORES Y EXCEPCIONES

A lo largo de la aplicación existen algunos mensajes que indican que el usuario no está realizando alguna acción como debería. Para evitar arrastrar errores e incoherencias en la aplicación se le indica al usuario que no está haciendo algo bien. Algunas situaciones como estas son:

- En la pantalla de Login muestra un mensaje de error si no hay conexión a Internet o si se ha escrito alguno de los datos mal.
- En todas las pantallas en las que el usuario debe escribir algún texto, se controla que esos campos no se queden vacíos, en el caso de que el usuario los deje así, no se le dejará avanzar y se le indicará su error.
- Para salir de la aplicación se muestra un texto en el que se pregunta al usuario si de verdad quiere realizar esta acción, para prevenir al usuario de que realice una acción equivocada.
- Al intentar borrar una familia, no se permite que se borre si existen artículos asociados a esa familia para evitar incoherencias.
- Antes de borrar cualquier elemento, se pregunta al usuario si quiere hacerlo, para evitar posibles equivocaciones por parte de los usuarios.

APLICACIÓN DE GESTIÓN DE USUARIOS

La segunda aplicación que se ha desarrollado para el proyecto integrado, es una aplicación de escritorio realizada con el IDE Netbeans y en lenguaje Java orientada para el uso de un administrador. Consiste en una ventana formada por una tabla y tres botones con los que se podrá realizar una gestión de los usuarios que se han registrado en la aplicación principal. Las distintas acciones que se pueden realizar son las siguientes:

- **Consultar usuarios:** El administrador podrá realizar una consulta de los usuarios que existen registrados en la base de datos de Firebase. Estos le aparecerán en la tabla de la aplicación. Cuando el usuario pulse esta opción, el botón desaparecerá de la interfaz, pero, aun así, si se añaden

usuarios o se eliminan en la aplicación principal se seguirán reflejando estos cambios en esta aplicación.

- **Modificar rol:** El administrador deberá seleccionar un usuario y después este botón donde podrá cambiar el rol de ese usuario seleccionado.
- **Eliminar usuario:** El administrador podrá eliminar usuarios de la base de datos.

Diseño

A continuación, se mostrará de forma gráfica como se ve la interfaz en sus distintas partes:

Así se verá la aplicación nada más abrirla.



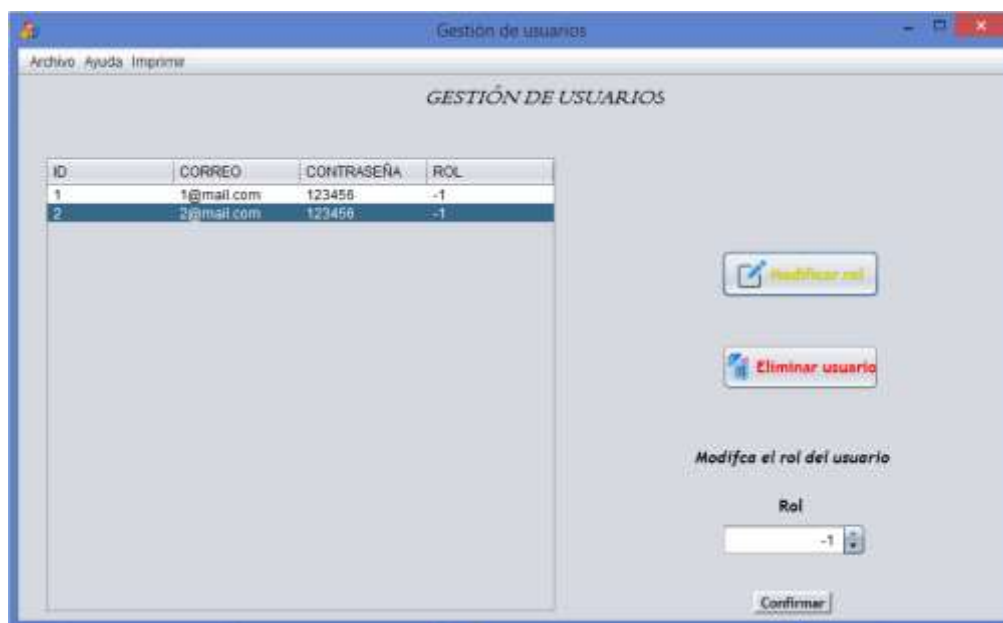
En cuanto pinchemos en el botón de consultar usuarios, aparecerán los usuarios en la tabla y desaparecerá el botón anterior. Esto último se hace para evitar problemas con la conexión a la base de datos. Además, aparecerán los botones restantes.



Si queremos modificar el rol de un usuario deberemos pinchar en las filas disponibles, sino, nos aparecerá el siguiente mensaje.



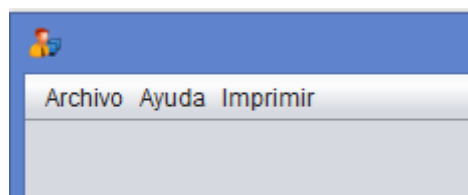
Una vez seleccionada la fila, nos aparecerá la opción para cambiar el rol del usuario. Se cambiará en cuanto pulsemos el botón confirmar.



Por último, podemos eliminar un usuario. Igual que en la opción anterior si no seleccionamos a ninguno nos aparecerá la misma advertencia. Al pulsar el botón nos aparecerá un mensaje preguntando si se quiere eliminar el usuario. Si no aceptamos se cerrará y si aceptamos desaparecerá el usuario de la tabla y la base de datos.



Como añadido, también encontramos un menú en el que podremos salir de la aplicación o en la que podremos encontrar información adicional de la aplicación y poder generar un PDF.



Si seleccionamos imprimir y generar PDF, nos saldrá el siguiente mensaje:



Ahora deberemos encontrar un PDF en el escritorio con el nombre Usuarios donde se encuentra la tabla con los usuarios existentes.



Normas de usabilidad aplicadas

Se aplican las mismas normas que en la aplicación principal.

PROBLEMAS ENCONTRADOS Y SOLUCIONES APORTADAS

La mayor dificultad es la falta de experiencia, que hace que se pierda mucho tiempo intentando solucionar errores y consultando foros para poder solucionarlos.

El mayor problema encontrado ha sido controlar y realizar las relaciones que existen entre las tablas mediante código en la base de datos no relacional, ya que, es algo que no había practicado antes.

Las soluciones aportadas a este problema han sido, por un lado, la búsqueda de vídeos y de documentación en la página de Android Developers, y, por otro, alguna explicación que se ha solicitado a los profesores Paco y Jose Manuel sobre este tema.

Otro problema encontrado y el cual no se ha podido solucionar, ha sido intentar utilizar y hacer compatible la base de datos de Firebase con Maven y JavaFX.

FUTURAS MEJORAS

Por falta de tiempo o por que no se ha llegado a conseguir su correcto funcionamiento, hay algunas funciones que no se han podido llegar a implementar, pero que en futuras actualizaciones se podrían añadir:

- Realizar una gestión de salones de forma gráfica para colocar las mesas de estos en un plano.
- Crear una gestión y control del stock de productos.

- Crear perfiles personalizados para los usuarios.
- Hacer el diseño de la aplicación responsive.
- Realizar el diseño de la aplicación secundaria con JavaFX.

BIBLIOGRAFÍA

Crear un instalador con Java y MySQL.

https://www.youtube.com/watch?v=LRR_Gg8LZps (Parte 1)

<https://www.youtube.com/watch?v=Yb8P-bwCC1Y> (Parte2)

Responsive.

[Cómo brindar compatibilidad con diferentes tamaños de pantalla | Desarrolladores de Android | Android Developers](#)

Drag & Drop.

<https://www.youtube.com/watch?v=CPgnZ8N7zeM>

Página genérica para encontrar soluciones a diversos problemas.

[Stack Overflow - Where Developers Learn, Share, & Build Careers](#)