

# PROYECTO INTERMODULAR

## 2º DAM

Autores:

María Del Real

Yeray García

Pablo López

# Contenido

|                                     |    |
|-------------------------------------|----|
| INTRODUCCIÓN.....                   | 3  |
| CASOS DE USO Y FUNCIONALIDAD .....  | 4  |
| LA API Y LOS MODELOS.....           | 5  |
| MANUAL DE USO ANDROID.....          | 8  |
| 1. Estructura.....                  | 8  |
| 2. Inicio de Sesión y Registro..... | 9  |
| 3. Mapa de Mesas.....               | 12 |
| 4. Mesa.....                        | 14 |
| MANUAL DE USO ESCRITORIO .....      | 19 |
| 1. Estructura.....                  | 19 |
| 2. Inicio de Sesión y Registro..... | 20 |
| 3. Mapa Mesas.....                  | 22 |
| 4. Mesa.....                        | 26 |
| 5. Opciones Administrador .....     | 29 |
| 5.1 Abrir/Cerrar Caja .....         | 30 |
| 5.2 Salir.....                      | 30 |
| 5.3 Zonas .....                     | 31 |
| 5.4 Mesas .....                     | 32 |
| 5.5 Empleados.....                  | 35 |
| 5.6 Productos .....                 | 37 |
| 5.7 Totales.....                    | 39 |

## INTRODUCCIÓN

El objetivo final de este proyecto es la creación de una TPV funcional, en nuestra versión hemos decidido hacerla adaptable para cualquier tipo de cliente o empresa que podrá registrarse e iniciar sesión desde varios dispositivos ya sean móviles o de escritorio.

Una vez pasado el inicio de sesión en Android nos pedirá elegir un empleado con el que trabajar mientras que, en escritorio, como no es de uso unipersonal, cada vez que accedamos a una mesa tendremos que introducir la contraseña de nuestro usuario o empleado y éste deberá haber fichado entrada.

Una vez terminado el inicio de sesión todos los dispositivos de un mismo cliente estarán interconectados y podrán acceder a las mismas mesas, cuentas, productos....

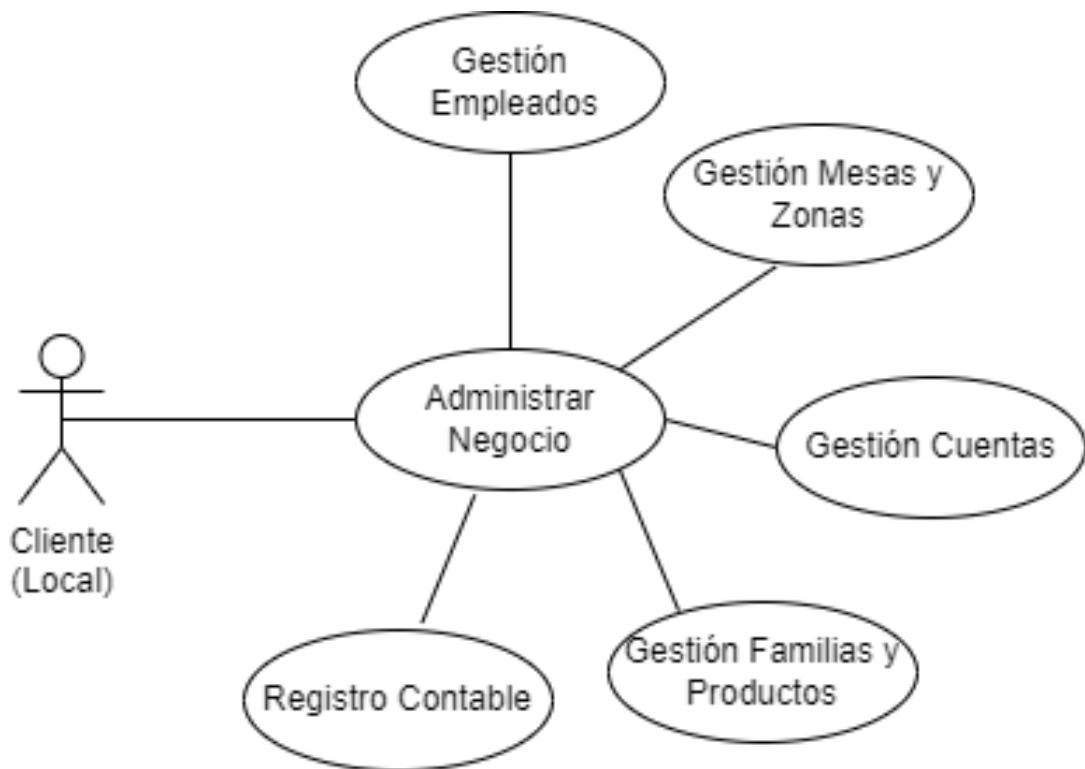
El diseño es totalmente responsive para adaptarse a cualquier dispositivo tanto en Android como en escritorio.

Para el desarrollo hemos utilizado Android Studio y Kotlin para la parte móvil, Visual Studio y C# para la parte de escritorio, Visual Studio Code con JavaScript para la API y MongoDB Atlas para la base de datos.

Procedemos explicando el código conjunto para después continuar con el manual de Android y Escritorio.

## CASOS DE USO Y FUNCIONALIDAD

La idea principal es satisfacer la necesidad de un cliente que quiere controlar su negocio, para ello necesitará poder crear, modificar, eliminar, registrar etc. los diferentes empleados con sus roles, mesas, zonas, los tickets y los cobros, los productos y las familias a las que pertenecen y por último llevar un registro económico de todas las transacciones.



## LA API Y LOS MODELOS

La API a la que accede tanto Android Studio como Visual Studio contiene los métodos para conectarse con MongoDB. Cada Modelo tiene un archivo .js con su mismo nombre especificando sus rutas:

```
APIREST
  models
    caja.js
    client.js
    family.js
    prodImage.js
    product.js
    table.js
    ticket_line.js
    ticket.js
    user.js
    zone.js
  node_modules
  routes
    caja.js
    client.js
    family.js
    prodImage.js
    product.js
    table.js
    ticket_line.js
    ticket.js
    user.js
    zone.js
  www
  .env
  index.js
  package-lock.json
  package.json
  request:http

models > JS table.js < ...
models > JS table.js < ...
routes > JS table.js > router.put("/tables/:id") callback
```

```
const mongoose = require("mongoose");
const tableSchema = mongoose.Schema({
  name: {
    type: String,
    required: true
  },
  status: {
    type: Boolean,
    required: false,
    default: true
  },
  ocupada: {
    type: Boolean,
    required: false,
    default: false
  },
  id_zone: {
    type: String,
    required: true
  },
  comensales: {
    type: Number,
    required: false,
    default: 0
  },
  num_row: {
    type: Number,
    required: false
  },
  num_column: {
    type: Number,
    required: false
  },
  comensalesMax: {
    type: Number,
    required: true
  },
  id_user: {
    type: String,
    required: false,
    default: "Error"
  },
  id_ticket: {
    type: String,
    required: false,
    default: "Error"
  }
})
module.exports = mongoose.model('Table', tableSchema);
```

```
const express = require("express");
const router = express.Router();
const tableSchema = require("../models/table");

//Create new Table
router.post("/tables", (req, res) => {
  const table = tableSchema(req.body);
  table
    .save()
    .then((data) => {
      res.json(data);
      console.log(`\nNew Table: \n ${data}`);
    })
    .catch((err) => {
      res.json({message:err});
      console.error(` Error get /api/table : ${err}`);
    })
});

//Get all Tables
router.get("/tables", (req, res) => {
  tableSchema
    .find()
    .then((data) => {
      res.json(data);
      console.log(`\nAll Tables: \n ${data}`);
    })
    .catch((err) => {
      res.json({message:err});
      console.log(` Error get /api/tables : ${err}`);
    })
});

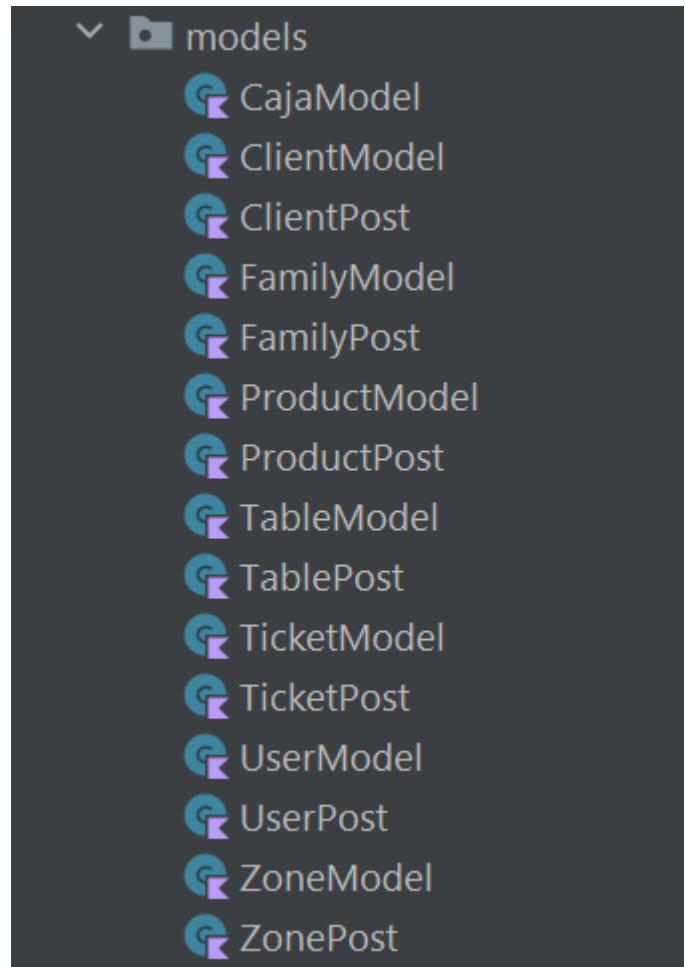
//Get specific table
router.get("/tables/:id", (req, res) => {
  const {id} = req.params;
  tableSchema
    .findById(id)
    .then((data) => {
      res.json(data);
      console.log(`\nTable: \n ${data}`);
    })
    .catch((err) => {
      res.json({message:err});
      console.log(` Error get /api/tables/${id} : ${err}`);
    })
});

//Update table, si algun campo no se pone no se elimina
router.put("/tables/:id", (req, res) => {
```

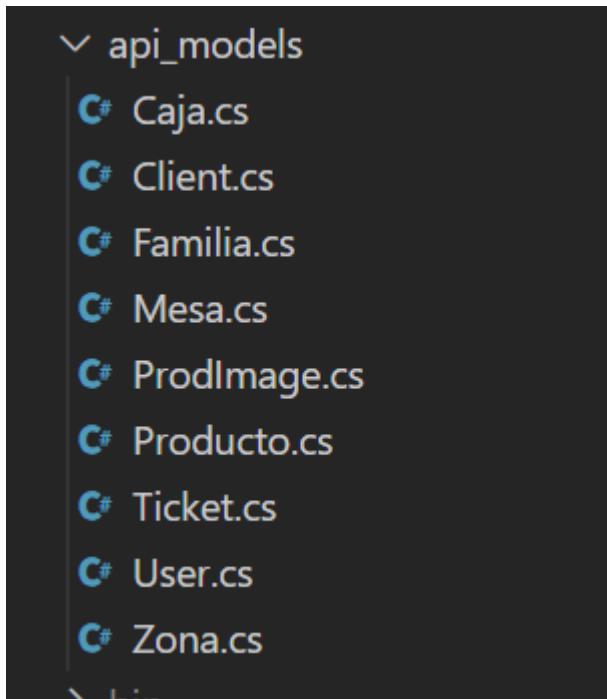
Estos modelos configuran las colecciones en la base de datos y serán los objetos que utilizamos en ambas plataformas. En el caso de Escritorio las clases que instancian estos objetos llevan los métodos asíncronos para llamar a la Api mientras que en Android hemos creado para cada colección un modelo y una versión reducida del mismo, el “Post”. Utilizamos mongoose de Node.js para la conexión con la Base de datos.

Esta versión reducida la hemos necesitado para crear por primera vez un objeto nuevo en MongoDB, como tiene atributos en falta no se mandan a la API y ésta genera automáticamente los valores. Esto no ocurre en el caso de Visual Studio ya que al poder crear varios constructores en una clase normal podemos mandar el objeto por primera vez como queramos.

Android Studio



Visual Studio



En el caso de Android además los métodos de conexión de la Api se encuentran en los ViewModels y, a su vez, los ViewModels instancian la interfaz donde se encuentran las rutas y la instancia de Retrofit necesaria para hacer las peticiones.

## Modelos (todos tienen un id único autogenerado por MongoDB)

- Caja: Permite guardar el total de todos los tickets en un turno con fecha de apertura y salida.
- Cliente: Es el restaurante o empresa donde trabajan el grupo de usuarios, es necesario iniciar sesión con un cliente para realizar cualquier acción. La contraseña viaja cifrada por el método SHA-256
- Familia: Son las agrupaciones de los productos
- Mesa: Donde se guarda la posición, numero de comensales, si tiene algún ticket asignado, si está ocupada o no...
- ProdImage: Es una prueba para poder guardar las imágenes, no implementado.
- Producto: Contiene la cantidad, descripción, precio, familia...
- Ticket\_Line: Es una copia de producto pero que decidimos tener en otra colección aparte donde tenga un id\_ticket y así conformar cada una de las líneas que constituyen el ticket.
- Ticket: Guarda la mesa, el total, el empleado, la fecha y su id está en todas las líneas de ticket que lo conforman.
- User: Es cada uno de los empleados que trabajan para el cliente, tienen nombre y contraseña numérica.
- Zone: Son las agrupaciones de las mesas.

# MANUAL DE USO ANDROID

## 1. Estructura

En la MainActivity se encuentran el NavHost que utilizamos para cambiar entre las diferentes screens.

Las screens necesitan de un ViewModel para hacer las peticiones a la API, por tanto, cada uno de los paquetes de screens corresponde a un ViewModel del paquete view\_models.

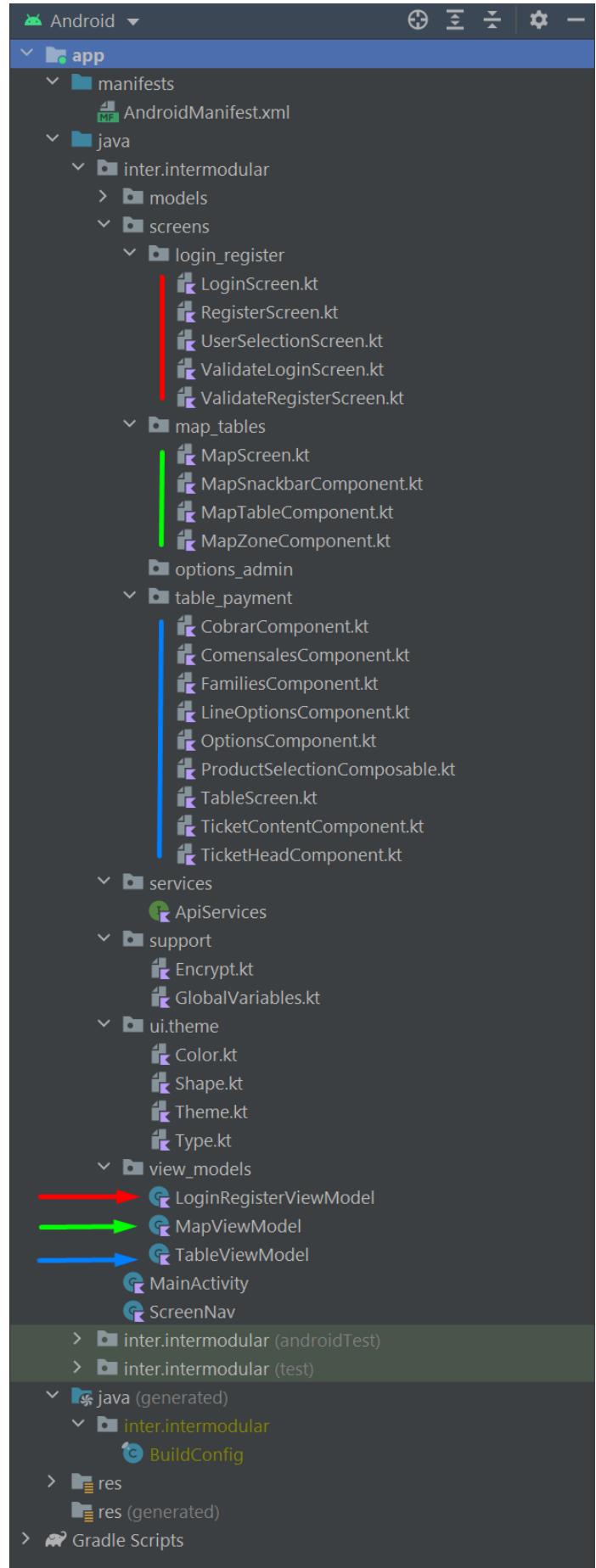
Las rutas a cada una de las screens se encuentran guardadas de manera estática en el ScreenNav y lo llamaremos cada vez que queramos utilizar el navigate del navController.

Por otro lado, tenemos el paquete services que contiene las rutas de la api y el objeto Retrofit necesario para hacer las peticiones.

Por último, tenemos los modelos que ya comentamos en la descripción de la Api y el paquete de soporte que guarda la clase del cifrado SHA-256 necesario para las contraseñas de los clientes y las variables globales que hemos necesitado para el desarrollo.

### IMPORTANTE

Para utilizar la aplicación hay que cambiar la IP en services -> ApiService, hay que especificar la dirección donde corre la API



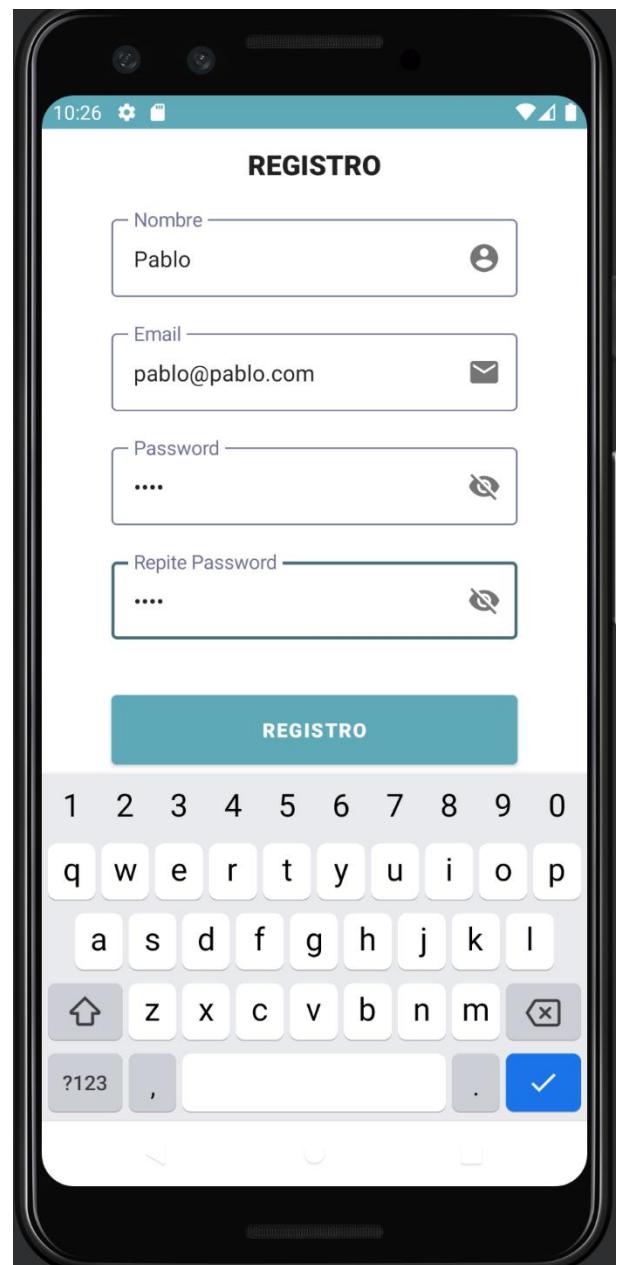
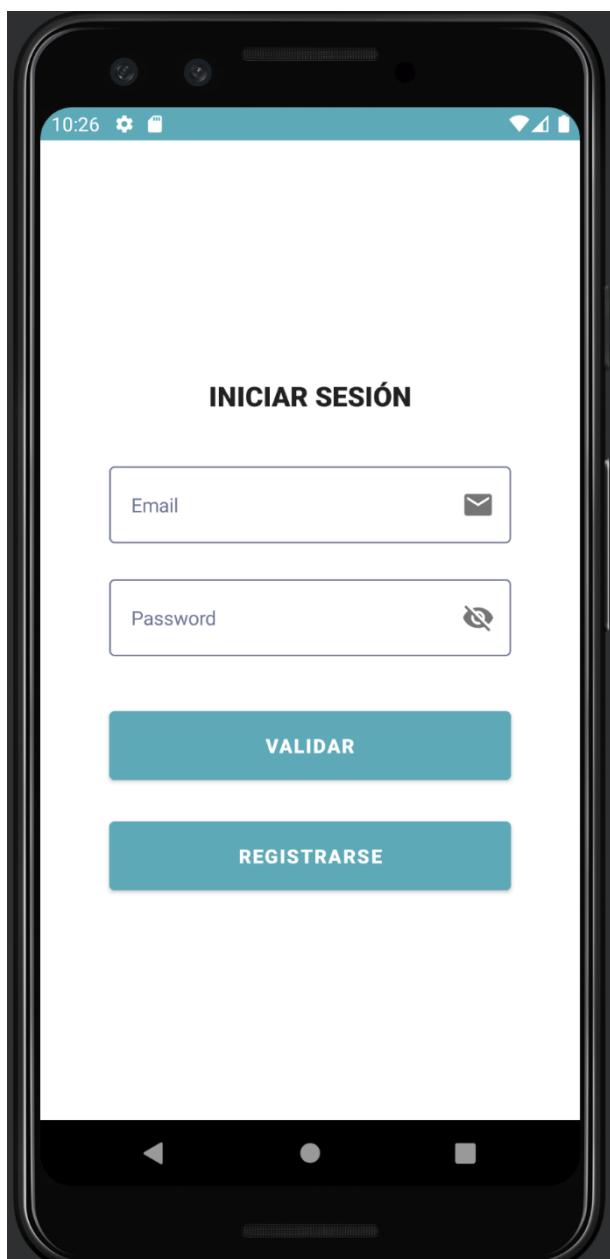
```
fun getInstance(): ApiServices{
    if(apiServices == null){

        /**TODO CADA UNO TIENE QUE PONER SU IP LOCAL DONDE CORRE LA API*/
        /**cmd → ipconfig → IPv4Address**/

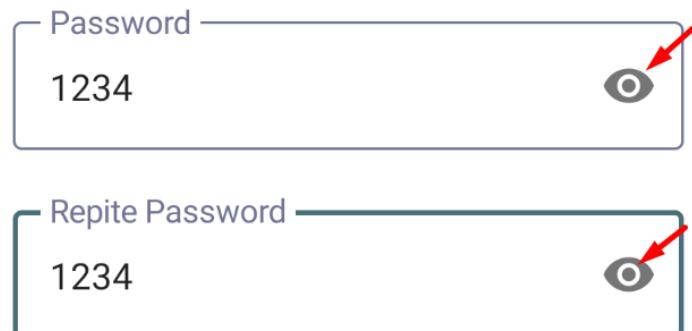
        //val address : Array<String> = arrayOf("http://192.168.56.1:8081/api/", "Pablo")
        val address : Array<String> = arrayOf("http://192.168.1.93:8081/api/", "PabloPhone")
    }
}
```

## 2. Inicio de Sesión y Registro

Para empezar con la aplicación podremos iniciar sesión o registrarnos.

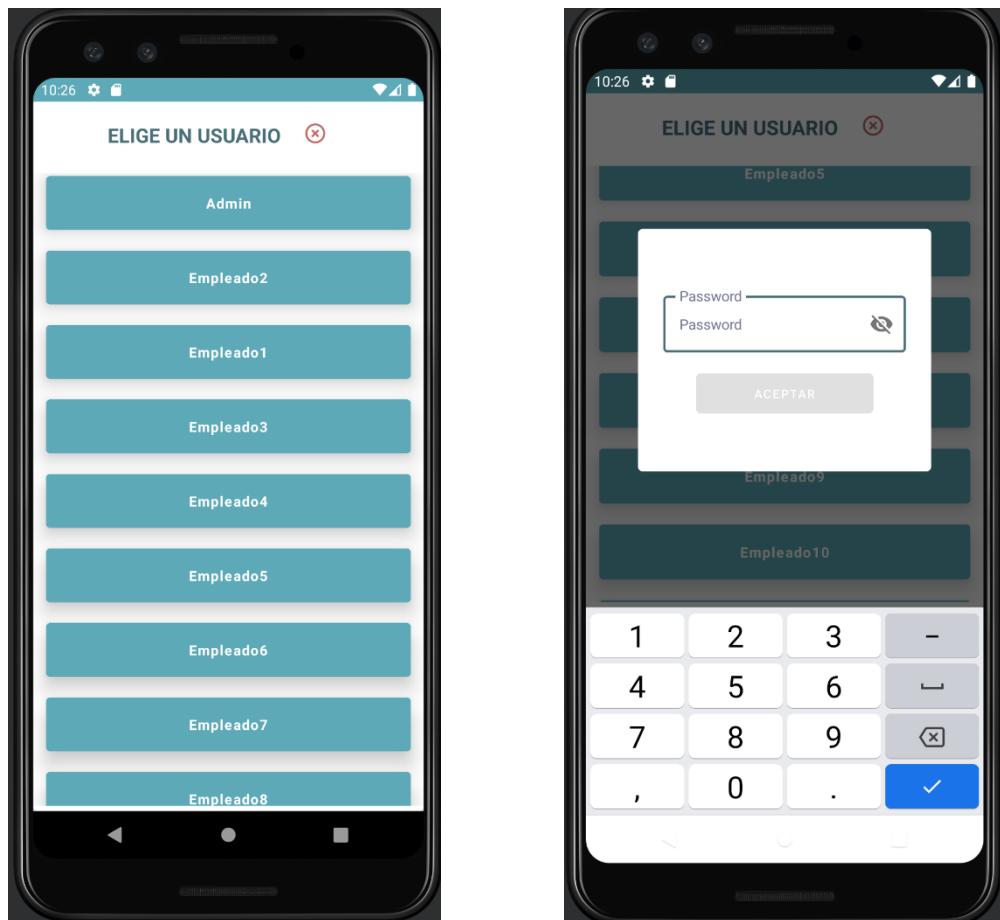


Como se puede apreciar en las imágenes la columna que contiene el formulario se desplaza hacia arriba cuando emerge el teclado en pantalla que, además, en el caso de las contraseñas, desplegará el teclado seguro y contaremos con el ícono de visibilidad para ver o no lo que escribimos.

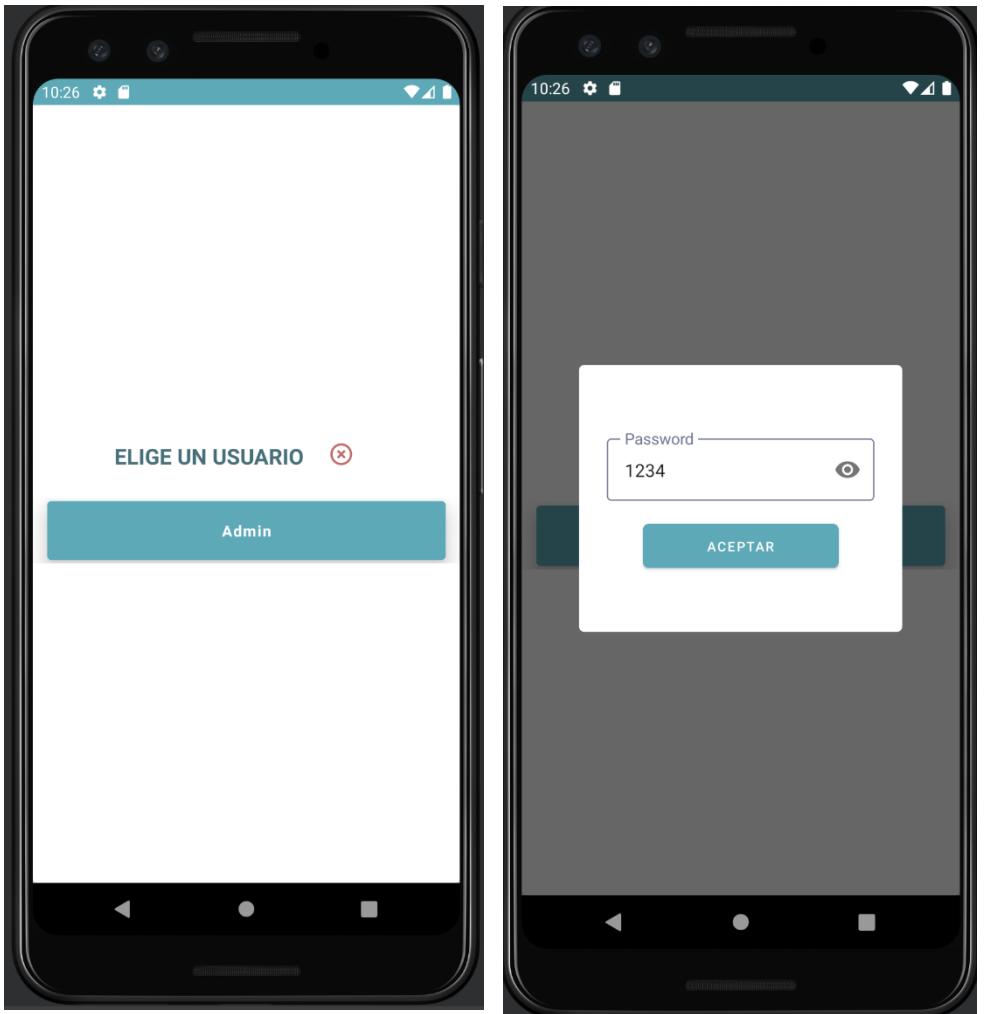


Si registramos un nuevo cliente automáticamente nos creará un nuevo usuario o empleado por defecto con el rol de administrador y accederemos directamente. Además, se creará una Zona Comedor con 20 Mesas, Familia Barril con 2 Productos por defecto para compensar la falta de opciones de administrador en Android.

Si ya tenemos un cliente registrado e iniciamos sesión la aplicación nos pedirá elegir un empleado con el que acceder.



¿Y qué pasa si iniciamos sesión con un cliente recién creado? Como el usuario administrador se crea por defecto mostrará la contraseña directamente mientras el usuario siga siendo Admin con password 1234.



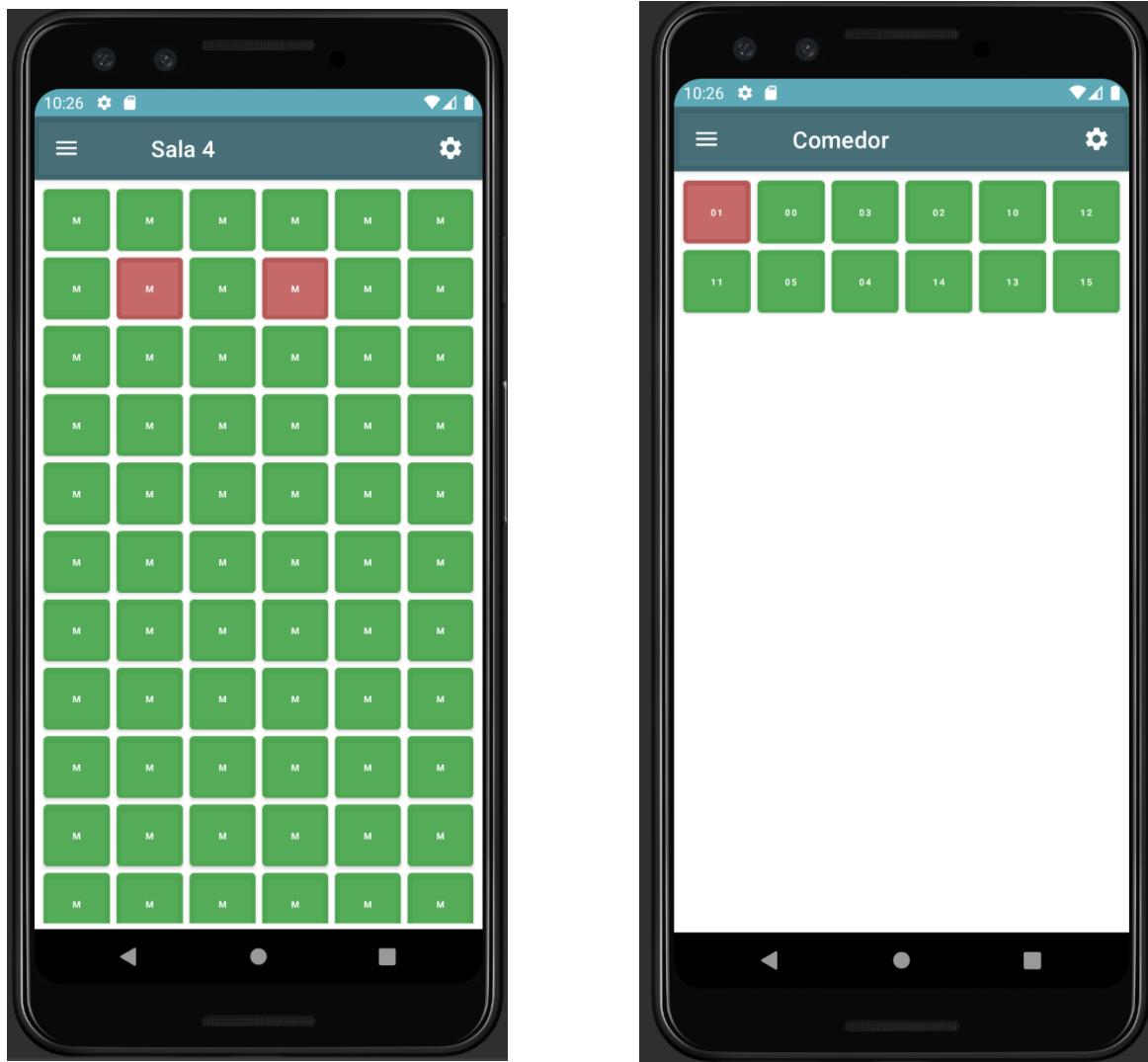
Por otro lado, no podremos registrar un correo que ya exista y, si introducimos 3 veces mal el cliente-contraseña al iniciar sesión la aplicación se cerrará automáticamente.

La elección de usuario tras hacer el login del cliente es la principal diferencia con Escritorio ya que consideramos que un dispositivo móvil solo lo utilizará una persona y no tiene por qué estar introduciendo su contraseña todo el rato, en el caso de escritorio varios empleados estarán manipulando el mismo programa.

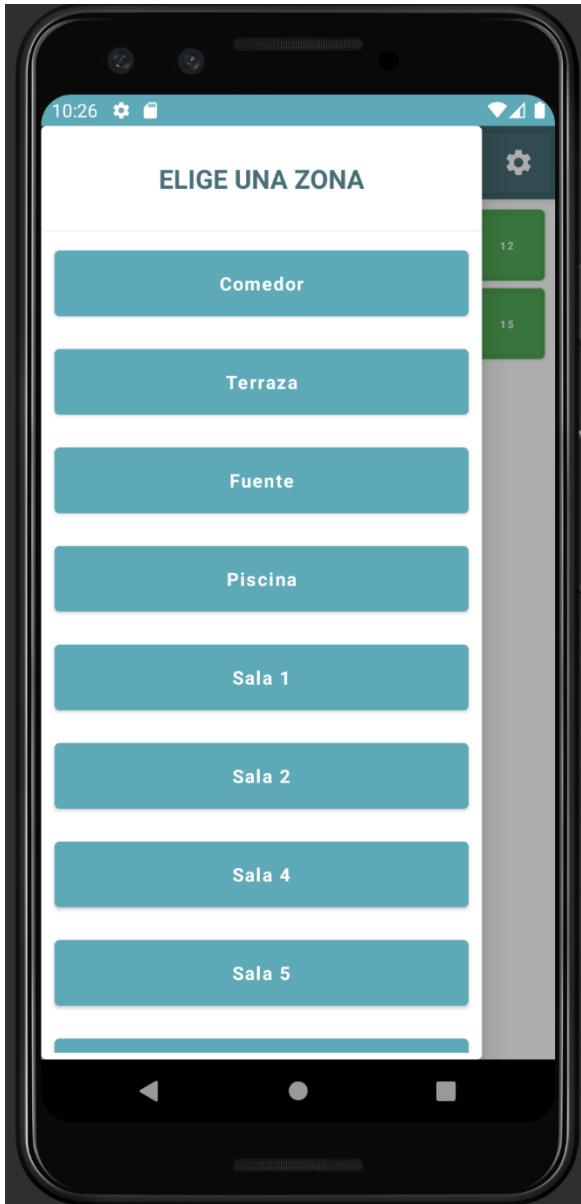
Una vez pasado este bloque donde ya tenemos el cliente y el usuario seleccionado cargamos por primera vez las mesas, zonas, familias, productos y caja si la hubiera abierta.

### 3. Mapa de Mesas

Este bloque consta fundamentalmente de un Scaffold con TopBar, Drawer y Snackbar, luego el contenido es un LazyVerticalGrid que dispone los botones de las mesas en filas de 6 como en la aplicación de escritorio hasta ocupar la pantalla y, cuando hay mas elementos se añade la función de hacer scroll como a todas las listas en la aplicación.

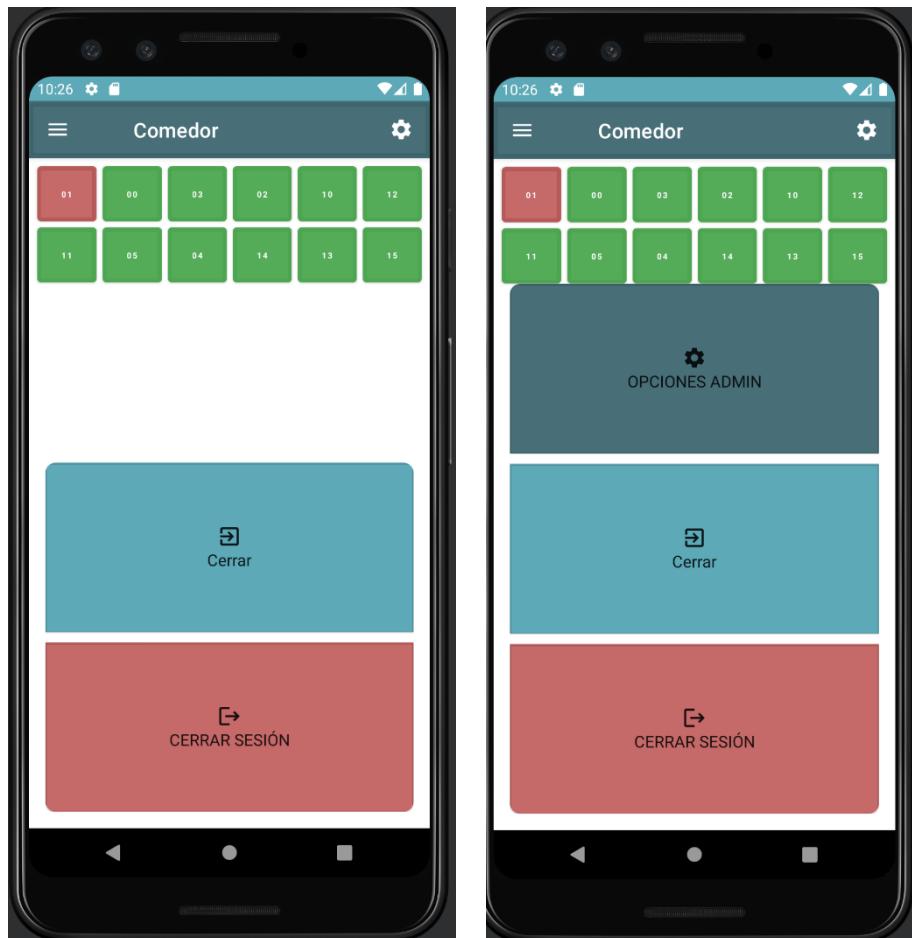


La topbar muestra como titulo la zona actual, a su izquierda se encuentra el icono que despliega las zonas y a la derecha el de las opciones.



El Drawer lo podemos sacar deslizando hacia la derecha o pulsando el icono de arriba a la izquierda en la topbar, nos mostrará las diferentes zonas que tiene configurado el cliente y pulsar en cualquiera de ellas cambiará el contenido mostrando las mesas correspondientes a esa zona.

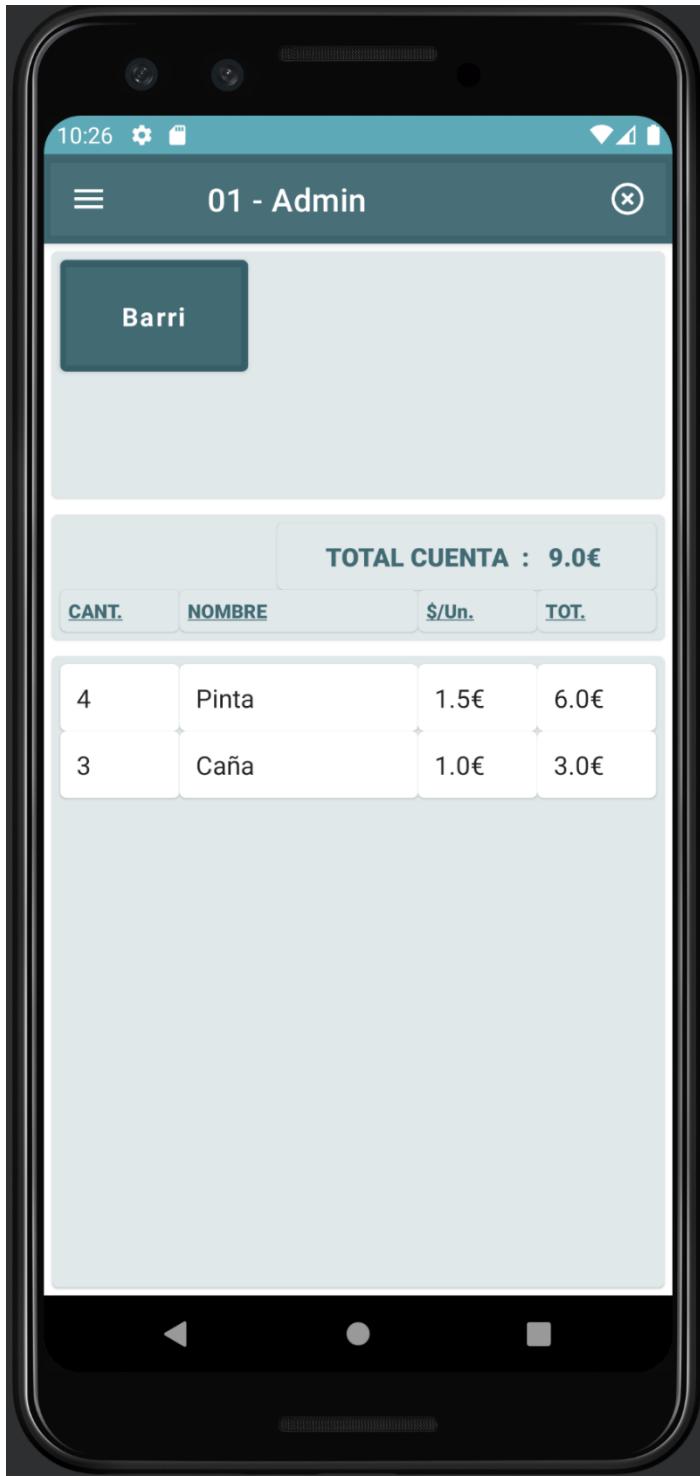
El snackbar nos muestra las opciones durante unos segundos, cerrar sesión nos conducirá de vuelta a la pantalla de inicio de sesión, cerrar cierra las opciones y dependiendo de si el usuario tiene o no permisos de administrador se mostrará el botón de Opciones Administrador cuya funcionalidad no hemos tenido tiempo de desarrollar.



Por último, si pulsamos en cualquier mesa accederemos al último bloque donde se cargarán las líneas de ticket o cuenta si las hay.

#### 4. Mesa

Es el bloque mas complicado, tenemos un Scaffold como en el anterior pero en esta ocasión hay numerosos cuadros de diálogo como componentes que aparecen y desaparecen según nuestra interacción.

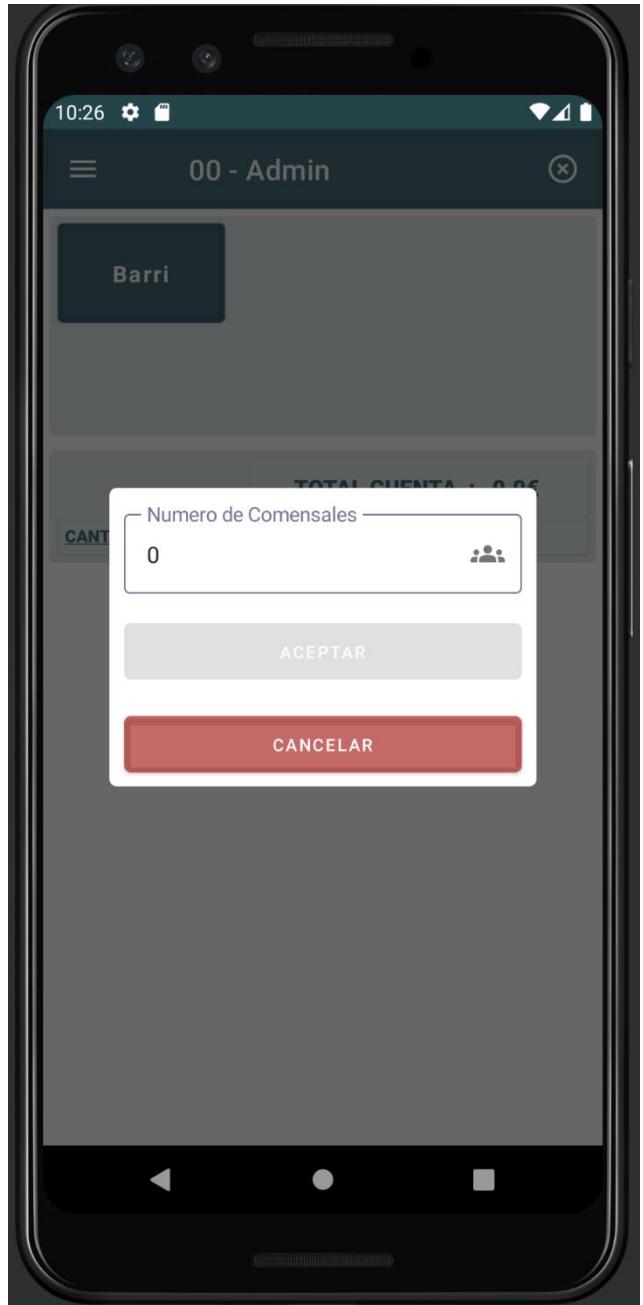


La top bar nos indica el empleado que está accediendo a la mesa y el nombre de ésta. La "X" de la derecha nos devuelve al mapa y borrará el ticket si no hay ningún producto.

Si deslizamos a la derecha o pulsamos el ícono a la izquierda de la topbar desplegaremos el drawer que por el momento solo tiene las opciones de cobrar o modificar comensales.

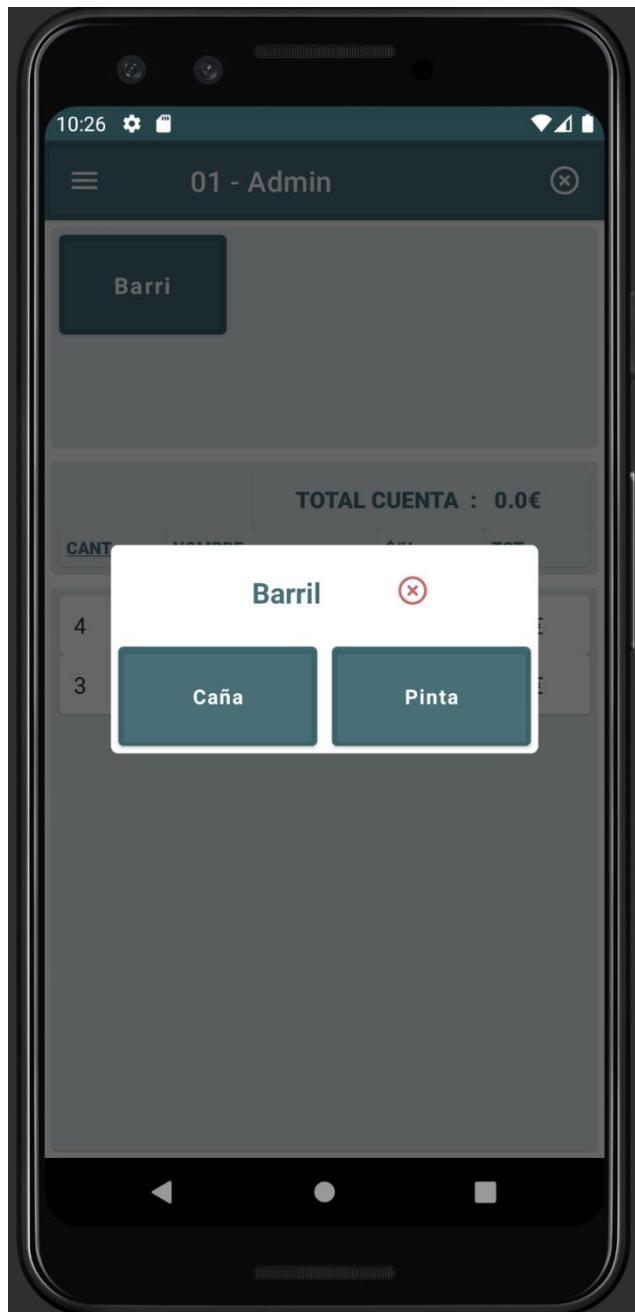


Si pulsamos en Mod. Comensales o es la mesa en la que entramos está libre saltará el diálogo de comensales, se nos informará con un toast del numero máximo de comensales que podemos introducir y también si introducimos algún carácter inválido, el botón de aceptar no estará activo hasta que se cumplan estos requisitos, pero podemos darle a cancelar y actualizar el numero de comensales mas adelante.



Si pulsamos en cobrar se desplegará el diálogo de cobro donde nos indica la mesa y el total. La opción de pagar en efectivo aparece seleccionada por defecto y para poder cobrar deberemos introducir una cantidad superior al total o pulsar en el botón de calcular que iguala la cantidad del introducido a la del total. Si seleccionamos tarjeta podremos cobrar directamente.

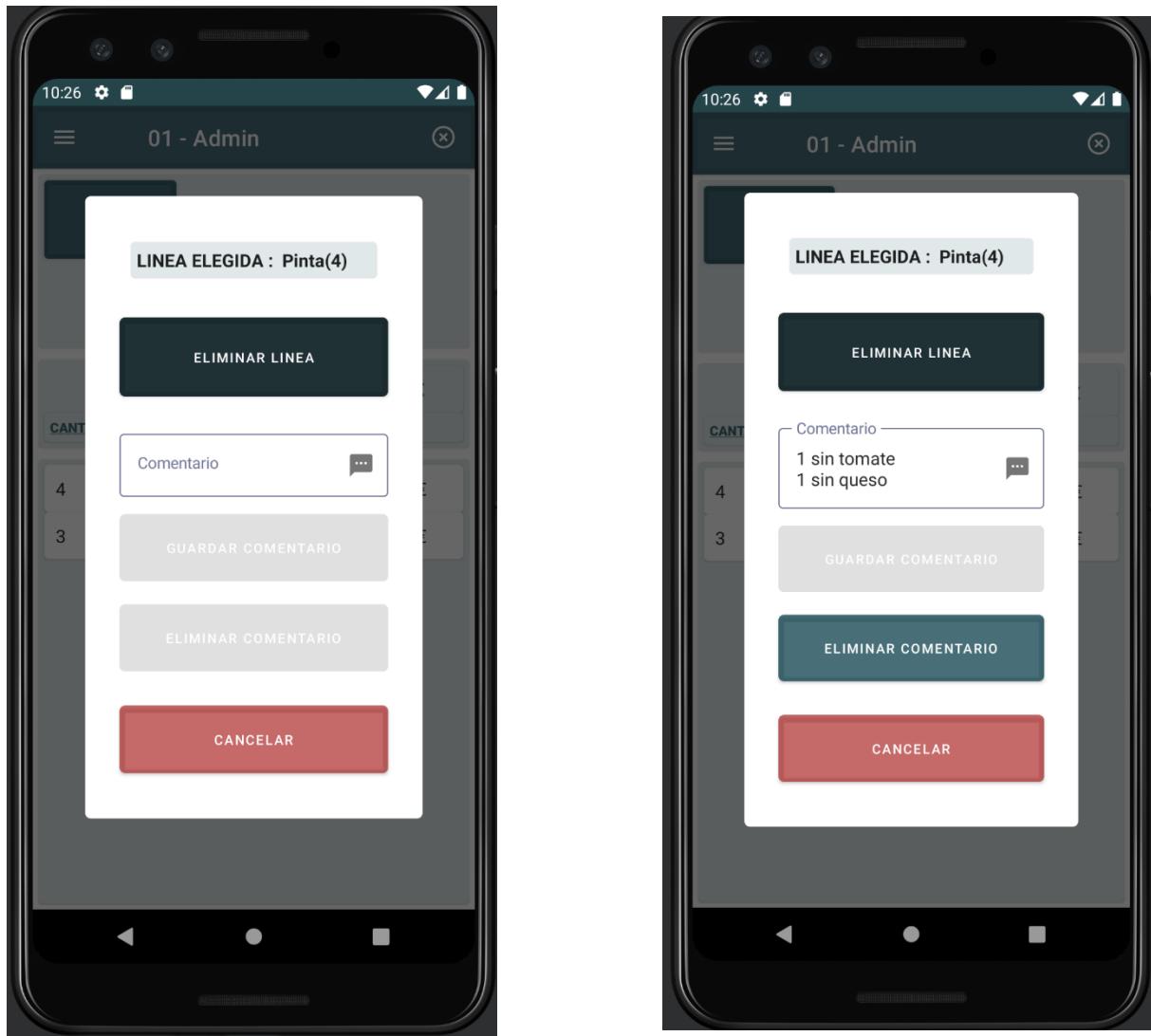
El contenido del Scaffold a su vez está dividido en tres partes, el primero lo constituyen las familias de productos que siempre ocuparán el mismo porcentaje de espacio y si hay más de seis se podrá hacer scroll. A continuación, se encuentra un pequeño componente que nos indica el total de la cuenta y titula las columnas del siguiente componente que es donde se encuentran las diferentes líneas de la cuenta.

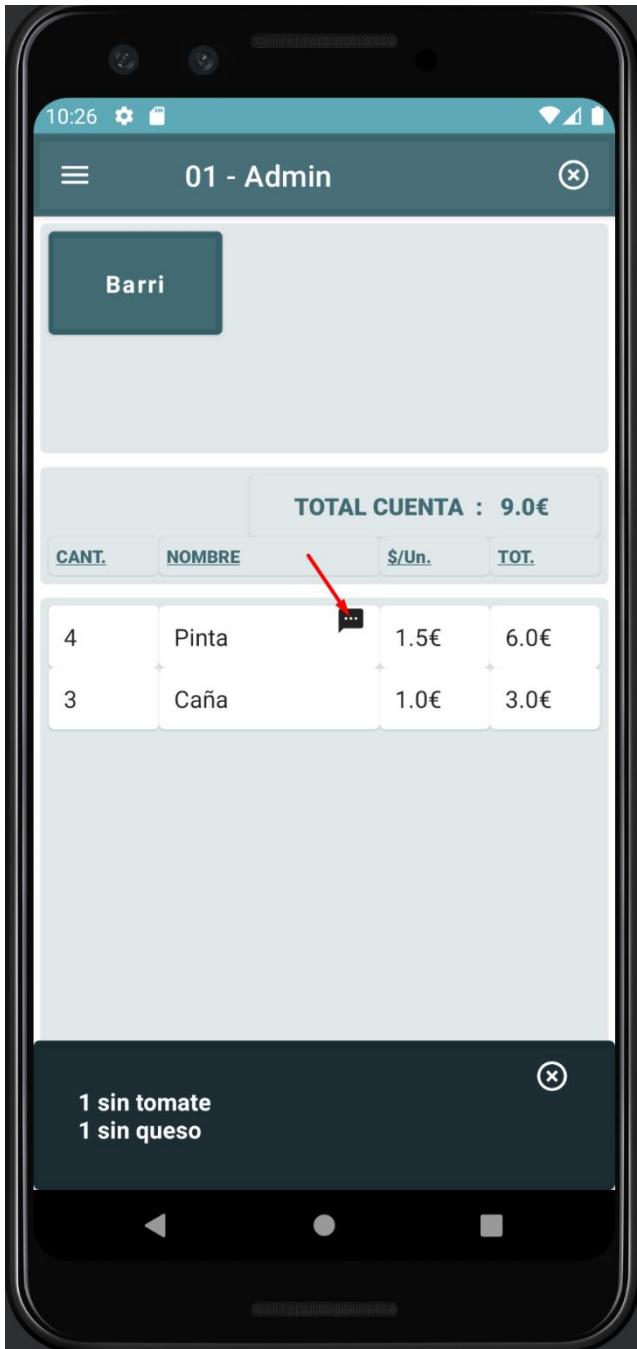


Cuando pulsamos en una familia aparece un dialogo con botones donde se encuentran los productos de la familia y podemos pulsarlos para añadirlos a la cuenta tantas veces como queramos hasta que cerremos el dialogo pulsando la x. El cuadro de diálogo se adapta en tamaño a la cantidad de productos que contenga.

Por último, podemos pulsar en cada una de las líneas que forman la cuenta y nos saltará un toast indicando que podemos acceder a opciones con un LongClick

De esta forma podremos eliminar toda la línea o añadir un comentario, el botón de aceptar no estará activo hasta que añadamos algo de texto y el de eliminar no estará activo hasta que volvamos a acceder a una línea que ya tiene comentario (no nos dejará guardar hasta que no modifiquemos algo el actual)





Si guardamos un comentario aparecerá un ícono en la línea correspondiente.



Y si lo pulsamos aparecerá un Snackbar rápido que lo muestra.

Si cobramos la mesa o si salimos de ella sin ningún producto la mesa volverá actualizarse como libre y su color cambiará a verde, por otro lado, si salimos de una mesa que aún tiene cuenta está se mantendrá de color rojo y no nos pedirá el número de comensales cuando volvamos a acceder a ella además de recargar todas las líneas de la cuenta desde la base de datos.

# MANUAL DE USO ESCRITORIO

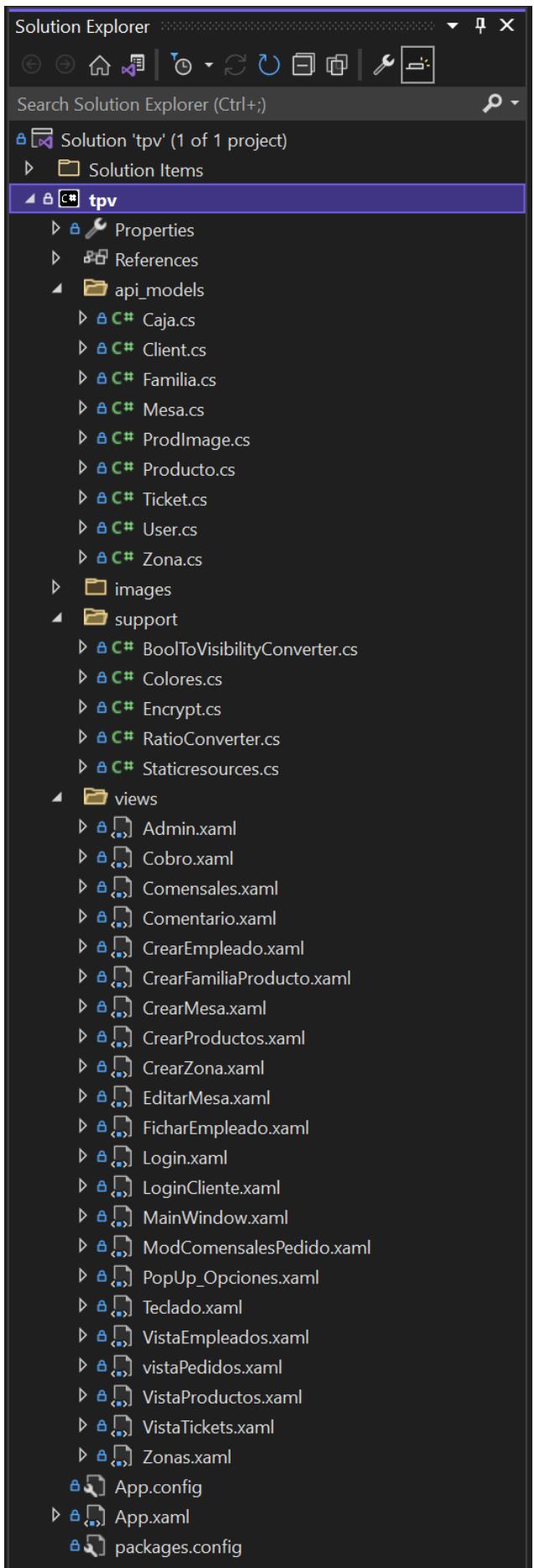
## 1. Estructura

Primero tenemos las clases de api\_models que describen los objetos que se mandarán y recibirán desde la API además de contener los métodos para realizarlo mediante el HttpClient.

Luego tenemos el fichero images que contiene los archivos para ilustrar iconos, botones etc.

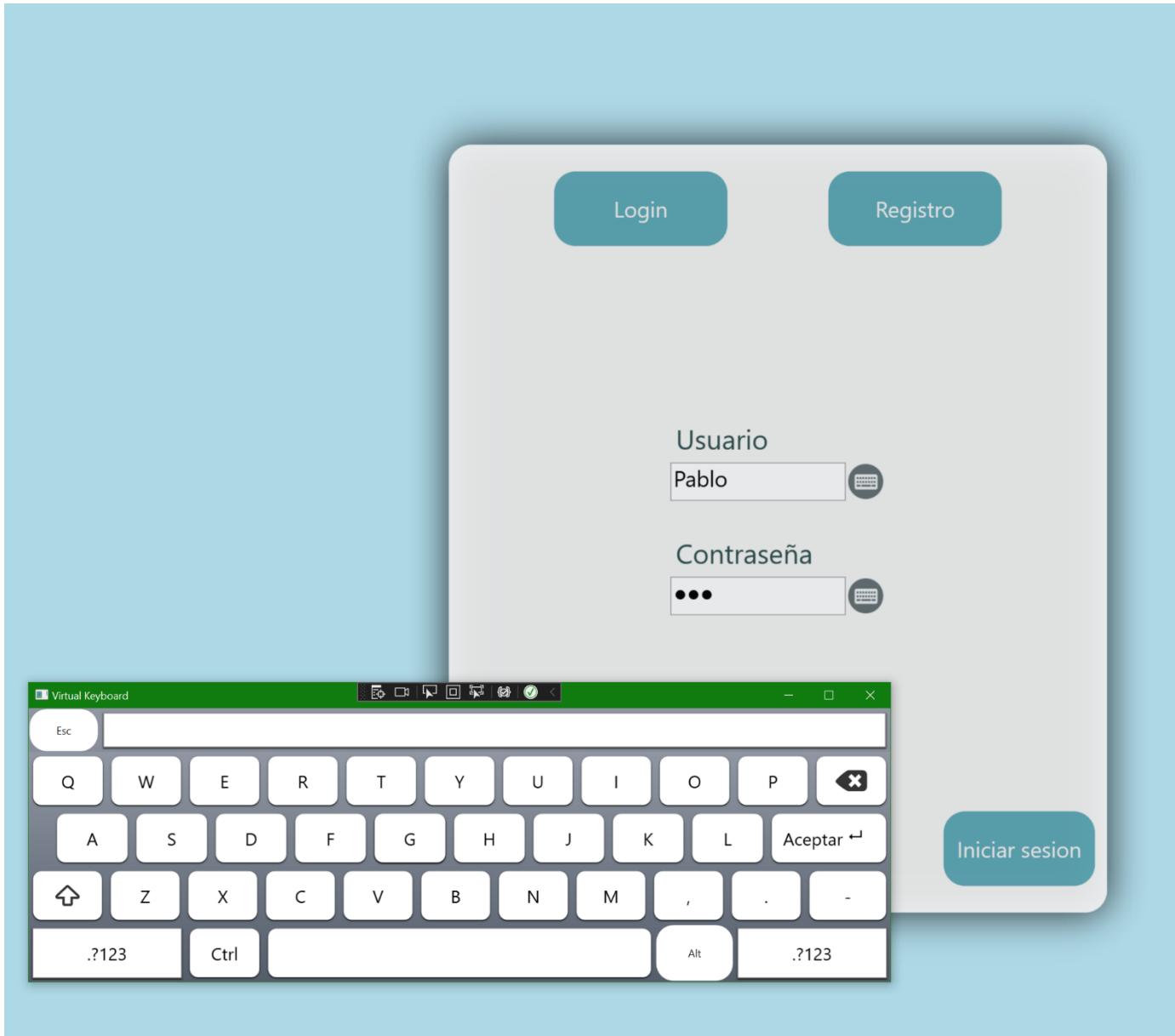
Dentro del archivo support tenemos Encrypt utilizado para el cifrado SHA-256 de las contraseñas de clientes y las variables estáticas dentro de Staticresources, también hemos almacenado en un archivo aparte los colores y dos clases adicionales para el funcionamiento del teclado en pantalla y el calculo de la pantalla.

Por ultimo tenemos el fichero mas importante que es el de las views, aquí se encuentran todas las vistas con su respectivo .cs de código C# asociado describiendo la funcionalidad.



## 2. Inicio de Sesión y Registro

Como principal diferencia con Android el inicio de sesión o registro conducirán directamente a la actividad mapa sin tener que elegir un usuario o empleado antes. En todos los huecos donde hay que introducir información aparece el icono de teclado, si lo pulsamos aparecerá un teclado en pantalla que nos dejará escribir en el control seleccionado, está pensado para el uso de pantallas táctiles.



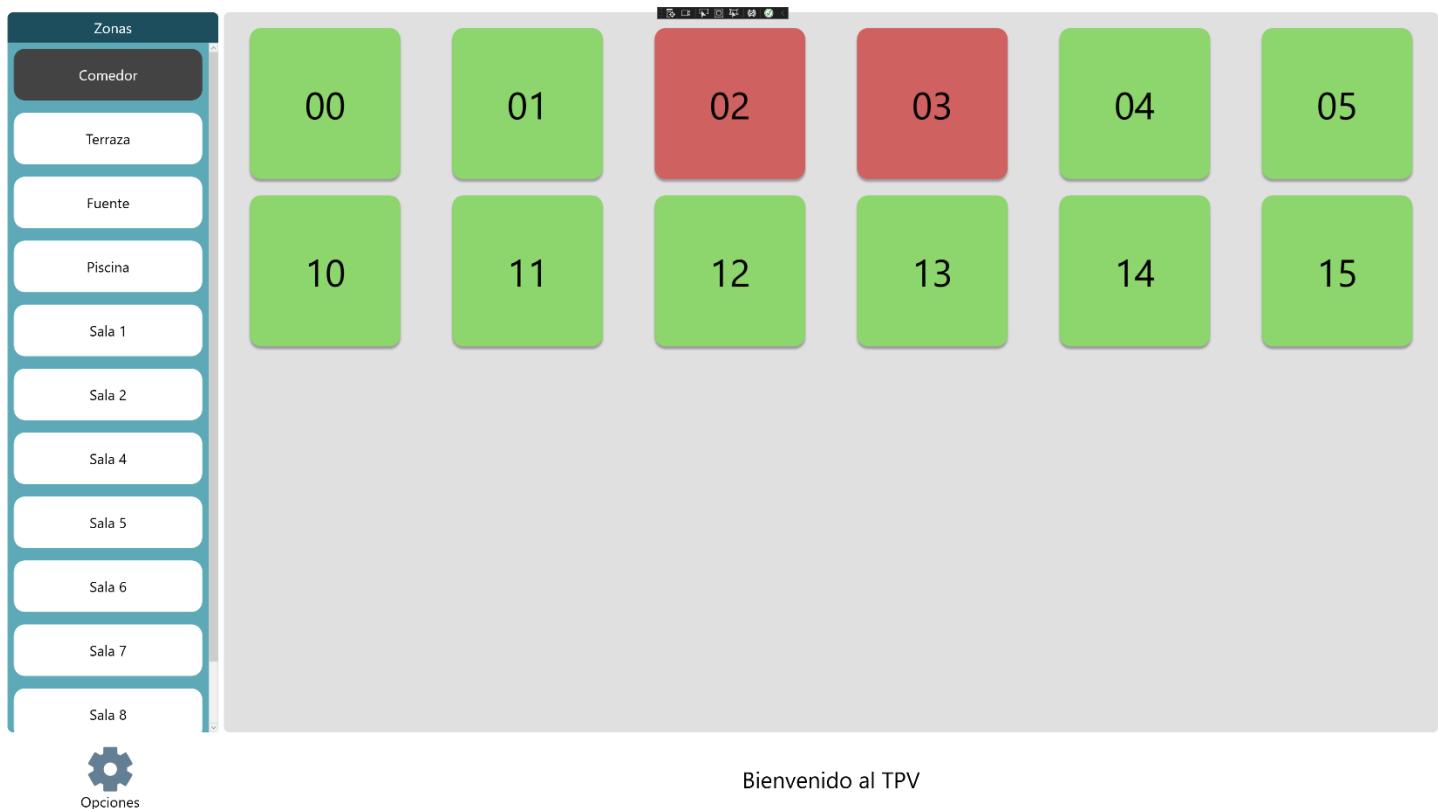
Si creamos un nuevo cliente veremos una notificación indicando que se ha creado un nuevo usuario o empleado llamado Admin cuya contraseña es 1234. La contraseña del cliente se cifrará antes de guardarse en la base de datos.



### 3. Mapa Mesas

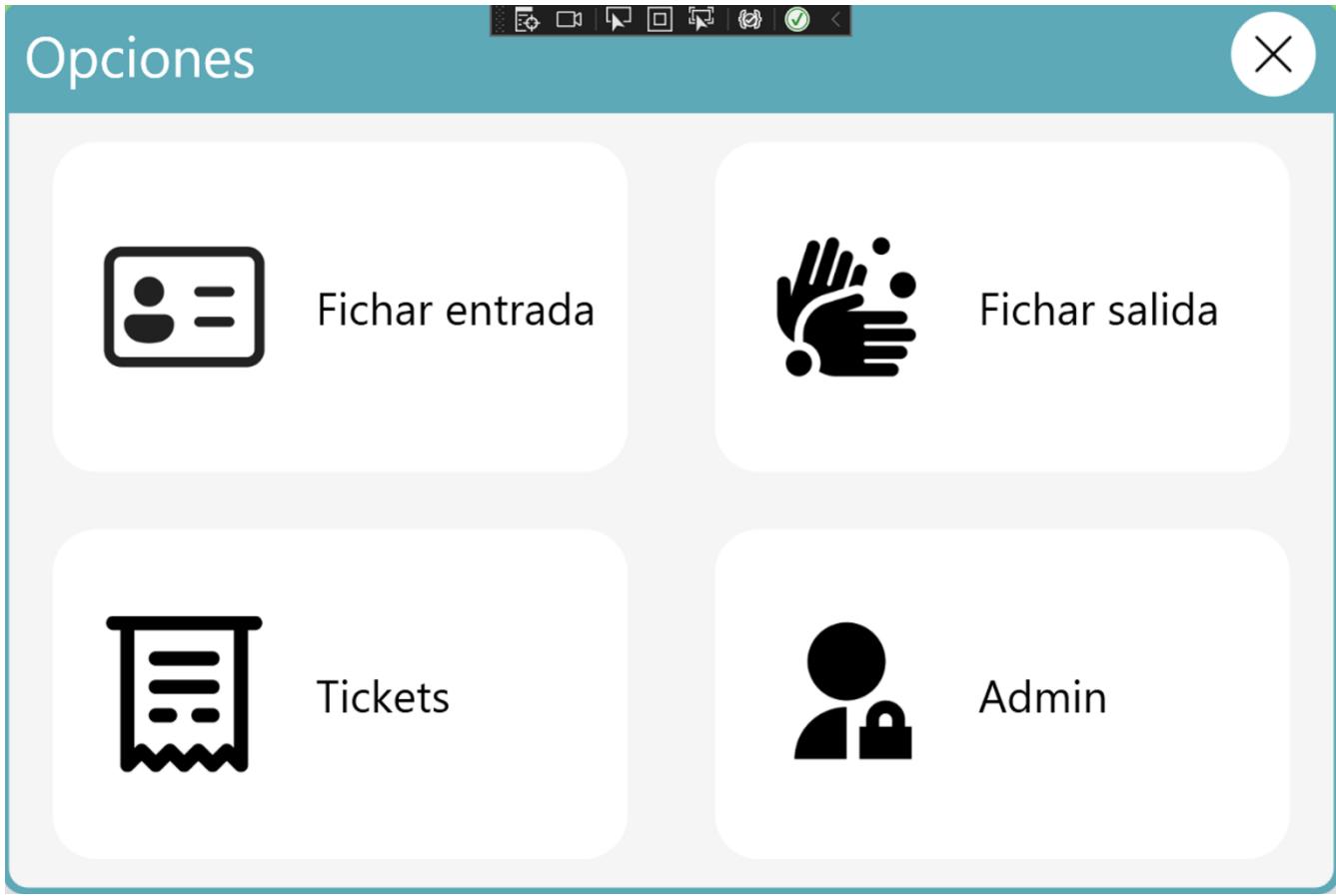
Si acabas de registrarte observarás que no hay ninguna zona ni mesa creadas, esto es porque en escritorio se puede modificar todo fácilmente. Para empezar deberás ir a Opciones -> Admin -> Zonas para crear una Zona y después Opciones -> Admin -> Mesas para crear las Mesas. Una vez terminado e intentas acceder a una mesa te saltará un mensaje de que no hay ninguna caja abierta y si la abres en Opciones -> Admin -> Abrir Caja encontrarás otro problema y es que ningún usuario a fichado entrada por lo que habrá que ir otra vez a opciones -> Fichar Entrada. Por último si consigues entrar a la mesa verás que no hay ningún producto que puedas añadir a la cuenta, para ello deberás ir a Opciones -> Admin -> Familias para crear una nueva familia de productos y finalmente Opciones -> Admin -> Productos para crear los productos que se añadirán a la cuenta.

A continuación iremos explicando cada de las funciones en detalle



La MainWindow o Mapa de Mesas contiene a la izquierda una columna del grid donde están todas las zonas disponibles y a su derecha las mesas correspondientes a cada zona, si la cantidad de mesas o zonas fuera muy grande se genera un horizontal scroll.

Abajo a la izquierda tenemos el botón de opciones que nos hace saltar la siguiente vista:

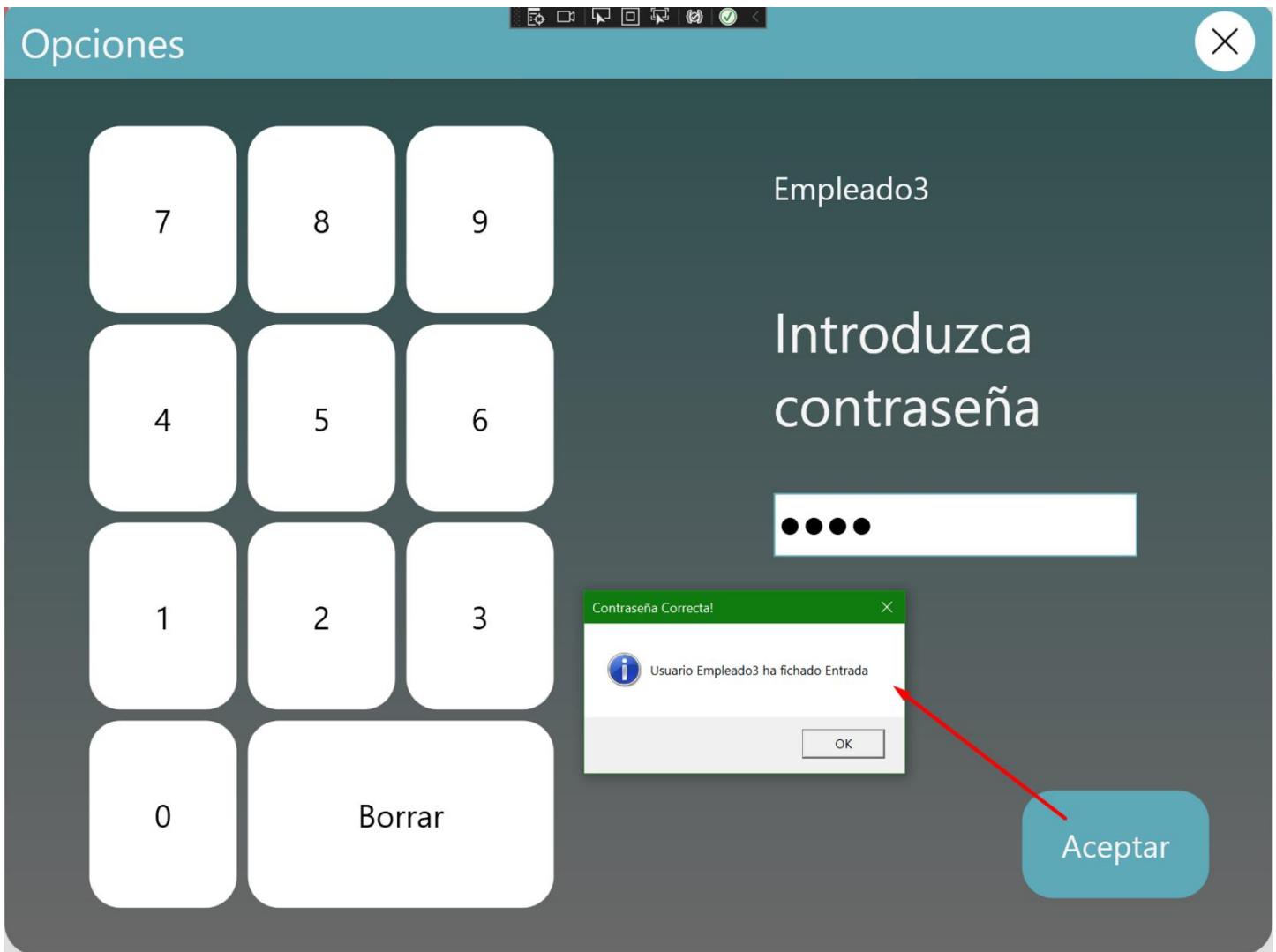


Fichar entrada nos hará aparecer un desplegable con todos los usuarios/empleados del cliente que no hayan fichado ya, aquí podremos seleccionar uno de estos usuarios e introducir la contraseña para que fiche y, desde ese momento, cuando entremos en una mesa y nos vuelva a salir el menú desplegable nos aparecerán todos los usuarios que hayan fichado entrada. Fichar salida es el proceso inverso, aparecerán todos los usuarios que han fichado entrada e introducir correctamente la contraseña hará que fichen salida.

A screenshot of a dropdown menu titled "Empleados". The menu lists ten entries: Empleado1, Empleado3 (which is highlighted with a dark blue background), Empleado4, Empleado5, Empleado6, Empleado7, Empleado8, Empleado9, and Empleado10. The menu has a standard OS X style title bar at the top.

| Empleados  |
|------------|
| Empleado1  |
| Empleado3  |
| Empleado4  |
| Empleado5  |
| Empleado6  |
| Empleado7  |
| Empleado8  |
| Empleado9  |
| Empleado10 |

Cuando pulsemos en Admin también aparecerá esta ventana con la botonera para introducir la contraseña, pero no tendremos que elegir un administrador, cualquier contraseña de administrador para ese cliente será válida para acceder al menú de opciones de administrador.



En resumen esta vista de elegir empleado e introducir contraseña la hemos reaprovechado para fichar entrada, salida y elegir empleado al entrar dentro de una mesa, la vista de la contraseña aparte también para el acceso a las opciones de administrador.

Por último tenemos la opción Tickets que nos mostrará todos los tickets abiertos en cualquiera de las zonas, si pulsamos sobre cualquiera de los tickets además podremos ver una breve descripción de la cuenta.

## Tickets abiertos

Mesa: M Comensales: 1

Mesa: M Comensales: 4

Mesa: 01 Total: 13.5€ X

8 Pinta 12€  
3 Caña 1.5€

OK

Mesa: 01 Comensales: 4

Mesa: 00 Comensales: 6

Mesa: 02 Comensales: 2

#### 4. Mesa

Aquí enlazamos varios grids para estructurar la vista;

The screenshot shows a user interface for a restaurant POS system. On the left, there is a sidebar with a teal header labeled "Categorías" containing buttons for "Barril", "Carne", "Pescado", "Zumos", "Menu", and "Snacks". Below this sidebar is a row of four buttons: "Cobrar" (with a dollar sign icon), "Comensales" (with a fork and knife icon), "Borrar pedido" (with a trash bin icon), and "Atrás" (with a back arrow icon). The main content area has a header "Camarero: Admin" and a date "22/02/2022". It displays "Comensales: 4" and two large white boxes labeled "Caña" and "Pinta". At the bottom, there is a table titled "Cuenta" with columns "Cantidad", "Producto", "€/U", and "Total = 16€". The table contains two rows: one for "Pinta" (quantity 8, price 1.5, total 12) and one for "Caña" (quantity 4, price 1, total 4).

| Cantidad | Producto | €/U | Total = 16€ |
|----------|----------|-----|-------------|
| 8        | Pinta    | 1.5 | 12          |
| 4        | Caña     | 1   | 4           |

Arriba a la izquierda tenemos las diferentes familias de productos, pulsar en cada una de ellas cargará a la derecha (donde pone Caña y Pinta) los diferentes productos que componen la familia.

Si pulsamos en alguno de los botones de productos se añadirá una línea a la cuenta o se modificará la cantidad si se trata de algún producto repetido. En la cabecera de la cuenta tenemos los títulos de cada una de las columnas y el total que se va calculando de forma dinámica.



Si pulsamos en el ícono del comentario nos saltará un cuadro de diálogo para introducir un nuevo comentario, si lo guardamos y volvemos a pulsar veremos el comentario.

Por último tenemos las opciones de abajo a la izquierda donde podremos volver atrás al mapa, borrar todo el pedido, cobrar y editar los comensales.

Volver atrás guardará el estado y el pedido de la mesa, si está vacía volverá a ponerla como libre y borrar pedido elimina todas las líneas de ticket.

Si pulsamos en una de las líneas de la cuenta aparecerá una nueva opción, la de eliminar línea que borrará únicamente ese producto con toda su cantidad.



Cobrar



Comensales



Borrar pedido



Atrás

| Cantidad | Producto |
|----------|----------|
| 4        | Caña     |

Cobrar  
Comensales  
Eliminar línea

La ventana emergente que aparece cuando pulsamos en Comensales es la misma que aparece si estamos entrando en una mesa libre, nos pedirá los comensales antes de empezar a crear la cuenta.

### Editar comensales

Número comensales

(X) Cancelar Siguiente >

Los comensales se verifican según el número máximo que haya especificado en la mesa, se puede modificar desde las opciones de administrador.

Por último el cobro nos permitirá elegir entre efectivo y tarjeta, actualizará la mesa a libre y nos devolverá al mapa.

### Cobro

Empleado Camarero: Admin  
Mesa 00

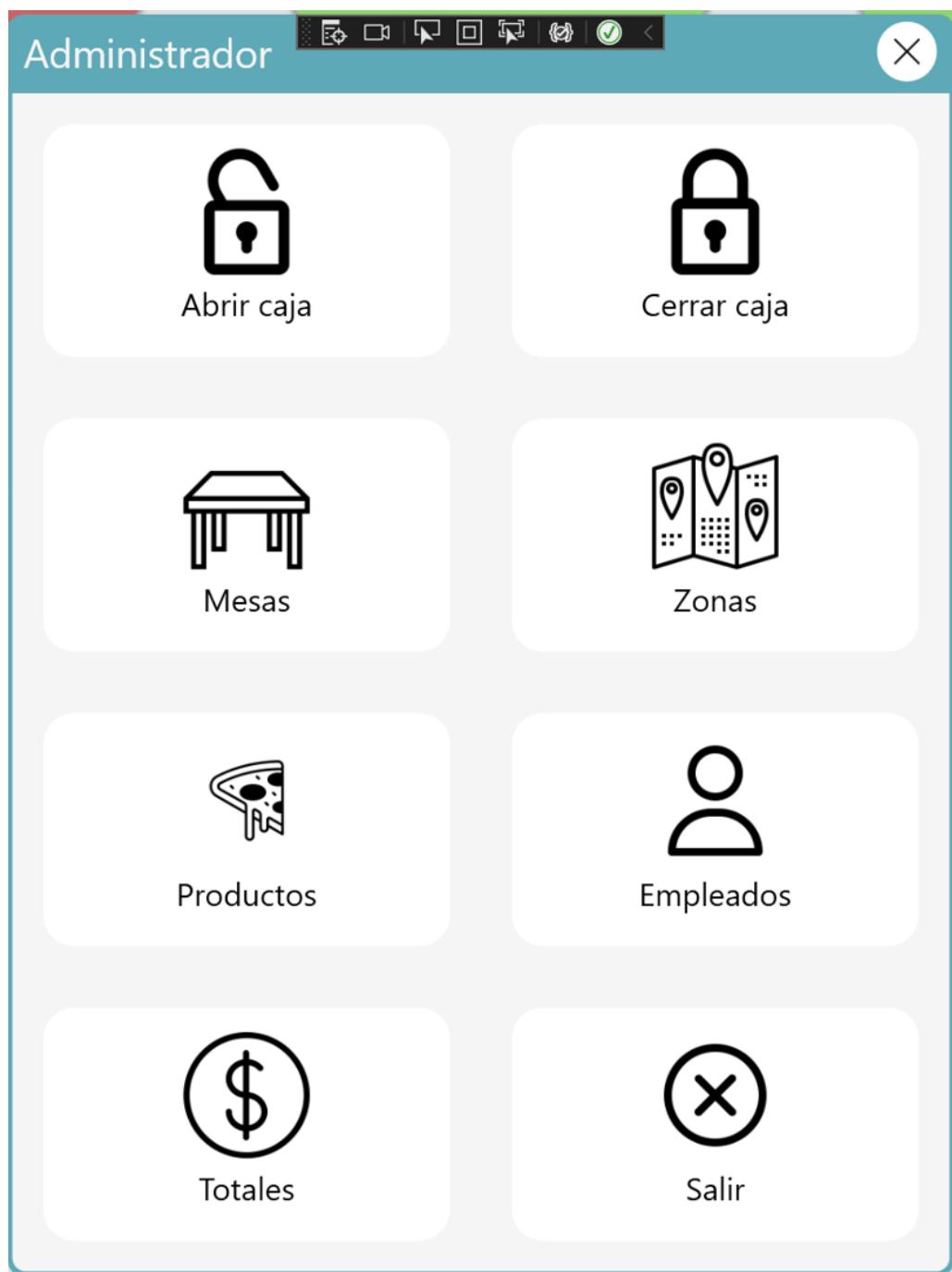
Efectivo Tarjeta

Total Total 20€

## 5. Opciones Administrador

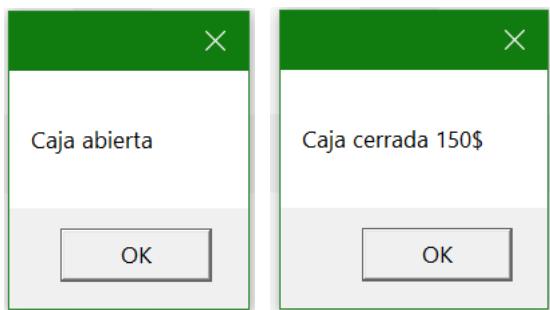
La mayor diferencia con Android ya que no hemos tenido tiempo de implementar estas funcionalidades allí, sin embargo al estar interconectadas el mismo cliente puede acceder a las opciones desde un ordenador aunque luego utilice los dispositivos móviles para trabajar, mencionaremos como hemos ido solventando estos déficits en Android a lo largo de este apartado.

Desde esta vista a la que se accede desde las opciones del mapa, tras introducir una contraseña válida de administrador, podremos modificar todos los aspectos de la aplicación, iremos explicando cada opción por separado

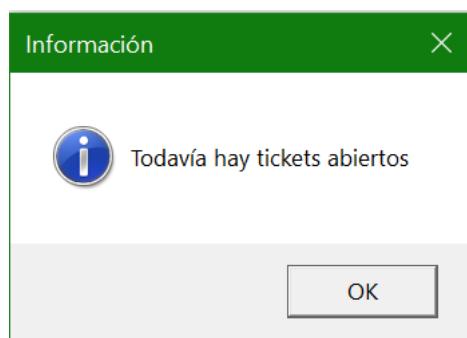


## 5.1 Abrir/Cerrar Caja

Para poder acceder a las mesas es necesario abrir caja, esto no ocurre en Android ya que al no tener la opción de abrir caja no podría ser testeada de manera individual, por tanto, si un cliente tiene una caja abierta el dispositivo Android utilizará esa caja para ingresar sus tickets, si no hay ninguna caja abierta utilizará por defecto una que siempre lo está, para que pueda funcionar independiente.

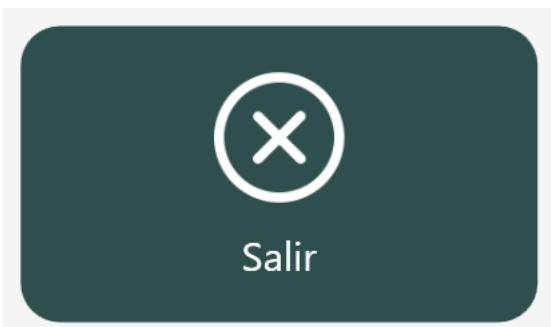


Solo una de estas dos opciones pueden estar disponibles para clicar ya que un cliente solo puede tener una caja abierta al mismo tiempo.



Cerrar la caja nos dará una breve información del turno donde podemos ver el total facturado, no podremos cerrar la caja si queda algún ticket abierto o mesa ocupada.

## 5.2 Salir



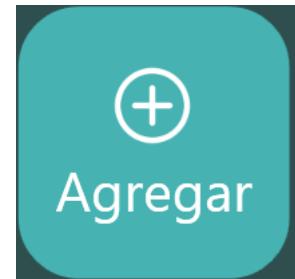
Esta opción termina con el programa, solo se puede acceder desde Administrador porque suponemos que los empleados no deberían poder salir y navegar por el ordenador, sino que con una pantalla táctil ésta interfaz es a lo único que podrán acceder.



### 5.3 Zonas

Aquí podremos modificar, eliminar y crear nuevas zonas.

Para crear pulsaremos el botón de agregar y a continuación en la ventana emergente introduciremos un nombre válido para la nueva zona.



Para modificar o eliminar una zona primero la seleccionaremos de la lista, nos saldrá entonces la opción de eliminar y se cargarán los datos a la derecha, podremos editarlos y si el nuevo nombre es válido podremos guardar los cambios.

#### 5.4 Mesas



#### Mesas

Esta es la opción editable más compleja, volveremos a la actividad mapa pero llamada ésta vez a través de un parámetro que nos permite crear nuevas mesas, modificarlas y eliminarlas.

En primer lugar si no tenemos ninguna zona creada nos saldrá una advertencia avisando que debemos crear una zona primero en la opción paralela.

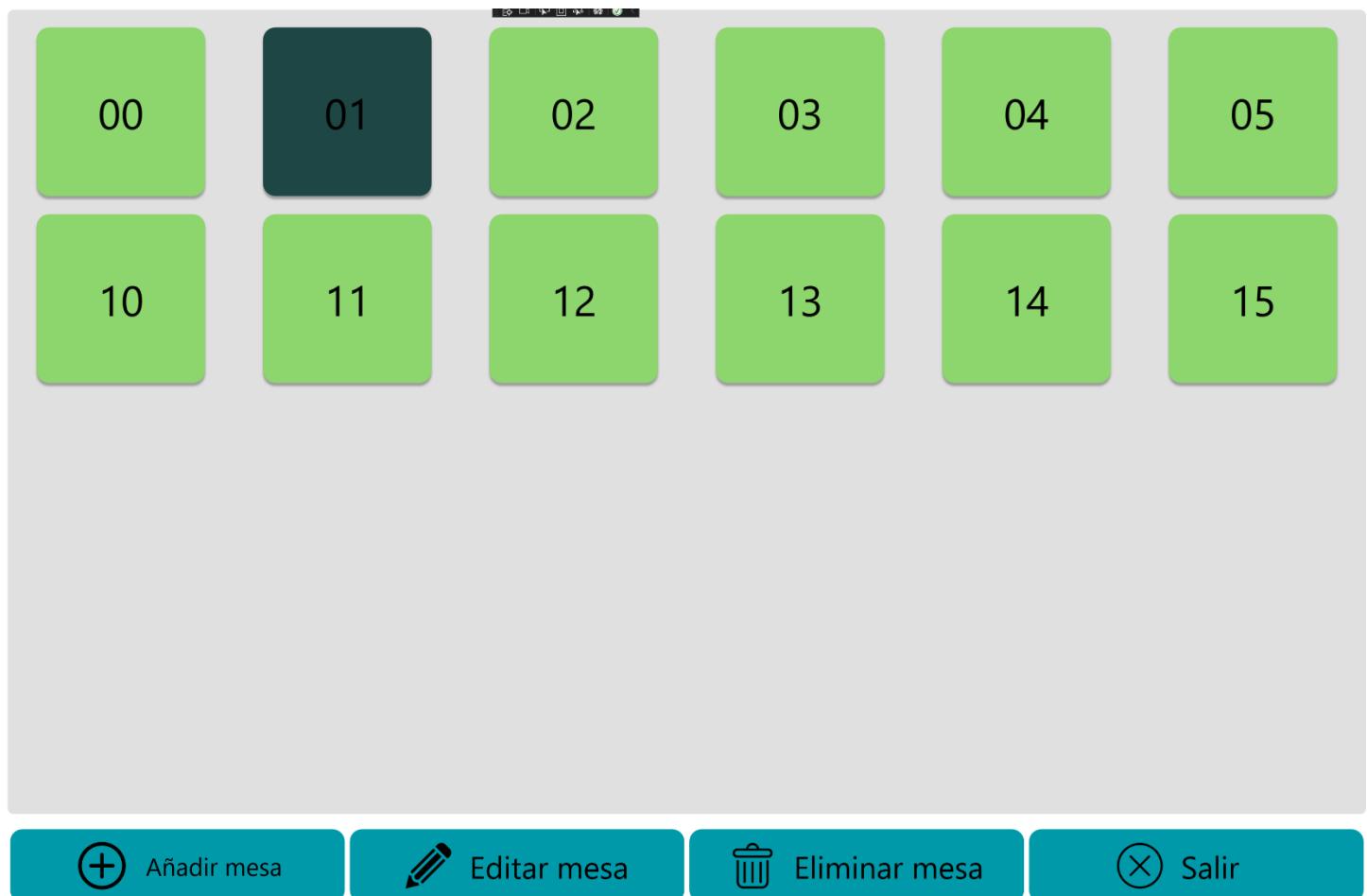
The screenshot shows the 'Mesas' (Tables) screen. On the left, there is a sidebar titled 'Zonas' (Areas) with the following items: Comedor, Terraza, Fuente, Piscina, Sala 1, Sala 2, Sala 4, Sala 5, Sala 6, Sala 7, and Sala 8. Below this sidebar is an 'Opciones' (Options) button. The main area displays a grid of 15 green rounded rectangles, each containing a number: 00, 01, 02, 03, 04, 05, 10, 11, 12, 13, 14, and 15. At the bottom of the screen are two buttons: a dark blue button with a plus sign and the text 'Añadir mesa' (Add table), and a teal button with a white 'X' and the text 'Salir' (Exit).

Si pulsamos en añadir mesa nos saldrá la siguiente ventana para verificar los datos

El numero de comensales se refiere al máximo de ocupantes que podrá tener esa mesa.

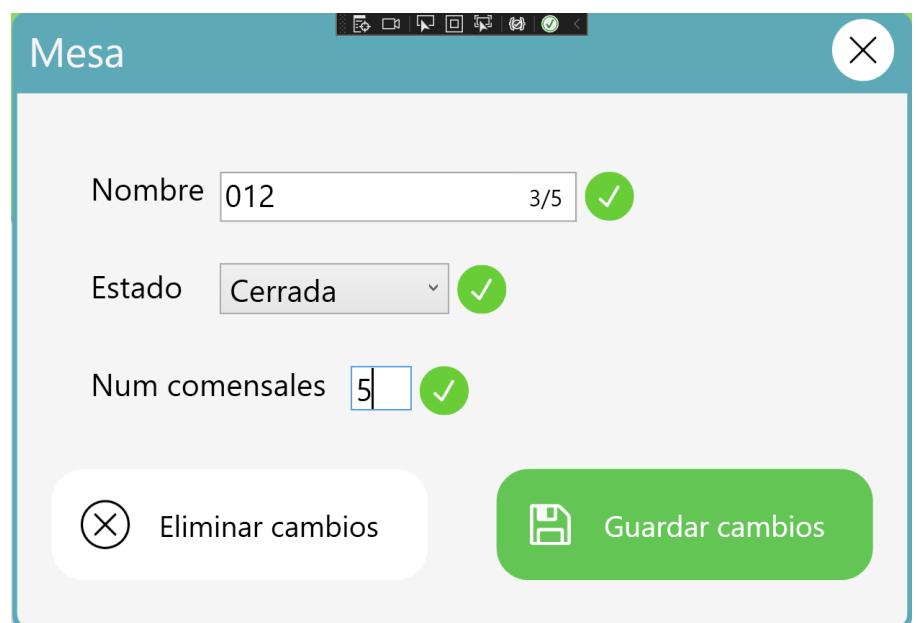
The screenshot shows a 'Opciones' (Options) dialog box. It has a title bar with icons and a close button. Inside the dialog, there are two input fields: 'Nombre de la Mesa' (Table name) with the value 'nueva' and a checked checkbox, and 'Número de comensales' (Number of guests) with the value '12' and a checked checkbox. At the bottom of the dialog are two buttons: 'Cancelar' (Cancel) and 'Siguiente >' (Next).

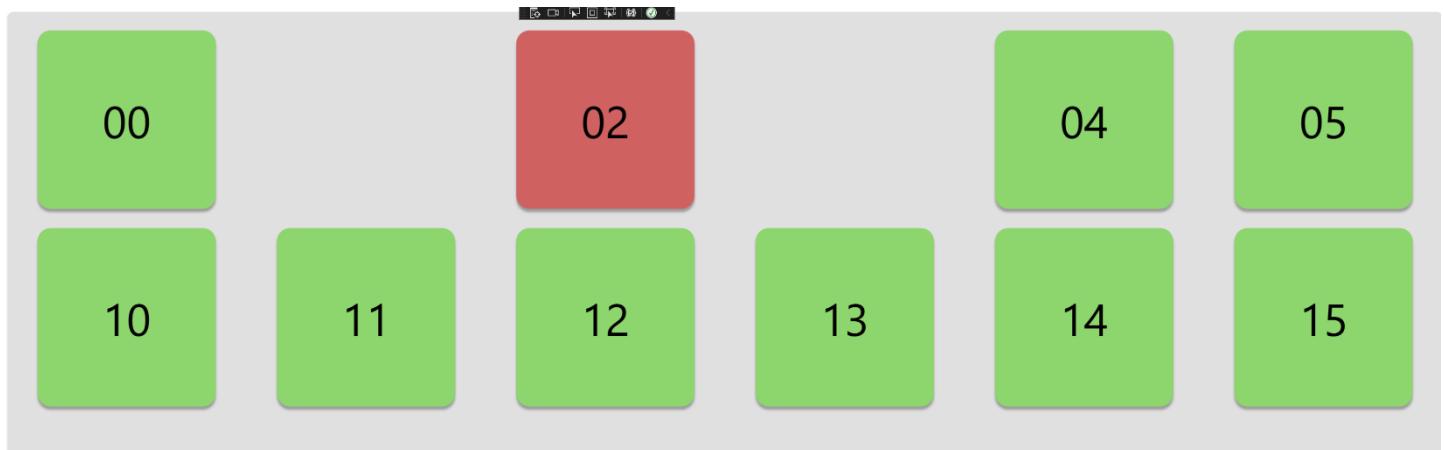
Después si seleccionamos una de las mesas aparecerán nuevas opciones.



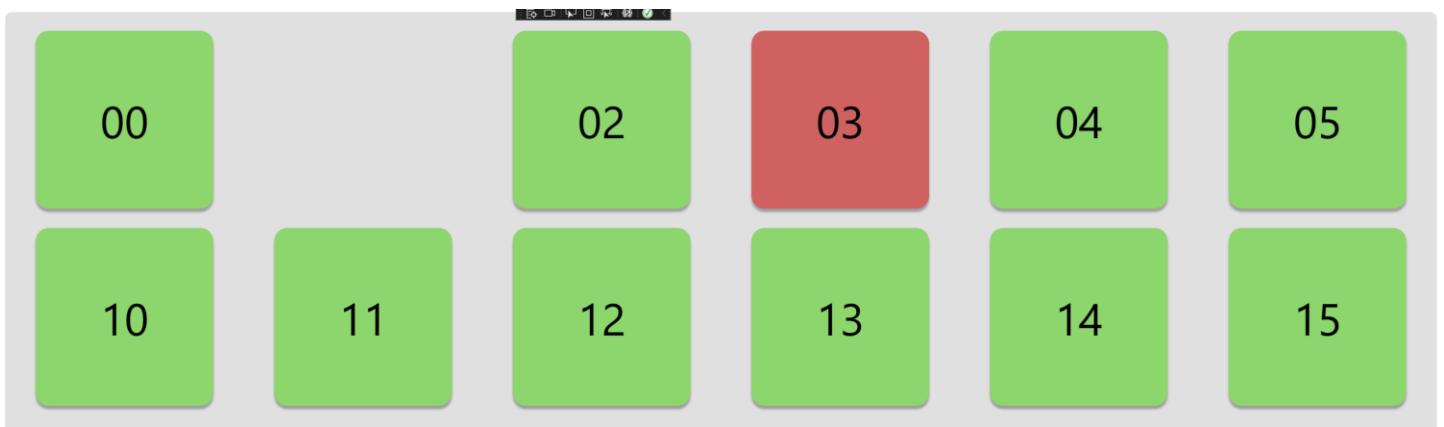
Editar mesa nos hará aparecer una ventana parecida a la de crear donde también podremos modificar el estatus.

Tanto si eliminamos una mesa como si ponemos que su estado sea cerrada no aparecerán en la vista normal del mapa, sin embargo, si la eliminamos se borrará definitivamente mientras que si se cambia su estado aparecerá en rojo en la ventana de administrador pero invisible en el mapa de empleado.





En este ejemplo tenemos una mesa eliminada y otra que su estatus es cerrada, a vista de empleado ninguna existe.



Sin embargo desde la vista de administrador podemos apreciar que la mesa 03 si que existe solo que su estatus es cerrado.



## 5.5 Empleados

Aquí podremos modificar las contraseñas, nombres de usuario, roles, agregar y eliminar usuarios.

Si pulsamos en el botón de agregar nos saltará un cuadro de diálogo que verificará la información que introduzcamos para la creación de un nuevo empleado.



### Crear empleado

Nombre  ✓

Email  ✓

Contraseña   ✓

Rol    ✓

Por otro lado si pulsamos sobre un empleado podremos editarlo y guardar los cambios así como eliminarlo

### Empleados

|           |
|-----------|
| Admin     |
| Empleado2 |
| Empleado1 |
| Empleado3 |
| Empleado4 |

**Información de empleado**

Nombre: Empleado22 ✓

Email: nuevo@mail.com ✓

Contraseña: 12345 ✓

Rol:

- Estándar
- Estándar
- Admin

**Opciones:**

- Agregar
- Eliminar
- Eliminar cambios
- Guardar cambios

## 5.6 Productos



### Productos

Al igual que con las zonas, para poder crear un producto primer debemos tener una familia creada y seleccionada, también podemos eliminar y modificar las familias y los productos.

Si pulsamos en agregar Familia nos saltará la siguiente ventana

### Crear Familia Producto

Nombre  ✓

× Cancelar Siguiente >



Si tenemos una familia seleccionada aparecerán nuevas opciones además de los productos que se cargarán a la derecha, donde también podremos agregar un nuevo producto en su correspondiente botón de agregar.

Agregar producto nos pide agregar una imagen, sin embargo no hemos tenido tiempo de guardarla en la base de datos para que se pueda acceder a ella desde

### Agregar producto



Nombre  ✓

Precio  ✓

Stock  ✓

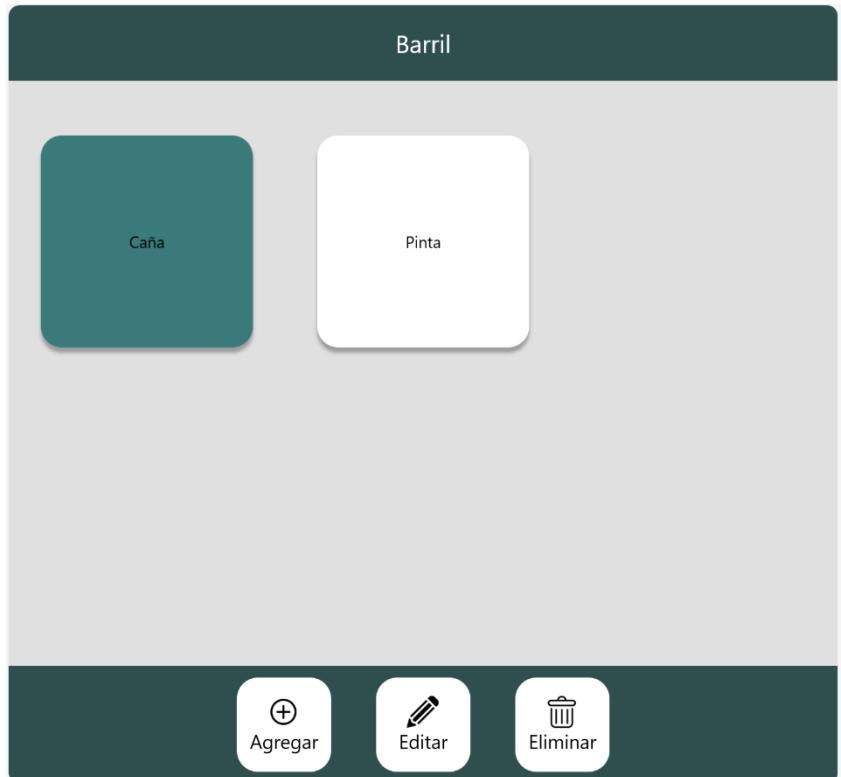
× Cancelar Siguiente >

### Productos

The screenshot shows a mobile application interface titled "Productos". On the left, a sidebar titled "Familia" contains five items: "Barril", "Carne", "Pescado", "Zumos", and "Menu". Each item has a small downward arrow icon to its right, indicating it can be expanded. Below the sidebar is a row of three buttons: "Agregar" (Add) with a plus sign, "Editar" (Edit) with a pencil icon, and "Eliminar" (Delete) with a trash bin icon. The main area is titled "Barril" and contains two items: "Caña" and "Pinta". At the bottom of this section is another "Agregar" button.

Luego podremos modificar o eliminar cualquier producto o familia pulsando en la opción correspondiente una vez tenemos uno de los elementos seleccionados. Cuidado, si eliminamos una familia se eliminarán todos los productos de ésta.

Se abrirá una ventana emergente muy parecida a la de crear familia y producto respectivamente.





### 5.7 Totales

En esta vista podremos ver todos los tickets que están vinculados a la actual caja, si pulsamos sobre cualquiera de ellos obtendremos más información.

The screenshot shows a software interface titled 'Totales'. At the top, there is a toolbar with various icons. Below the toolbar, the word 'Totales' is displayed in large letters. The main area contains two entries, each representing an open ticket:

- Mesa: 11    estado: Cerrado    Total: 100
- Mesa: 12    estado: Cerrado    Total: 30

Se mostrarán todos los tickets abiertos o cerrados en una lista y si cliclamos en alguno de ellos nos dirá su total.

