TecnoCampus
Escola Superior
Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grado en Ingeniería Informática de Gestión y Sistemas de Información

EAT & PAY

APLICACIÓN DE PAGOS INDIVIDUALES

PARA RESTAURANTES

Memoria

William Yang

Tutor: Josep Roure Alcobé

Ingeniería Informática de Gestión y Sistemas de Información

TecnoCampus
Mataró-Maresme

Abstract

This project is aimed to create a new application for restaurant management and also a new way of interacting with the restaurant by the customer, where they can order food, join a restaurant table, pay and much more.

Developed with MySQL as a database, Spring Boot as a back-end service to receive API requests and React Native as a front-end for the application, available on Android and IOS.

Resum

Projecte destinat a la creació d'una nova aplicació per a la gestió de restaurants i també una nova forma d'interactuar amb el restaurant per part del client, on podrà demanar menjar, unir-se a una taula d'un restaurant, pagar el comte i molt més.

Desenvolupat amb MySQL com a base de dades, Spring Boot com a servei back-end per a rebre peticions API i React Native com a front-end per l'aplicació, disponible amb Android e IOS.

Resumen

Proyecto destinado a la creación de una nueva aplicación para la gestión de los restaurantes y también una nueva forma de interactuar con el restaurante por parte del cliente, donde podrá pedir comida, unirse a una mesa de un restaurante, pagar y mucho más.

Desarrollado con MySQL como base de datos, Spring Boot como servicio back-end para recibir peticiones API y React Native como front-end para la aplicación, disponible en Android e IOS.

Índice

Índice de figuras

Índice de tablas

1. Objeto del proyecto

En este proyecto se pretende introducir una nueva modalidad de interacción entre restaurante y cliente, donde estará todo más digitalizado, ya sea realizar el pedido o pagar la cuenta, pero manteniendo todas las opciones que conllevan estas dos acciones.

La motivación del trabajo escogido se basa en poder llevar a los restaurantes otro paso más a la digitalización, con todas las ventajas que esto conlleva, y solventar algunos de los problemas típicos que hay a la hora de pagar la cuenta. A parte de funcionalidades extras que pueden aportar valor al restaurante.

Se pretende conseguir con este TFG mostrar una nueva forma de interacción entre restaurante y cliente, en una era en camino a la digitalización, este sería un paso más donde casi todos los procesos en los que está implicado un cliente en un restaurante podrían ser digitalizados.

Hemos detectado la necesidad de darle una solución a los pagos individuales en los restaurantes, para ello introducimos una aplicación donde centralizaremos todos los pasos que hay en un restaurante, a parte del pago, la reserva, visualizar la carta, realizar el pedido…

2. Estudio previo

En este apartado vamos a realizar un estudio de las aplicaciones más significativas en el mercado actual. Todas estas mantienen una cierta similitud con nuestro proyecto, la finalidad de realizar los pagos de manera individual.

-PayCui

Paycui es una app donde un cliente escanea un QR y pueden ver la carta, una vez hecho el pedido esta está asociada a esa mesa, el cliente mediante el QR escaneado puede hacer un seguimiento de lo que llevan en la cuenta, a parte también pueden pedir bebidas desde la misma app. A la hora de pagar pueden seleccionar que quieren pagar.

Paycui es una startup que está empezando en el sector de la restauración en Madrid fundado en 2015, en la actualidad tienen algunos clientes afiliados en la comunidad de Madrid.

El método por el que generan ingresos se basa en un porcentaje por transacción, por promociones y marketing, personalización de la app para el restaurante y venta de datos.

-Sundayapp

Sundayapp es una aplicación donde el cliente escanea el QR de la mesa para acceder a la carta, cuando realizan el pedido pueden hacer un seguimiento desde el QR escaneado.

A la hora de pagar pueden dividir la cuenta y dejar propinas, se trata de una aplicación web, por lo que no es necesario descargar la aplicación

Año de fundación 2021 y operan en Paris, London, Madrid & Atlanta.

Se trata de un método dedicado a agilizar el pago, escanear QR y pagar.

-Sumo Pedidomovil:

El camarero proporciona un QR a la mesa, el comensal escanea el QR, acceden a la web donde encuentran toda la carta y pueden realizar el pedido.

La funcionalidad es realizar los pedidos de buffet japones directamente desde el móvil, el pago se realiza de la forma tradicional.

Aplicación usada en la cadena de restaurante SUMO.

-Yumminn:

Funcionamiento:

El camarero atiende a los comensales para apuntar el pedido, a la hora de pagar los clientes tienen a su disposición un QR para acceder a la cuenta de esa mesa, donde pueden dividir el pago y pagar toda la mesa desde el móvil.

-Extenda Go:
Extenda Go ofrece un servicio TPV personalizado para cada cliente, dentro del mercado de la hostelería ofrece un servicio para aceptar pedidos en línea, como también realizar pedidos desde la aplicación móvil o desde la web. Estos pedidos pueden ser visualizados a través de la pantalla digital que ofrecen.

-TPV con cuenta personalizable:
Existen restaurantes que usan un TPV que pueden gestionar los pagos, a través de una pantalla pueden seleccionar los productos de una mesa para hacer el cobro individualmente, el cliente va a la barra a solicitar la cuenta al camarero y este pregunta a cada cliente que ha consumido para seleccionarlo en la cuenta total de la mesa y así generar la cuenta individual y cobrar individualmente.

A partir de estas aplicaciones podemos ver que se ha identificado el problema de los pagos en los restaurantes y de que solución le dan a este. Muchos restaurantes tienen que lidiar con dividir la cuenta por cliente de manera manual, lo que requiere tiempo y molestias a los clientes del restaurante al formarse colas a la hora de cobrar. Hay restaurantes que no aceptan pagos divididos dando un peor servicio a los clientes.
En el caso del TPV inteligente este solo permite el pago en la barra, donde el camarero va preguntando por cliente que ha consumido y genera su cuenta manualmente por la pantalla del TPV, esto suele generar colas y acumulación de personas en la entrada del restaurante.
Por otra parte, ExtendaGo si ofrece a través de la aplicación propia un servicio que cubre estas necesidad de realizar pedidos desde el móvil y pagar dichos pedidos desde el móvil, donde serán gestionados a través de una tableta digital propia de la marca.

La aplicación que proponemos en este proyecto tiene previsto todos los casos posibles que se pueden dar a la hora de cobrar y también ofrecer muchos más aspectos que pueden mejorar la relación entre cliente y restaurante. Por ejemplo, a partir de los datos de un cliente en un restaurante este puede saber que le gusta al cliente y así poder ofrecerle promociones u ofertas personalizadas para sus clientes.

También ofrecemos facilidad al restaurante para gestionar su restaurante, ya sea la carta, los pedidos o los pagos, entre otras muchas más funcionalidades.

3. Objetivos y alcance

Objetivo general:

- Crear una aplicación sencilla y flexible que dé solución a los restaurantes y a los clientes a la hora de pagar y a parte añadir funcionalidades para que puedan manejar todos los procesos por los que pasa el cliente en el restaurante desde la aplicación.

Objetivos del producto:

- Crear una aplicación que facilite la gestión del restaurante ya sea cara al cliente o cara al restaurante. Los restaurantes podrán gestionar los pagos y los pedidos desde la app, al igual que los clientes podrán realizar los pedidos y pagar desde la aplicación.
- Facilitar pedidos en restaurantes, los clientes podrán hacer los pedidos desde el móvil, este pedido llegará a un terminal de nuestra empresa y pasará a cocina.
- Facilitar el pago de la cuenta en restaurantes, los clientes podrán pagar la cuenta como ellos quieran directamente desde el móvil.
- Obtener datos de los clientes, para la venta de estos o permitir a los restaurantes ofrecer promociones u ofertas personalizadas a sus clientes.

Objetivos del cliente:

- Ofrecer al restaurante una aplicación que les facilite el trato con el cliente, podrán gestionar las reservas, los pedidos, los pagos, las ofertas y promociones directamente desde la aplicación.
- Ofrecer al restaurante un nuevo método de TPV digital.
- Añadir una nueva experiencia para el cliente, los clientes podrán realizar pedidos como ellos quieran y pagar como ellos quieran y todo desde el móvil.
- Atraer nuevos clientes, desde la app los clientes podrán ver un listado de los restaurantes más cercanos.
- Facilidad a la hora de cobrar a los clientes, los clientes podrán pagar directamente desde el móvil como ellos quieran.

- Tener el control de los comensales desde la aplicación, recomendar platos, promociones, ofertas, postres…
- Asignar una mesa directamente al cliente a la hora de que hagan una reserva desde la app, la aplicación podrá gestionar las mesas automáticamente a partir de las reservas que les van llegando al restaurante.
- Obtener pedidos para recoger, el restaurante podrá gestionar pedidos que les lleguen para recoger.

Objetivos del público potencial:

- Ofrecer al cliente una nueva manera de interactuar con el restaurante, donde podrán realizar reservas, los pedidos de mesa y pagar de la forma en la que quieran los comensales.
- Poder hacer pedidos individuales por comensal, al hacer el pedido por persona el comensal tiene a su cuenta lo que él ha pedido.
- Poder ver toda la carta desde la app, el cliente podrá ver toda la carta desde el móvil y hacer el pedido.
- Realizar el pago de la cuenta individualmente por comensal, al hacer el pedido individualmente podrán pagar solamente su parte, si así es como quieren pagar la cuenta todos los comensales.
- Realizar el pago de la cuenta entre todos los comensales
- Público potencial gente joven/adolescentes, poder tener todo el control de lo que hacen en el restaurante desde el móvil.
- Público potencial grupo de gente joven/adolescentes, poder dividir la cuenta de una manera rápida y directa, cada uno hace su pedido, cada uno paga lo que ha pedido.
- Poder realizar una reserva y añadir a todos los comensales, el cliente podrá realizar una reserva en un restaurante y añadir a todos los comensales que deberán de tener cuenta en la aplicación también.
- Poder realizar el pedido antes de ir al restaurante, los clientes podrán realizar un pedido individual o el de toda la mesa antes de asistir, para así al llegar a la hora reservada tener los platos preparados.
- Poder seleccionar productos compartidos y con quien, los clientes podrán seleccionar platos a compartir y añadir con que comensales compartirlo, para así a

la hora de pagar tener dividido el precio entre todos los comensales que lo han compartido.

- Poder pagar toda la mesa, el cliente podrá seleccionar pagar la cuenta de toda la mesa.
- Poder pagar entre todos, el cliente podrá seleccionar pagar la cuenta entre todos los comensales, dividir el precio total entre todos los comensales.
- Poder pagar productos de otros comensales, el cliente podrá pagar productos de otros comensales o todos los productos de otros comensales.

A partir de todos los objetivos redactados el proyecto del TFG tendrá un alcance establecido, la aplicación se centrará en el problema principal, los pagos individuales.
Para ello la aplicación del TFG podrá realizar pedidos por cliente y podrá realizar el pago individualmente, todo esto directamente desde la aplicación.

4. Metodología

Para este proyecto usaremos la metodología Waterfall, adaptado a un entorno de trabajo unipersonal, para un proyecto de una sola persona vemos que waterfall se puede adaptar de una manera más sencilla que otras metodologías y también al prescindir solamente de una persona, esta conoce perfectamente el producto que quiere desarrollar y así no nos encontramos con algunos de los inconvenientes de la metodología waterfall.

Pasos a seguir:

1. Toma de requerimientos, en base a la idea del proyecto se redactarán los requerimientos a desarrollar para obtener el producto mínimo viable.
2. Diseño, se diseñará la base de datos usando un modelo lógico y posteriormente el modelo relacional, se diseñará el back end, el patrón a usar y la estructura del proyecto (paquetes y clases) y por último se diseñará el front end, la estructura del proyecto y la interfaz de usuario.
3. Desarrollo de todo el software, crear la base de datos con todas las tablas respectivas, crear todos los paquetes y las clases con todas las funcionalidades necesarias del back end y por último desarrollar toda la estructura de front end con todas sus funcionalidades.
4. Testing, se testeará todo el software desarrollado y se comprobará que se hayan cumplido todos los requerimientos.
5. Corrección de errores, a partir de los errores de testing se aplicarán las correcciones necesarias para el correcto funcionamiento.
6. Despliegue en dispositivos móviles.

Todos los pasos mencionados anteriormente se realizarán por una sola persona.

5. Requerimientos funcionales y tecnológicos

Requerimientos funcionales:

- La aplicación ha de permitir al restaurante añadir productos a su carta.
- La aplicación ha de permitir al restaurante editar los productos de su carta.
- La aplicación ha de permitir al restaurante añadir descuentos a su carta.
- La aplicación ha de permitir al restaurante aceptar y rechazar reservas.
- La aplicación ha de permitir al restaurante aceptar pedidos o definirlas que se acepten automáticamente.
- La aplicación ha de permitir al restaurante cancelar reservas.
- La aplicación ha de permitir al restaurante asignar o modificar las mesas asignadas en una reserva.
- La aplicación ha de permitir al restaurante enviar recomendaciones de platos a los comensales.
- La aplicación ha de permitir al restaurante añadir promociones, ofertas o códigos de descuento para sus clientes o clientes modelo del tipo de restaurante o comida ofertada.
- La aplicación ha de permitir al restaurante añadir un descuento en una mesa.
- La aplicación ha de permitir al restaurante añadir platos a una mesa.
- La aplicación ha de permitir al restaurante quitar platos de una mesa.
- La aplicación ha de permitir al restaurante asignar una mesa a los clientes.
- La aplicación ha de permitir al restaurante añadir fotos a los productos.
- La aplicación ha de permitir al restaurante realizar pedidos para las mesas desde la aplicación.
- La aplicación ha de permitir al restaurante gestionar los pedidos.
- La aplicación ha de permitir al restaurante recibir pagos de pedidos.
- La aplicación ha de permitir al cliente crearse una cuenta.
- La aplicación ha de permitir al cliente añadir un método de pago.
- La aplicación ha de mostrar los restaurantes cercanos al cliente en la pantalla inicial.
- La aplicación ha de permitir al cliente seleccionar un restaurante.
- La aplicación ha de permitir al cliente realizar una reserva.

- La aplicación ha de permitir al cliente escanear un código QR para unirse a una mesa asignada en el restaurante.
- La aplicación ha de permitir al cliente visualizar la carta del restaurante.
- La aplicación ha de permitir al cliente añadir productos a su pedido.
- La aplicación ha de permitir al cliente editar su pedido.
- La aplicación ha de permitir al cliente finalizar su pedido.
- La aplicación ha de permitir al cliente enviar un pedido sin tener que esperar a que toda la mesa haya finalizado los pedidos.
- La aplicación ha de permitir al restaurante recibir pedidos.
- La aplicación ha de permitir al cliente visualizar un resumen de su cuenta.
- La aplicación ha de permitir al cliente realizar el pago de su cuenta.
- La aplicación ha de permitir al cliente seleccionar platos para compartir y asignar con que usuarios de la mesa compartirlos.
- La aplicación ha de enviar una notificación de que se quiere compartir un plato con ese comensal.
- La aplicación ha de permitir al cliente aceptar o rechazar un plato compartido.
- La aplicación ha de permitir al cliente pagar toda la cuenta de la mesa.
- La aplicación ha de permitir al cliente pagar platos de otros comensales de la mesa.
- La aplicación ha de permitir al cliente visualizar la cuenta total de la mesa, separada por comensales.
- La aplicación ha de permitir al cliente unirse a una mesa de manera anónima, sin cuenta de usuario.
- La aplicación ha de permitir al cliente crear una cuenta cuando haya pagado la cuenta y asociar esta cuenta con su cuenta de usuario.
- La aplicación ha de permitir al restaurante asignar un número de mesa y con el número de asientos.
- La aplicación ha de permitir al restaurante editar sus mesas.
- La aplicación ha de asignar automáticamente una mesa dependiendo del número de comensales.

- La aplicación ha de permitir al cliente realizar un pedido para una hora concreta, solo se hará efectiva con el pago previo.
- La aplicación ha de permitir al restaurante notificar de falta de pago de algún pedido.
- La aplicación ha de poder cobrar automáticamente a un cliente que se le haya notificado de falta de pago al pasar un tiempo determinado.
- La aplicación ha de poder gestionar las mesas automáticamente.
- La aplicación ha de permitir al restaurante "abrir" y "cerrar" el restaurante.
- La aplicación ha de permitir al restaurante seleccionar los platos que ya se han servido.
- La aplicación ha de mostrar al cliente el menú (menú interactivo) de un restaurante, visualizar la carta, reservar, entrar a una mesa e información del establecimiento.
- La aplicación ha de obtener los datos del GPS del dispositivo para saber dónde se encuentra el cliente y así mostrar los restaurantes más cercanos.

Requerimientos tecnológicos:

- La aplicación ha de funcionar en móviles Android e IOS.
- La aplicación ha de estar disponible en castellano, catalán e inglés.
- La aplicación no ha de tardar más de 15 segundos en mostrar los restaurantes cercanos al cliente.
- La aplicación no ha de tardar más de 15 segundos en entrar al menú de un restaurante.
- La aplicación no ha de tardar más de 15 segundos en asignar una mesa de una reserva.
- La aplicación no ha de tardar más de 15 segundos en realizar un pedido.
- La aplicación no ha de tardar más de 15 segundos en realizar un pago.
- La aplicación no ha de tardar más de 15 segundos en cualquier gestión por parte del restaurante
- La aplicación no ha de tardar más de 15 segundos en cualquier gestión por parte del cliente.
- El sistema ha de proporcionar una disponibilidad del 99% cuando un usuario quiera acceder a la aplicación.

- El sistema ha de poder operar con más de 500 usuarios con sesiones concurrentes (inicialmente).

Requerimientos a desarrollar, (algunos de la lista total de requerimientos):

- La aplicación ha de permitir al restaurante añadir productos a su carta.
- La aplicación ha de permitir al restaurante editar los productos de su carta.
- La aplicación ha de mostrar los restaurantes al cliente en la pantalla inicial.
- La aplicación ha de permitir al cliente seleccionar un restaurante.
- La aplicación ha de permitir al cliente unirse a una mesa mediante un código de mesa.
- La aplicación ha de permitir al cliente visualizar la carta del restaurante.
- La aplicación ha de permitir al cliente añadir productos a su pedido.
- La aplicación ha de permitir al cliente editar su pedido.
- La aplicación ha de permitir al restaurante recibir pedidos.
- La aplicación ha de permitir al cliente visualizar un resumen de su cuenta.
- La aplicación ha de permitir al cliente realizar el pago de su cuenta.
- La aplicación ha de permitir al cliente visualizar la cuenta total de la mesa, separada por comensales.
- La aplicación ha de mostrar al cliente el menú (menú interactivo) de un restaurante, visualizar la carta, entrar a una mesa e información del establecimiento.
- La aplicación ha de permitir al cliente pagar productos de otros comensales de la misma mesa.
- La aplicación ha de permitir a los usuarios crear una cuenta.

Estos requerimientos a desarrollar también establecen el producto mínimo viable, centrando todo el enfoque a que un cliente pueda generar pedidos y pagar su cuenta desde la aplicación.

6. Desarrollo

En este apartado se explicará todo el contenido realizado para el desarrollo de la aplicación.

6.1. Base de datos

En este apartado explicaremos el diseño seguido y los datos usados para la ceración de la base de datos.

6.1.1. Modelo lógico

- La base de datos se ha realizado con MySQL.
- Representación modelo lógico en la Fig 6.1.1.1

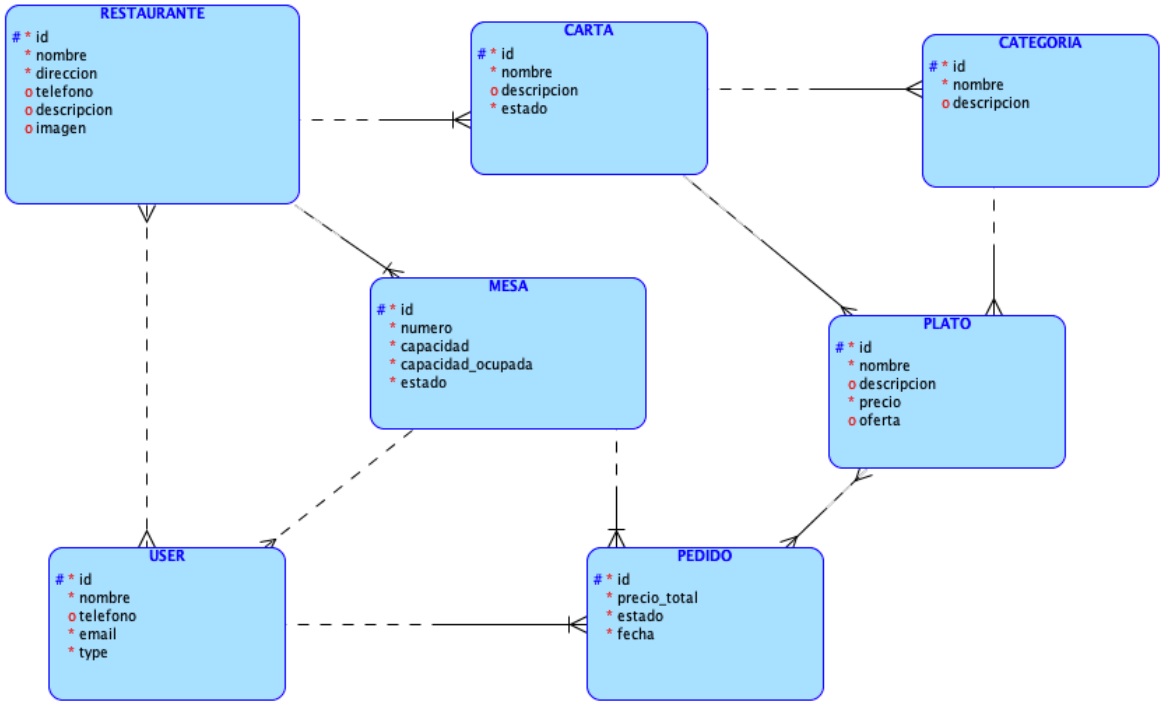


Fig 6.1.1.1. Modelo lógico

- Un restaurante puede o no tener una a varias cartas.
- Un restaurante puede o no tener varias mesas.
- Un restaurante puede o no tener varios usuarios, que son los que dirigen ese restaurante.
- Una carta es solamente de un restaurante.
- Una carta puede o no tener varias categorías.
- Una carta puede o no tener varios platos.

- Una categoría tiene que ser solamente de una carta.
- Una categoría puede o no tener varios platos.
- Un plato debe de ser solamente de una carta.
- Un plato debe de ser solamente de una categoría.
- Un plato puede o no ser de un pedido.
- Un pedido debe de tener si o si uno o varios platos.
- Un pedido debe de ser solamente de una mesa.
- Un pedido debe de ser solamente de un usuario.
- Una mesa debe de ser solamente de un restaurante.
- Una mesa puede o no tener varios usuarios.
- Un usuario puede o no tener varios pedidos.
- Un usuario puede o no tener varios restaurantes (administrar restaurantes).
- Un usuario puede o no estar en una mesa.

6.1.2. Modelo relacional

- Representación modelo relacional en la Fig 6.1.2.1.

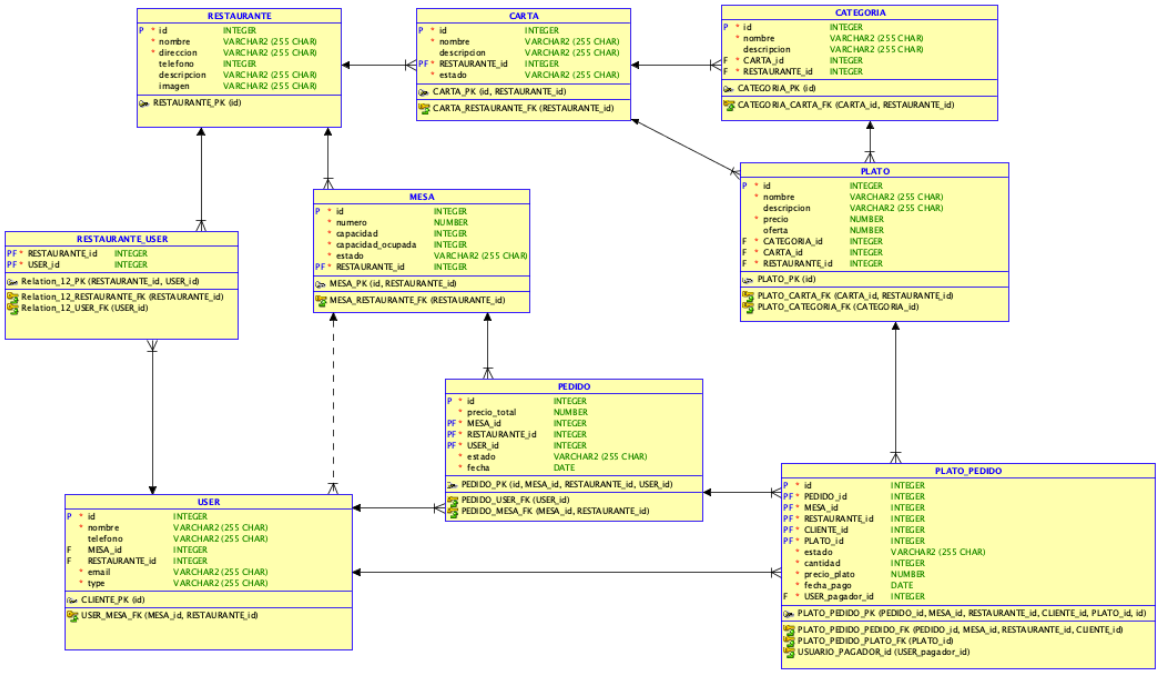


Fig 6.1.2.1. Modelo relacional.

- Contamos con un total de 9 tablas
- Tabla restaurante: contiene la clave primaria id, un nombre, una dirección, un teléfono, una descripción y un imagen. La imagen contiene la URL donde se almacena dicha imagen, en este proyecto hemos usado un repositorio de imágenes de comida gratuita.
- Tabla carta: contiene la clave primaria id, un nombre, una descripción, el id del restaurante y un estado. Un restaurante puede tener varias cartas y mediante el estado se gestionan las cuales estarán activas e inactivas. Una carta es solamente de un restaurante.
- Tabla categoría: contiene la clave primaria id, un nombre, una dirección, la id de la carta y la id del restaurante. Una carta puede contener varias categorías y cada categoría pertenece solamente a una carta y a un restaurante.
- Tabla plato: contiene la clave primaria id, un nombre, una descripción, un precio, una oferta, id de la categoría, de la carta y del restaurante. Una categoría puede tener varios platos y una carta también puede tener varios platos. Cada plato pertenece solamente a un restaurante, a una carta y a una categoría.

  Para mantener el precio original en caso de querer aplicar un descuento usamos el campo oferta, aplicando el porcentaje que se quiere descontar al precio inicial.
- Tabla mesa: contiene la clave primaria id, un número de mesa, una capacidad, una capacidad ocupada, un estado y la id del restaurante. El número de mesa corresponde con el número de mesa del restaurante. La capacidad ocupada indica el número de usuarios que están en la mesa y el estado indica si la mesa está ocupada o libre.
- Tabla user: contiene la clave primaria id, un nombre, un teléfono, un email, un tipo, la id de una mesa y de un restaurante. El tipo indicará si el usuario es normal (“1”) o si administra un restaurante (“2”). La id de mesa y de restaurante son opcionales, se introducirán solamente cuando un usuario se sienta en una mesa, de esta manera asignamos a dicho cliente en una mesa de un restaurante.
- Tabla restaurante user: contiene la id del restaurante y del usuario, esta tabla se usa como lista para obtener los restaurantes o el restaurante que gestiona un usuario.
- Tabla pedido: contiene la id del pedido, el precio total, el estado, la id del restaurante, de la mesa y del usuario. El estado indica el estado del pedido.
- Tabla plato pedido: contiene la clave primaria id, el estado, la id del pedido, la id del restaurante, de la mesa, del cliente, del plato, la cantidad, el precio del plato, el usuario pagador del plato y la fecha de pago.

Esta tabla se usa se manera de lista para obtener todos los paltos de un pedido.

El precio del plato se usa para guardar el precio del plato en el momento que se hizo el pedido, usamos este campo para mantener el historial de pedidos del usuario acorde con el precio de cuando hizo el pedido.

El usuario pagador indica la id de la persona que pagó el plato, en caso de ser el mismo que realizó el pedido no se modificará este campo.

- Representación de las tablas creadas en MySQL, usamos DataGrip o Intellij de JetBrains para visualizar el contenido de la base de datos, Fig 6.1.2.2.

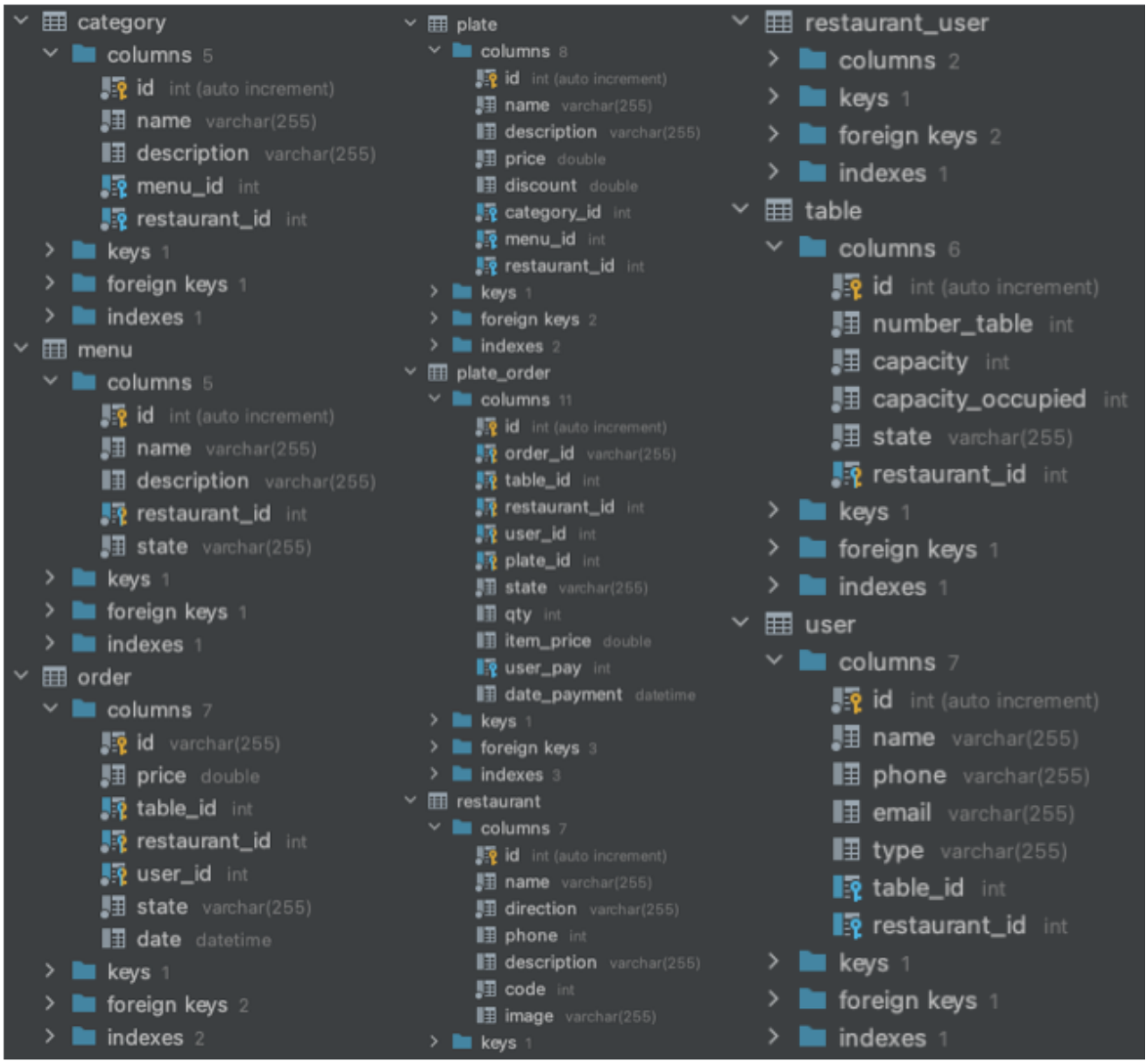


Fig 6.1.2.2. Base de datos creada con DataGrip.

6.2. Back End

- El back end se ha desarrollado usando la tecnología Spring Boot mediante Java como lenguaje de programación.
- Este servicio aloja todas las peticiones REST APIs necesarias para el funcionamiento del front end, realiza las acciones debidas y actualiza la base de datos.
- Se ha desarrollado usando Intellij IDE de JetBrains, siguiendo la estructura de patrón por capas como podemos observar en la Fig. 6.2.1.

  El paquete api aloja todas las peticiones REST API, estas peticiones son pasadas a la capa "application" donde encontramos los controladores, lugar donde reside la lógica de nuestra aplicación, los controladores pasan la información final a los DAOS para que estos la introduzcan en la base de datos o también piden información a los DAOS para obtenerla de la base de datos.

  También en la capa "application" encontramos los DTOS, que son representaciones de los objetos para pasar información entre las capas, las interfaces de los DAOS y excepciones propias de la aplicación. En el paquete "configuration" encontramos diferentes configuraciones para el correcto funcionamiento del back-end.

  En el paquete "domain" encontramos los objetos de la aplicación que son los que podemos observar en la Fig 6.2.2 y por último la capa "persistance" que encontramos los DAOS, donde implementan las interfaces y son los encargados de acceder a la base de datos y pasar los datos a objetos de nuestra aplicación.

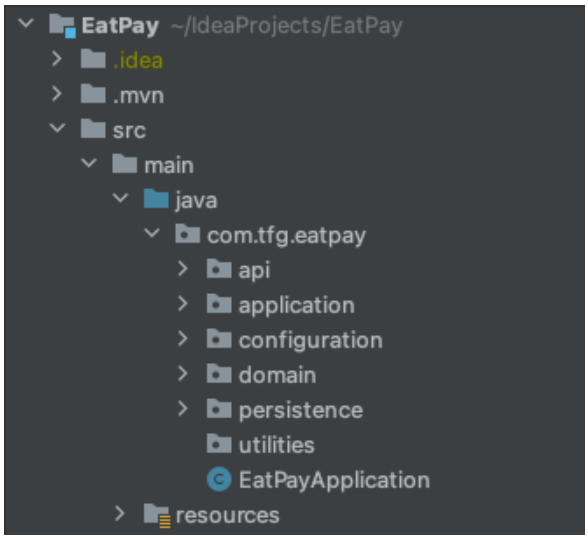


Fig. 6.2.1. Patrón capas.

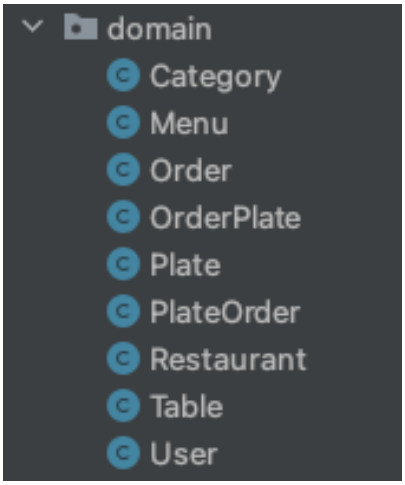


Fig. 6.2.2. Capa Domain.

- A partir del paquete api explicaremos todas las funcionalidades del back-end, dentro del paquete api encontramos los diferentes “Rest Controllers” que podemos observar en la Fig 6.2.3. Explicación representada en tabla, tal como se ve en la Tabla 6.2.4

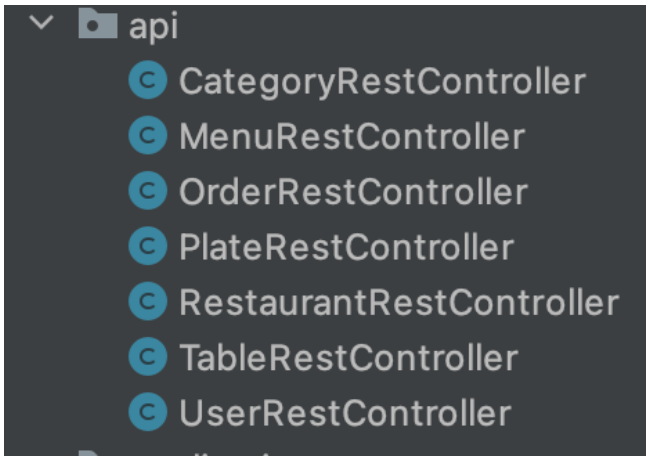


Fig. 6.2.3. Paquete api.

Petición	Tipo	Parámetros	Función
/category	POST	RequestBody CategoryDTO	Esta petición sirve para crear una categoría a través del parámetro CategoryDTO, se llama al controlador correspondiente donde creará la categoría a partir del DTO, para finalmente introducirla en la base de datos a través del DAO
/menu	POST	RequestBody MenuDTO	Esta petición sirve para crear menú a través del parámetro MenuDTO, se llama al controlador correspondiente donde creará el menú a partir del DTO, para finalmente introducirla en la base de datos a través del DAO
/order	POST	RequestBody OrderDTO	Esta petición sirve para crear una order a través del parámetro OrderDTO, este DTO contiene la lista de platos correspondiente con este pedido y la cantidad, también se crea una ID automática y aleatoria de 36 caracteres. Esta petición llama al controlador correspondiente donde creará el pedido para introducirla en la base de datos a través del DAO y por cada plato introducirá en el "plato pedido" de la base de datos la id del pedido, los datos del plato, la cantidad y el precio del plato

/order/get-by-user/ {userId}	GET	PathVariable userId	Esta petición sirve para obtener un listado de todos los pedidos hechos por un usuario. Dada la id del usuario obtenemos de la base de datos el listado de OrderDTO a través del DAO. Cada uno de los OrderDTO no contienen el listado de platos de ese pedido
/order/get/{userId}/ {restaurantId}	GET	PathVariable userId y restaurantId	Esta petición sirve para obtener un listado de todos los nuevos pedidos realizados por un usuario en un restaurante. Dada la id del usuario y del restaurante obtenemos de la base de datos el listado de OrderDTO de ese usuario en ese restaurante y con estado "NEW". Cada OrderDTO no contiene el listado de platos de ese pedido
/order /get-table-orders /{tableId} /{restaurantId}	GET	PathVariable tableId y restaurantId	Esta petición sirve para obtener un listado de todos los nuevos pedidos realizados en una mesa de un restaurante. Dada la id de la mesa y del restaurante obtenemos de la base de datos el listado de nuevos OrderDTO de esa mesa. Cada OrderDTO no contiene el listado de platos de ese pedido
/order/get-new-orders /{restaurantId}	GET	PathVariable restaurantId	Esta petición sirve para obtener un listado de todos los nuevos pedidos de un restaurante. Dada la id del restaurante obtenemos de la base de datos el listado de nuevos OrderDTO del restaurante

			Cada OrderDTO no contiene el listado de platos de ese pedido
/plate	POST	PlateDTO	Esta petición sirve para crear un plato, PlateDTO contiene la información del plato, la id del menú, de la categoría y del restaurante. La petición llama al controller responsable para introducirla en la base de datos a través del DAO
/plate/default	POST	PlateDTO	Esta petición sirve para crear un plato para el menú y categoría por defecto del restaurante. PlateDTO contiene la información del plato y la id del restaurante. La petición llama al controller responsable para obtener las id del menú y de la categoría por defecto del restaurante y añade el plato a la base de datos a través del DAO
/plate/get/{restaurantId}	GET	PathVariable restaurantId	Esta petición sirve para obtener todos los platos de un restaurante, a través del controller se obtiene un listado de PlateDTO, donde usa el DAO para obtener el listado desde la base de datos
/plate/get-cart/{orderId}	GET	PathVariable orderId	Esta petición sirve para obtener todos los platos de un pedido, a través del controller y del orderId obtenemos un listado de PlateOrderDTO, el controller usa el

			DAO correspondiente para obtener dicho listado. PlateOrderDTO contiene la id del pedido, la información del plato, la cantidad pedida de ese plato y el precio del plato
/plate/get-all-plates-cart /{userId}/{restaurantId} /{tableId}	GET	PathVariable userId, restaurantId y tableId	Esta petición sirve para obtener todos los platos pedidos de un cliente en una mesa de un restaurante, excepto aquellos platos que serán pagados por otro usuario. El controller responsable obtiene el listado de platos pedidos por un usuario en una mesa a través del DAO, finalmente elimina todos aquellos platos que serán pagados por otros usuarios y devuelve el listado de PlateOrderDTO
/plate/get-all-table-plates /{userId}/{restaurantId} /{tableId}	GET	PathVariable userId, restaurantId y tableId	Esta petición sirve para obtener todos los platos pedidos en una mesa de un restaurante, excepto las del usuario que realiza la consulta y también los platos que el usuario pagará de otro usuario. El controller responsable obtiene el listado de platos pedidos por una mesa a través del DAO, finalmente elimina todos aquellos platos pedidos por el usuario que realiza la consulta y aquellos que pagará de otro usuario y devuelve el listado de PlateOrderDTO

/plate/get-all-other-paying-plates /{userId}/{restaurantId} /{tableId}	GET	PathVariable userId, restaurantId y tableId	Eta petición sirve para obtener todos los platos que pagaré de otros usuarios. El controller responsable obtiene el listado de platos pedidos por una mesa a través del DAO, finalmente selecciona aquellos que el usuario que realiza la petición va a pagar exceptuando los que él ha pedido y devuelve el listado de PlateOrderDTO
/plate/change-pay-user/{plateOrderId}/{userId}	POST	PathVariable plateOrderId y userId	Esta petición sirve para cambiar el usuario pagador de un plato pedido. El controller responsable obtiene el plato pedido a través del DAO para comprobar que el pagador y el que realizó el pedido no sea el mismo usuario. En caso negativo actualiza el pagador a través del DAO
/plate/pay-plates-ordered/{userId}/{tableId}	POST	RequestBody OrderDTO, PathVariable userId y tableId	Esta petición sirve para pagar todos los platos de la lista que contiene OrderDTO, contiene los platos pedidos por el usuario que realiza la petición como aquellos que pagará de otros usuarios. El controller responsable además de marcar el plato como pagado se encargará de gestionar si todos los platos de un pedido están pagados, en ese caso marcará como pagado el pedido y por último se elimina el usuario de la mesa

/plate/get-order-plate/{id}	GET	PathVariable id	Esta petición sirve para obtener el "plato pedido" a través de su id. El controller correspondiente devuelve el "plato pedido" a través del DAO
/plate/get-user/{orderPlateId}	GET	PathVariable orderPlateId	Esta petición devuelve los datos de un usuario a través de la id del "plato pedido". El controller responsable obtiene los datos del "plato pedido" a través de su id, para posteriormente obtener el usuario a través de la id del usuario de "plato pedido", estas consultas se realizan a través del DAO
/plate/get-new-by-table/{tableId}	GET	PathVariable tableId	Esta petición sirve para obtener todos los platos pedidos de una mesa, pero sin platos repetidos. El controller responsable obtiene una lista de todos los platos pedidos de una mesa, recorre dicha lista y añade el plato a la lista a devolver o en caso de existir suma la cantidad
/plate/edit	POST	RequestBody PlateDTO	Esta petición sirve para editar un plato existente. El controller responsable actualiza la información del plato usando el DAO y la información del plateDTO
/restaurant	POST	RequestBody RestaurantDTO	Esta petición sirve para crear un restaurante. El controller responsable inserta los datos de restaurantDTO en la base de datos a través del DAO

/restaurant/get	GET	NO	Esta petición sirve para obtener todos los restaurantes disponibles. El controller responsable obtiene un listado de RestaurantDTO de la base de datos a través del DAO
/restaurant/get/{restaurantId}	GET	PathVariable restaurantId	Esta petición sirve para obtener el restaurante dada su id. El controller responsable obtiene de la base de datos a través del DAO el RestaurantDTO con la id restaurantId
/restaurant/get-from-user/{userId}	GET	PathVariable userId	Esta petición sirve para obtener el restaurante que un usuario gestiona. El controller responsable obtiene de "restaurant_user" la id del restaurante que gestiona el userId, posteriormente obtiene los datos del restaurante con esa id y devuelve el RestaurantDTO
/table/get/{tableId}	GET	PathVariable tableId	Esta petición sirve para obtener los datos de una mesa dada su id. El controller responsable obtiene de la base de datos a través del DAO los datos de la mesa con id "tableId"
/table/get-occupied/{restaurantId}	GET	PathVariable restaurantId	Esta petición sirve para obtener un listado de todas las mesas ocupadas del restaurante. El controller responsable obtiene un listado de TableDTO a través del DAO de todas las mesas con estado

			"OCCUPIED" del restaurante con id "restaurantId"
/user	POST	RequestBody UserDTO	Esta petición sirve para crear un usuario. El controller responsable inserta los datos del UserDTO en la base de datos a través del DAO
/user/get/{email}	GET	PathVariable email	Esta petición sirve para obtener los datos de un usuario dado su email. El controller responsable obtiene el UserDTO con el "email (columna base de datos)" "email (parámetro)" de la base de datos a través del DAO
/user/get-by-id/{userId}	GET	PathVariable userId	Esta petición sirve para obtener los datos de un usuario dada su id. El controller responsable obtiene el UserDTO con la id "userId" de la base de datos a través del DAO

Fig. 6.2.4. Tabla peticiones api.

- Para la inserción u obtención de datos, por ejemplo, nuevos pedidos, nuevos platos, pedidos realizados, datos del usuario, entre otros muchos, utilizamos un mensaje en formato JSON.
- JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos, usado para enviar los objetos del back-end al front-end y viceversa.

  Los "{ }" indican objetos y los "[ ]" indican un listado.
- Ejemplo de objeto JSON:
  - UserDTO: contiene la información de un usuario

```
{
 "email": "xavier@eatpay.com",
 "id": "31",
 "name": "Xavier Sanchez",
 "phone": "542345675",
 "restaurant_id": null,
 "table_id": null,
 "type": 1,
}
```


- OrderDTO: creación de un pedido que contiene 5 platos de Costillas.


```
{
 "plates": [
  Object {
   "id": "18",
   "plate": Object {
    "category_id": "1",
    "description": "Costillas a la barbacoa 2.5 KG",
    "discount": "0.0",
    "id": "18",
    "menu_id": "3",
    "name": "COSTILLAS",
    "price": "32.95",
    "restaurant_id": "2",
   },
   "qty": 5,
   "totalPrice": 164.75,
  },
 ],
 "price": "164.75",
 "restaurant": {
  "id": "2",
 },
 "state": "NEW",
```

```
 "table": {
  "id": "2",
 },
 "user": {
  "id": "31",
 },
}
```


- OrderPlateDTO: contiene la información de un plato de un pedido


```
{
 "date_payment": null,
 "id": "451",
 "item_price": "24.95",
 "order_id": "9ef84f23-74c6-4265-8a7a-d4c1cc7d8695",
 "plate_id": "5",
 "qty": "1",
 "restaurant_id": "2",
 "state": "NEW",
 "table_id": "2",
 "user_id": "34",
 "user_pay": null,
}
```


- List<PlateOrderDTO>: cada objeto indica el precio por el que se ha pedido, la cantidad y la información del plato


```
[
 {
  "id": "451",
  "item_price": "24.95",
  "plate": {
   "category_id": "1",
   "description": "Variedad de carnes",
   "discount": "0.0",
```

```
      "id": "5",
      "menu_id": "3",
      "name": "PAELLA CARNE",
      "price": "24.95",
      "restaurant_id": "2",
    },
    "qty": "1",
  },
  {
    "id": "452",
    "item_price": "12.95",
    "plate": Object {
      "category_id": "1",
      "description": "Burrito extragande",
      "discount": "0.0",
      "id": "6",
      "menu_id": "3",
      "name": "BURRITO PLUS",
      "price": "12.95",
      "restaurant_id": "2",
    },
    "qty": "1",
  },
]
```

6.3. Front End

- Todo el front end se ha desarrollado usando la tecnología de React Native.
- Aquí se encuentra la aplicación final con la que el usuario interactuará.
- La aplicación se encuentra bajo el paquete "app" y dentro de este encontramos los "assets", los "components", los "screens" y el "storage", como podemos observar en la Fig 6.3.1.

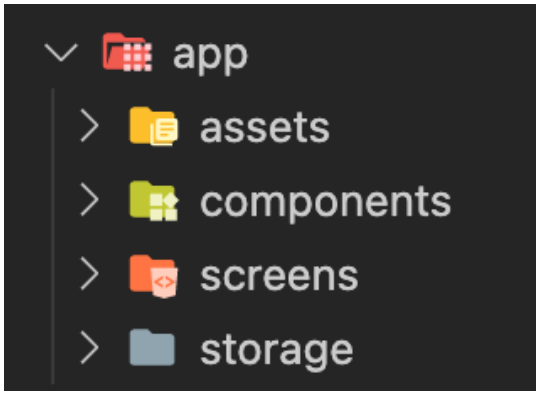


Fig. 6.3.1. Estructura front end.

- Dentro de "assets" encontramos los recursos estáticos que usaremos para la aplicación, en nuestro caso solamente tendremos las imágenes de nuestro logo.
- Seguidamente, en "components", encontramos todos los componentes propios de nuestra aplicación, en el apartado de tecnología usada se explica que son los componentes en React Native. Dentro de "components" también encontramos la carpeta "navigation" que contiene la programación de la navegación entre pantallas de nuestra aplicación.
- En la carpeta "screens" encontramos todas las pantallas que se usan en la aplicación.
- Por último, tenemos la carpeta "storage", donde encontramos diferentes llamadas para guardar datos en el almacenamiento local del dispositivo. Cabe recalcar que no se trata de una carpeta donde almacenemos datos, sino contiene funciones para almacenar datos en el dispositivo.

6.3.1. Diseño Front End

- El diseño del front end se basa en los requerimientos a desarrollar.
- El color que representa a la aplicación es el azul con código hexadecimal #67C6F0. Este color es el que encontramos en las letras del logo de la aplicación, como podemos

observar en la Fig 6.3.1.1, por lo que se ha considerado reutilizar este color en detalles de las pantallas de esta, estableciendo así un pequeño sello de identidad de la aplicación mediante dicho color.

Fig. 6.3.1.1 Logo Eat & Pay.

- En todas las pantallas relacionadas con realizar alguna acción en algún restaurante se ha establecido una cabecera común para todas estas pantallas. Dicha cabecera está formada por la foto del restaurante, el nombre e información de la pantalla en la que se encuentra el usuario, como podemos observar en la Fig 6.3.1.2

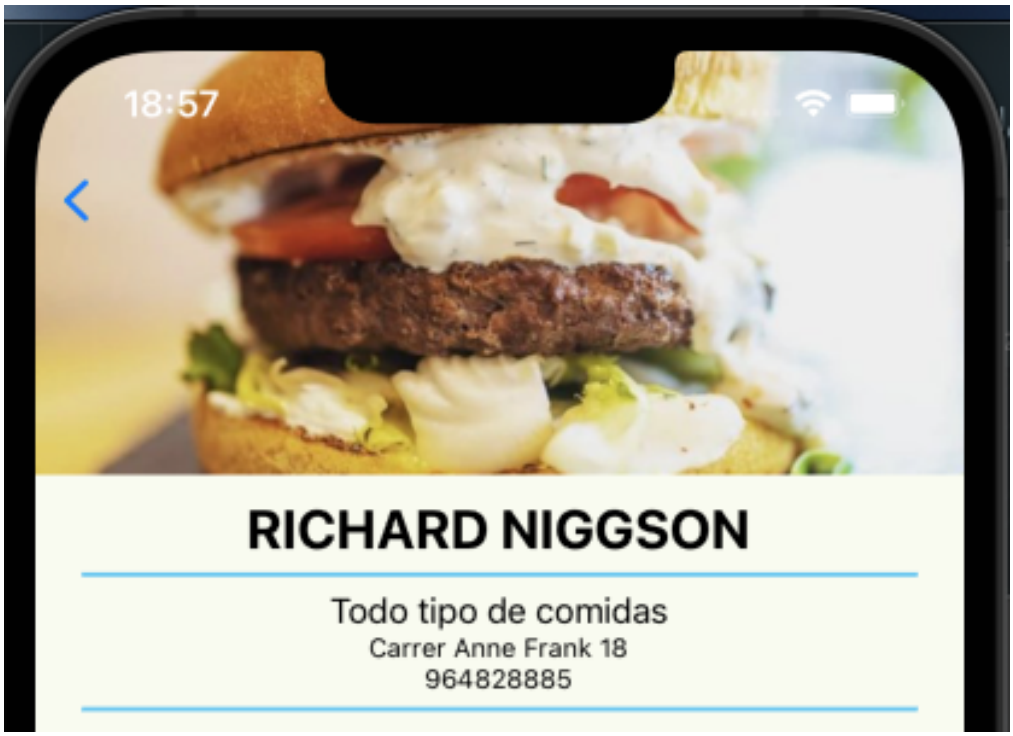


Fig. 6.3.1.2. Cabecera.

- En algunas pantallas encontramos información actualizable, por lo cual tenemos la funcionalidad de deslizar hacia arriba para recargar la página en la que nos encontramos, estas pantallas están indicadas en las siguientes explicaciones.
- En la pantalla principal encontramos una pantalla de inicio de sesión como podemos observar en la Fig 6.3.1.3, que contiene dos opciones, iniciar sesión o crearse una cuenta. Al tratarse de una demo se ha omitido el uso de una contraseña. Dependiendo del usuario que inicie sesión (usuario normal o gestor de restaurante) podremos ver una pantalla de bienvenida u otra.

  En el campo de correo se ha usado el teclado de correo, ejemplo de teclado de correo para IOS en la Fig 6.3.1.33.

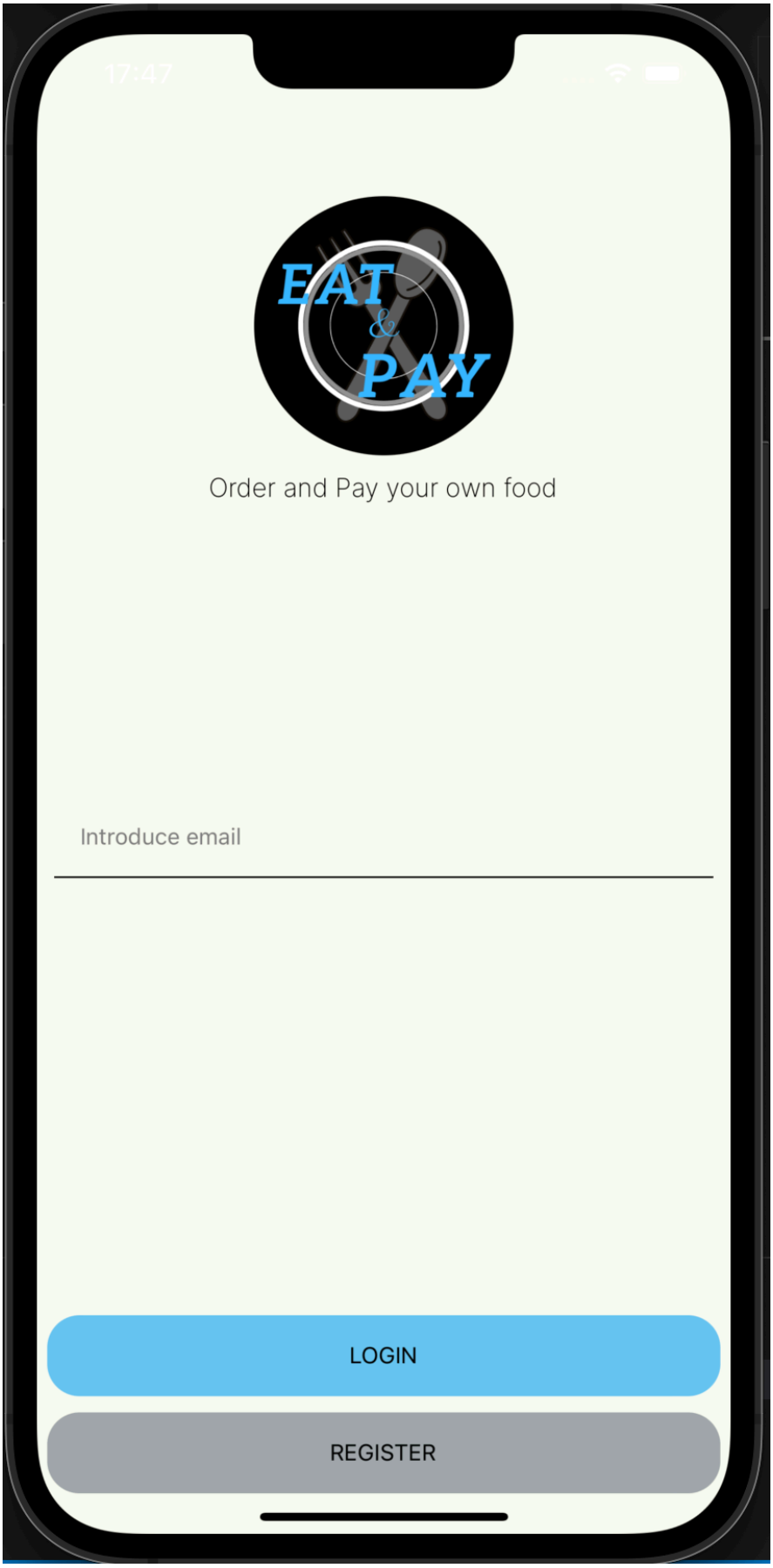


Fig. 6.3.1.3. Pantalla inicio de sesión.

- En la opción de registrarse encontramos un formulario de registro, con el nombre, el correo y el número de teléfono.

  Todos estos campos contienen un texto de ejemplo en el fondo del campo, como podemos observar en la Fig 6.3.1.4.

  En el campo de correo se ha usado el teclado de correo, como podemos observar en la Fig 6.3.1.31 y en el campo de teléfono se ha usado el teclado numérico, como podemos observar en la Fig 6.3.1.30.

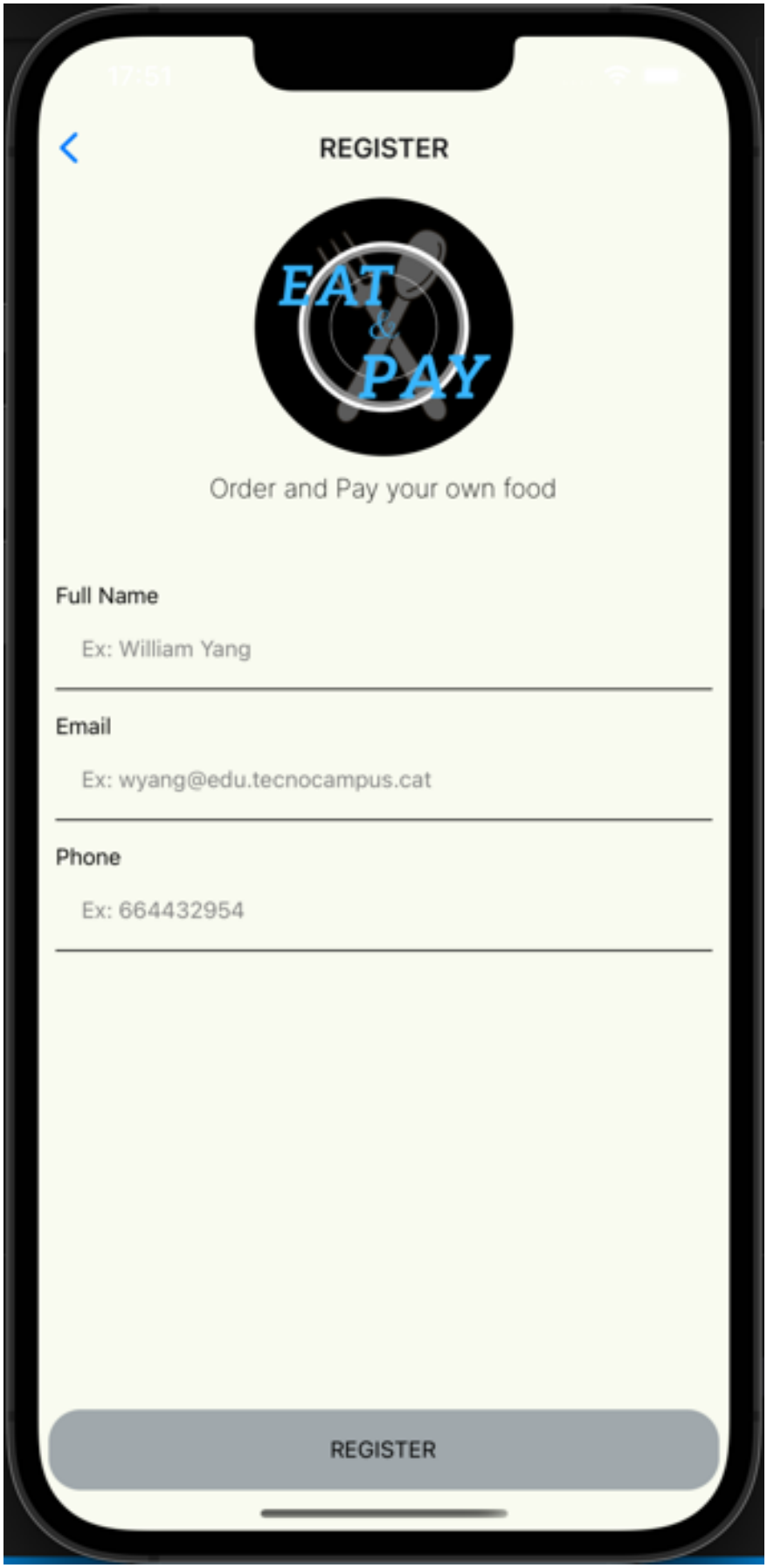


Fig. 6.3.1.4. Pantalla de registro.

- Para el diseño de la pantalla de bienvenida al cliente(usuario normal) nos hemos enfocado en los dos principales usos de la aplicación, visualizar restaurantes y realizar pedidos desde la aplicación. Por ello tenemos un menú navegable inferior de los restaurantes y de la mesa en la que se encuentra sentado el comensal, como podemos observar en la Fig 6.3.1.5.

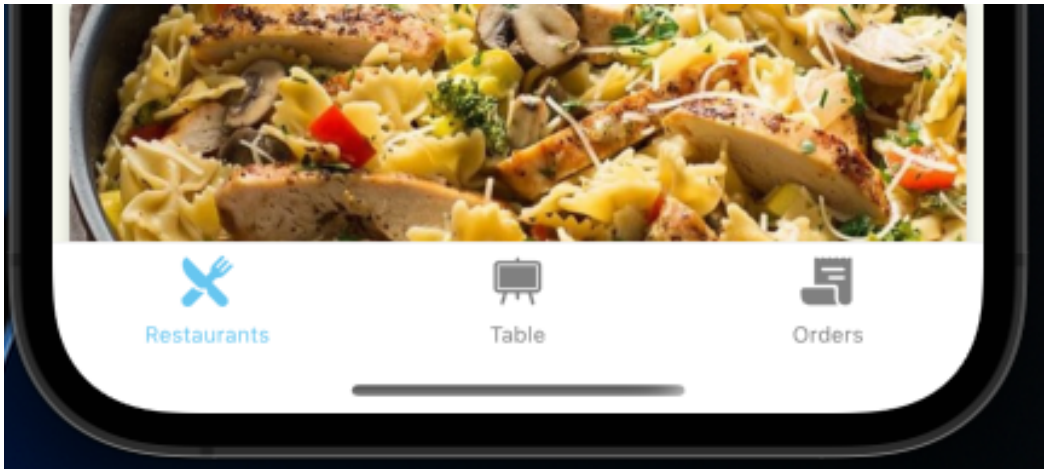


Fig. 5.3.1.5. Menú inferior.

- En la pantalla de restaurantes tenemos un listado que se puede deslizar y clicar de todos los restaurantes que hay disponibles en la aplicación. Cada restaurante está contenido en una tarjeta con la imagen del restaurante, el nombre y la descripción, como podemos observar en la Fig 6.3.1.6.
  Esta pantalla es actualizable.

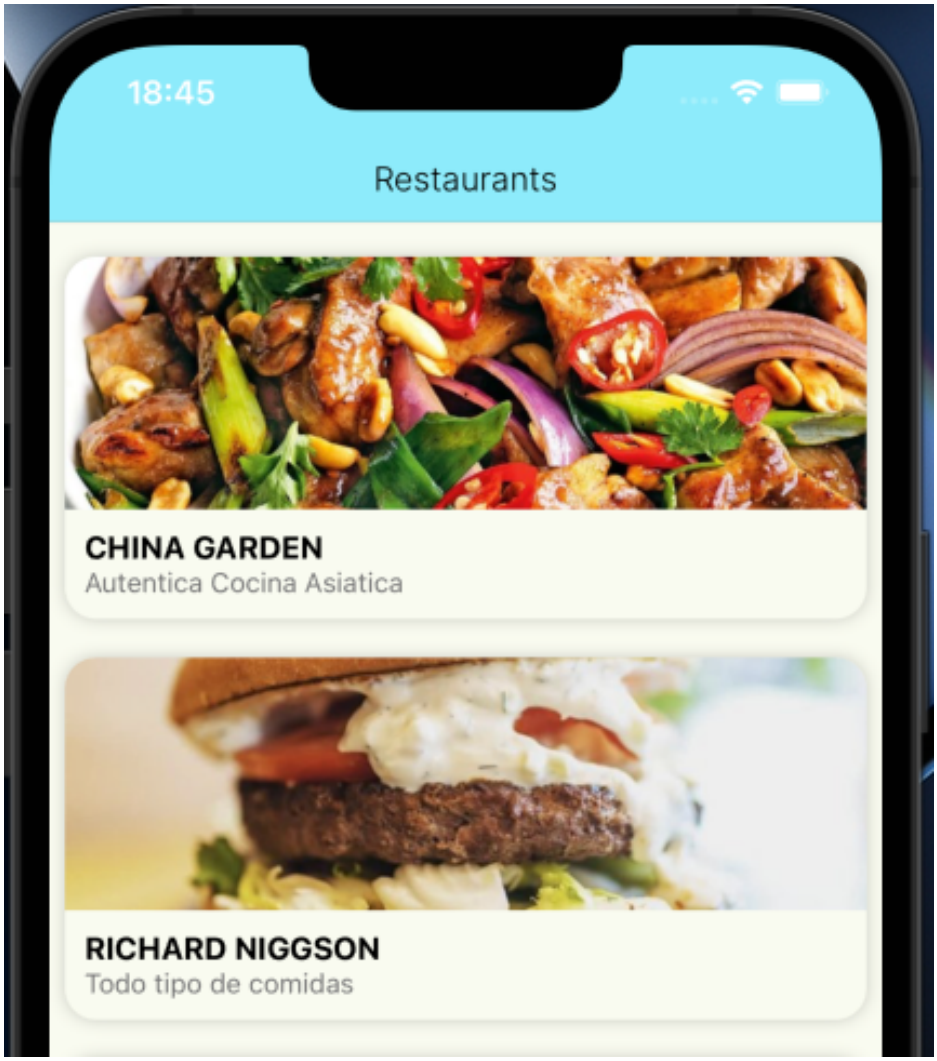


Fig. 6.3.1.6. Pantalla de restaurantes.

- En la pantalla de un restaurante encontramos en la cabecera la imagen del restaurante y la información de esta. En el cuerpo encontramos dos botones, carta y unirse a una mesa, el primero sirve para consultar la carta del restaurante y el segundo sirve para unirse a una mesa de un restaurante. Esta pantalla la podemos observar en la Fig 6.3.1.7.

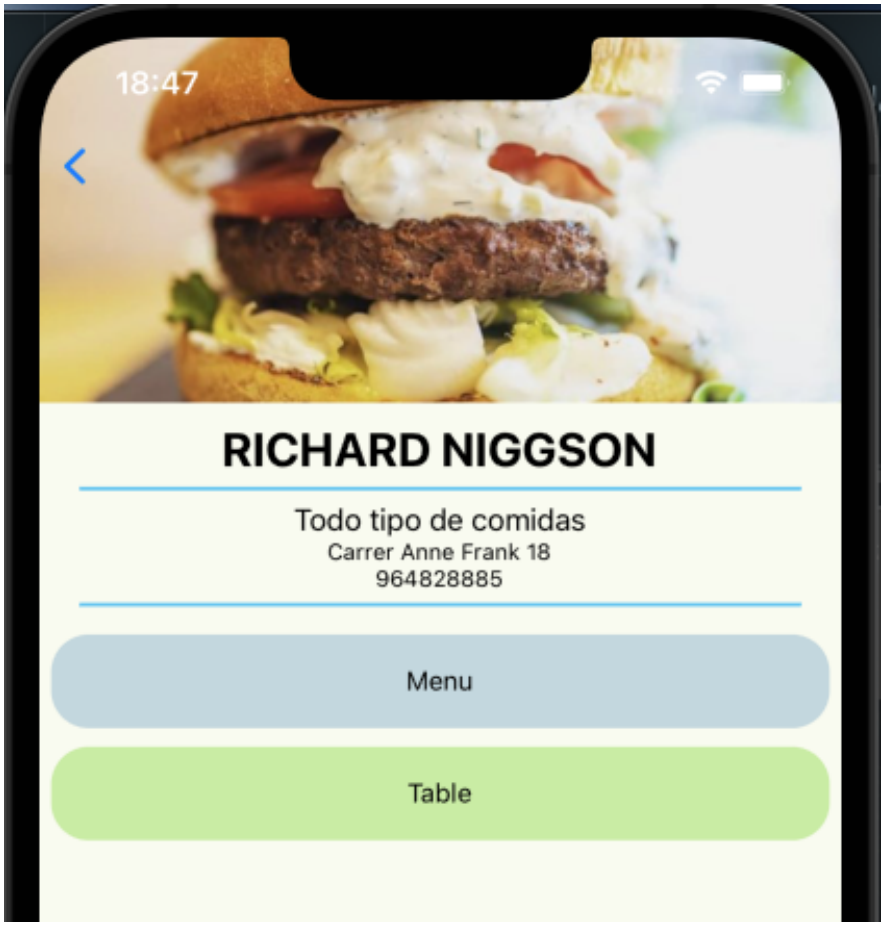


Fig. 5.3.1.7. Pantalla de un restaurante.

- En la pantalla de la carta podemos observar un listado deslizable de todos los platos del restaurante, con el nombre de este, la descripción y el precio, como podemos observar en la Fig 6.3.1.8.

  Esta pantalla es actualizable.

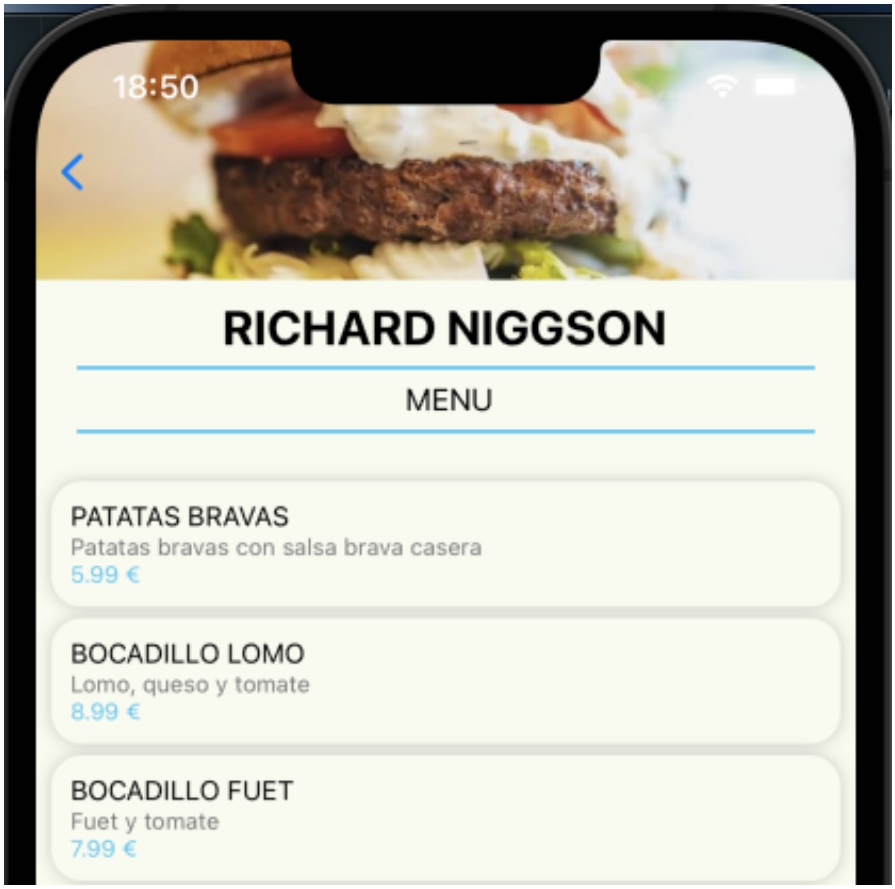


Fig. 6.3.1.8. Pantalla menú de restaurante.

- En la pantalla de unirse a una mesa podemos observar una casilla dedicada para introducir el número de la mesa del restaurante al que queremos unirnos y un botón de unirse, como podemos observar en la Fig 6.3.1.9.

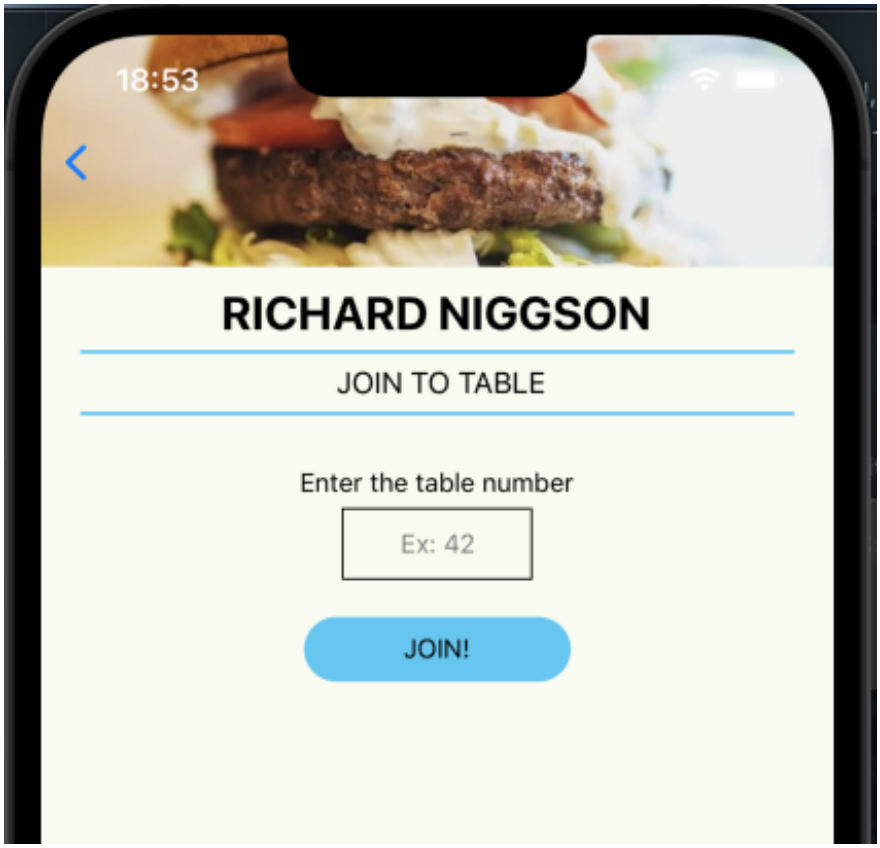


Fig. 6.3.1.9. Pantalla de unirse a una mesa.

- En la pantalla de la mesa en la que se encuentra unido el comensal encontramos diferentes botones, realizar pedido, mis pedidos, pedidos de la mesa y pagar, como podemos observar en la Fig 6.3.1.10.
  Esta pantalla es actualizable.

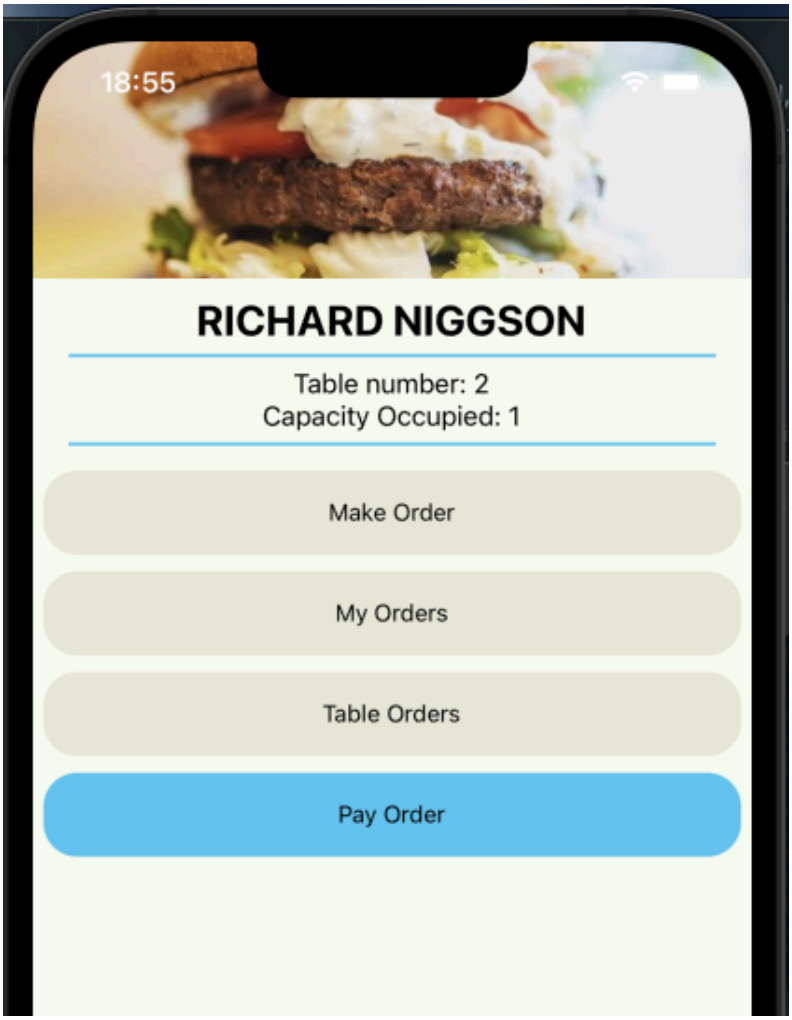


Fig. 6.3.1.10. Pantalla de mesa.

- En la pantalla de realizar pedido encontramos un listado deslizable de todos los productos de la carta del restaurante, cada producto cuenta con un botón de añadir o quitar y un contador de cantidad de ese producto, como podemos observar en la Fig 6.3.1.11. Además, en la cabecera encontramos un icono, que a la vez es un botón, de un carrito con un contador de productos seleccionados, como podemos observar en la Fig 6.3.1.12.

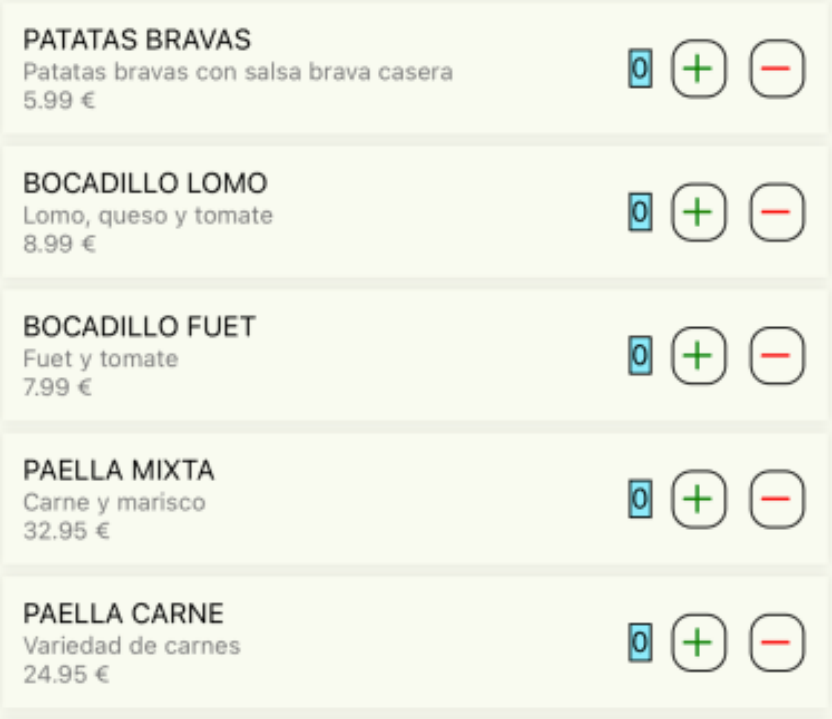


Fig. 6.3.1.11. Listado de productos.

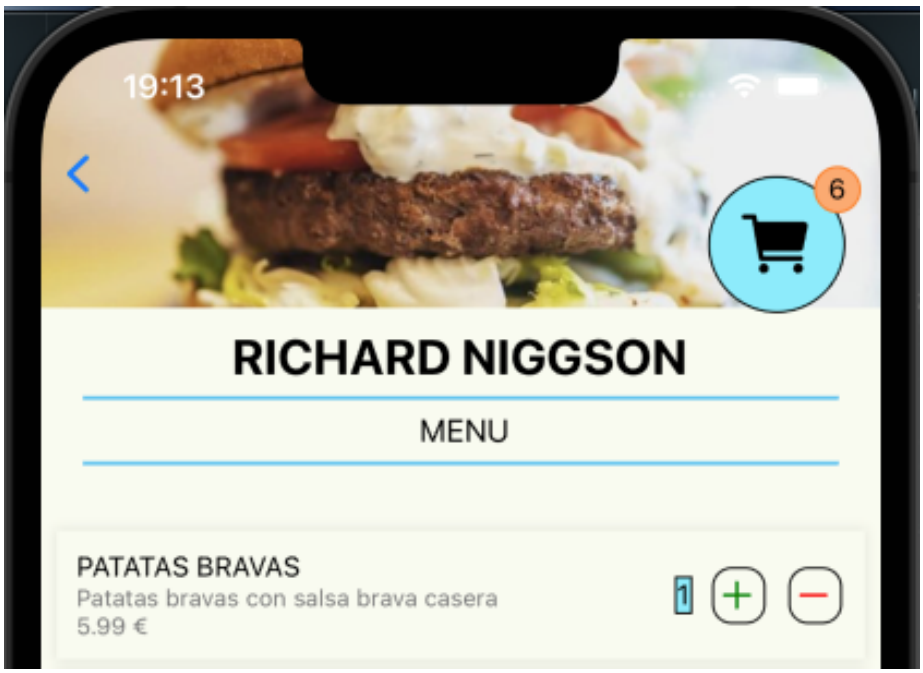


Fig. 6.3.1.11. Cabecera con carrito.

- Una vez pulsado el botón del carrito encontramos un resumen de nuestro pedido con todos los productos y con sus respectivos botones para editar el carrito. También tenemos un total de nuestro pedido y un botón de confirmar la order, como podemos observar en la Fig 6.3.1.12.

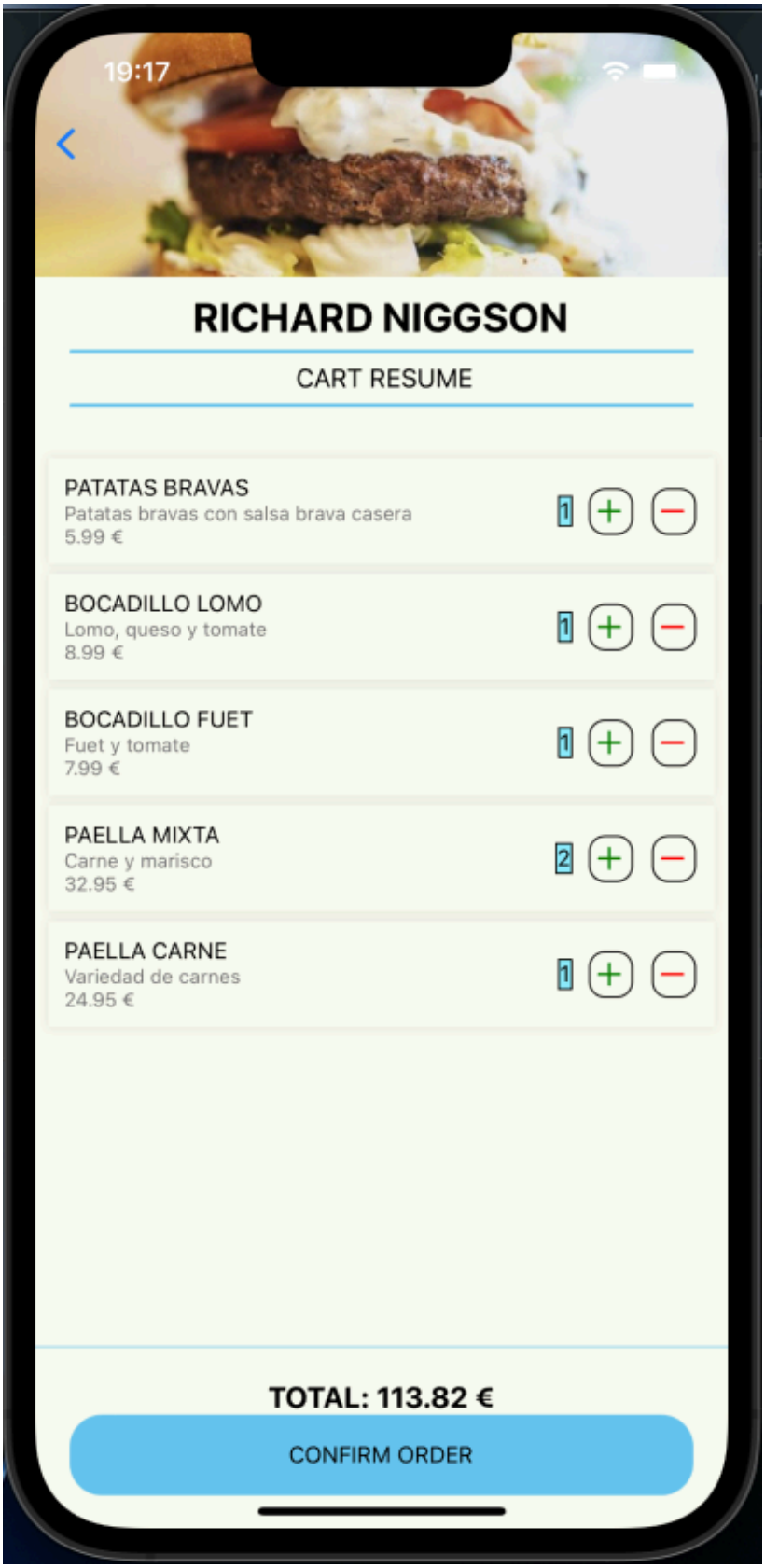


Fig. 6.3.1.12. Resumen de pedido.

- Una vez realizado el pedido se mostrará un mensaje de confirmación al usuario y se navegará de vuelta a la pantalla de mesa, como podemos observar en la Fig 6.3.1.13.

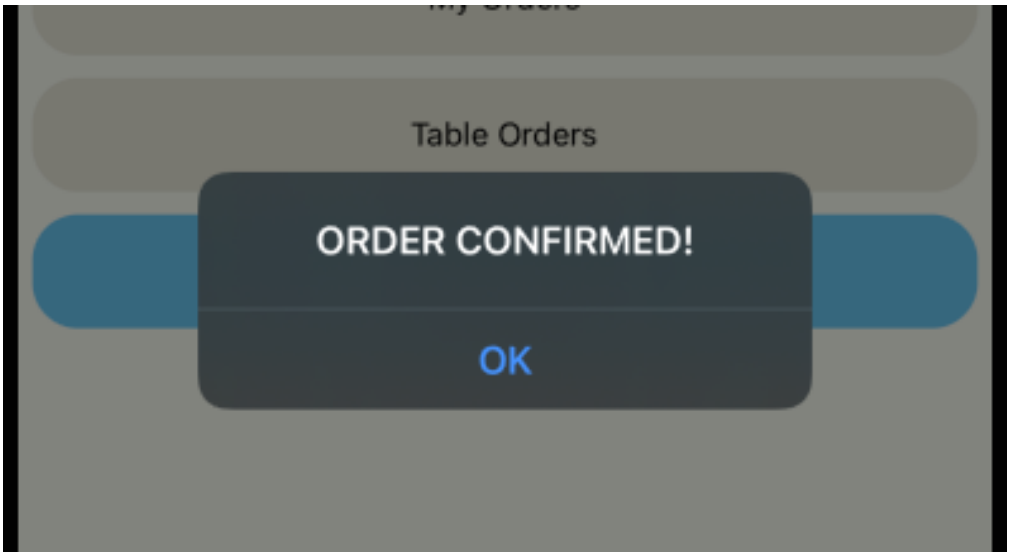


Fig. 6.3.1.13. Confirmación de pedido.

- En la pantalla de mis pedidos encontraremos un listado que se puede deslizar y clicar de todos los pedidos que hemos realizado en esa mesa, con fecha y hora del pedido y el total de este, como podemos observar en la Fig 3.3.1.14.
  Esta pantalla es actualizable.

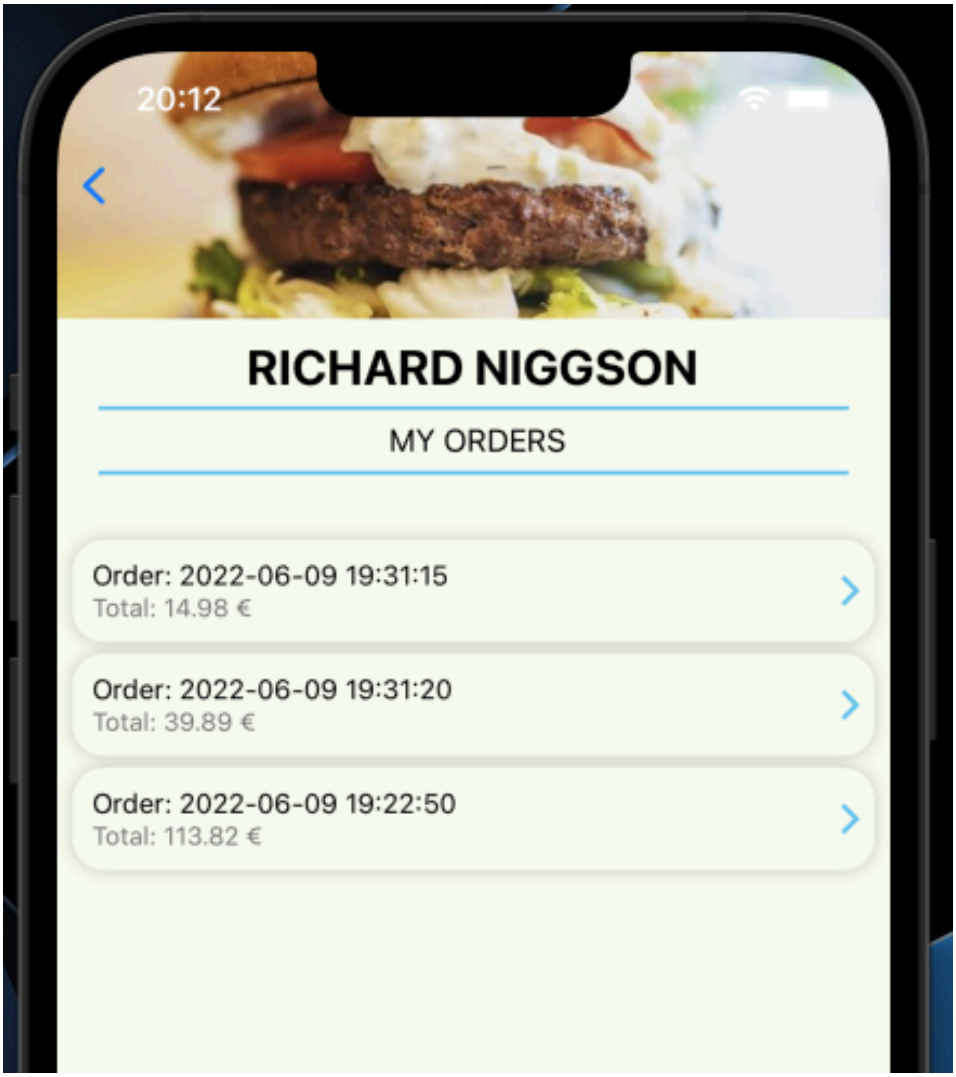


Fig. 6.3.1.14. Listado de mis pedidos.

- En los detalles de un pedido encontramos un listado de todos los productos que hemos pedido, la fecha y hora del pedido, la mesa del pedido, nuestro nombre y el total del pedido, como podemos observar en la Fig .3.1.15.

Fig. 6.3.1.15. Detalles de un pedido.

- En la pantalla de pedidos de la mesa encontramos un listado similar al de mis pedidos, pero de todos los pedidos realizados en dicha mesa y con el nombre del comensal que ha realizado el pedido, como podemos observar en la Fig 6.3.1.16.
  Esta pantalla es actualizable.

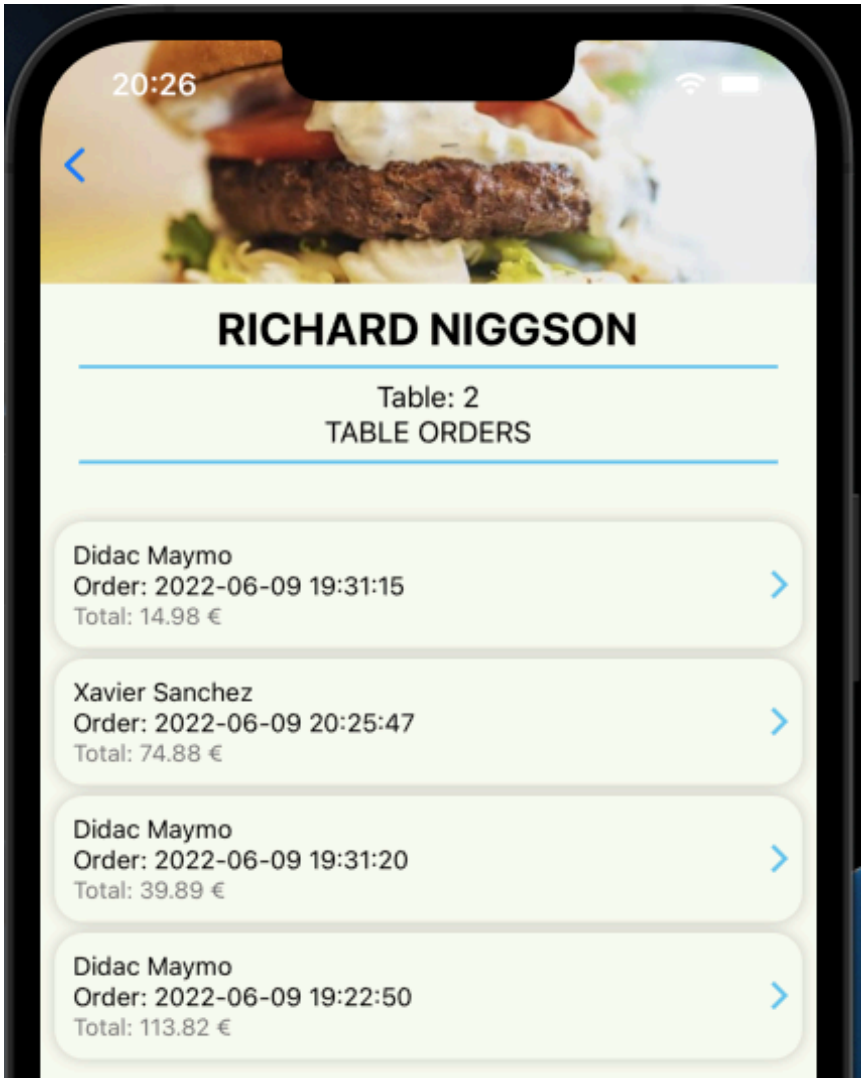


Fig. 6.3.1.16. Listado de pedidos de una mesa.

- En los detalles de los pedidos de una mesa encontramos el mismo listado que en los detalles de mis pedidos, pero con el nombre de quien ha realizado el pedido, como podemos observar en la Fig 6.3.1.17.

Fig. 6.3.1.17. Nombre de comensal.

- La pantalla de pagar pedido está dividida en tres apartados.

El primero es un listado deslizable de todos los platos que hemos pedido, como podemos observar en la Fig 6.3.1.18.

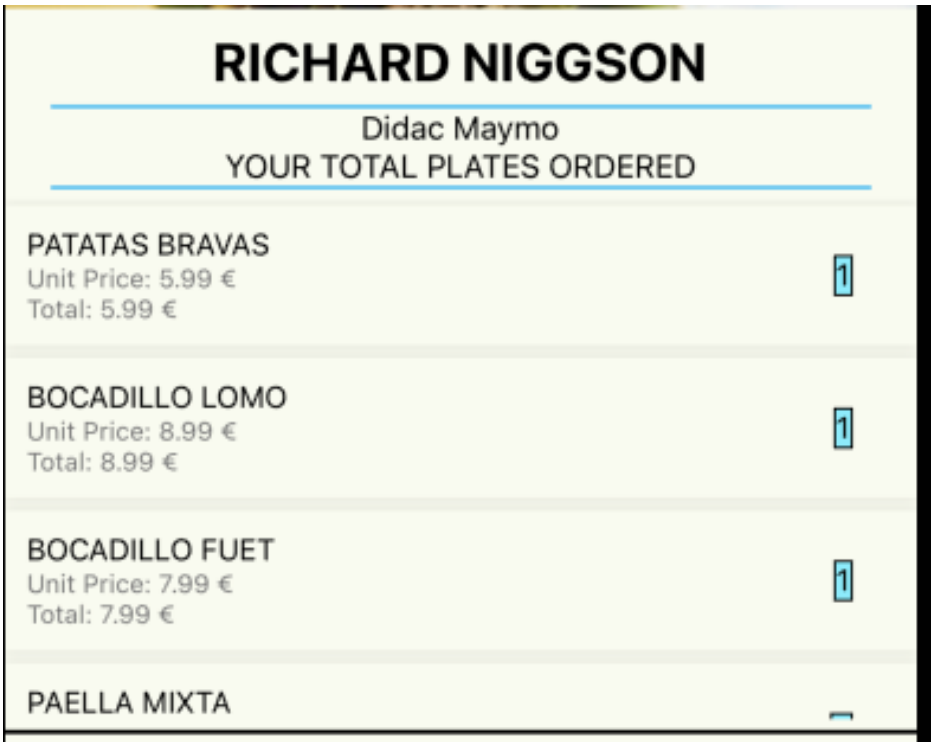


Fig. 6.3.1.18. Listado de platos pedidos.

En segundo apartado contiene todos los platos pedidos en la mesa, cada plato está formado por el nombre del comensal, el nombre del plato, la descripción del plato, el precio por unidad, el precio total, la cantidad pedida de ese plato y un botón de añadir, este botón tiene la funcionalidad de añadir dicho plato a nuestra cuenta, nosotros seremos los pagadores de ese plato pedido por otro comensal, como podemos observar en la Fig 6.3.1.19.

Fig. 6.3.1.19. Listado de platos pedidos en la mesa.

El tercer apartado está formado por los platos que pagaremos de otra persona, está formado como el segundo apartado, pero con un botón de eliminar, este botón realiza la función de devolver el plato a la cuenta del comensal que realizó el pedido de ese plato, como podemos observar en la Fig 6.3.1.20.

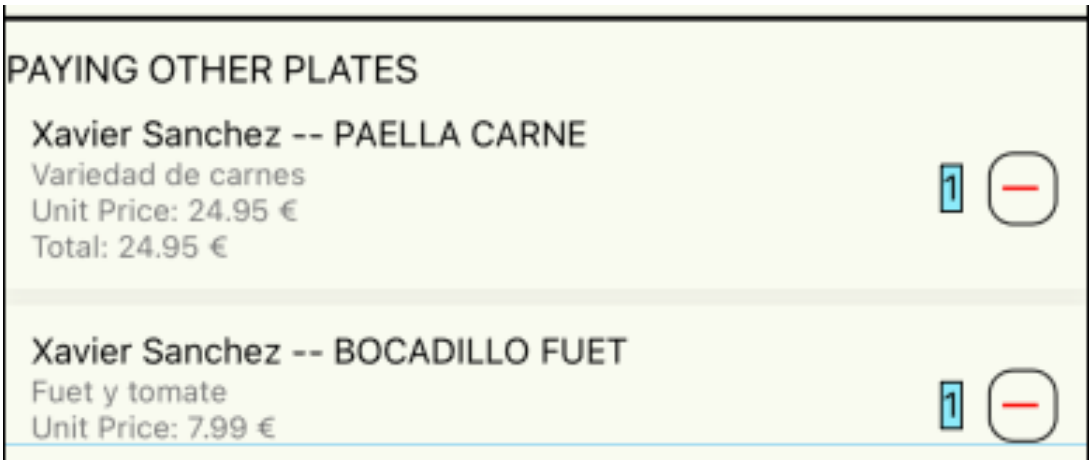


Fig. 6.3.1.20. Listado de platos pedidos a pagar de otros comensales.

Para mantener la consistencia de datos una vez que se pulse el botón de añadir plato de otro comensal, al comensal que le hemos quitado el plato le desaparecerá de su cuenta dicho plato y lo mismo ocurre con la función de devolver el plato al comensal que realizó el pedido, a dicho comensal le volverá a aparecer el plato en su cuenta.

Por último, encontramos el total de nuestra cuenta, sumando los platos a pagar de otra gente, y un botón de pagar cuenta, como podemos observar en la Fig 6.3.1.21.

Esta pantalla es actualizable.

Fig. 6.3.1.21. Total y botón de pagar.

- Una vez realizado el pago nos devuelven a la pantalla inicial de restaurantes y nos muestra un mensaje de confirmación de pago, como podemos observar en la Fig 6.3.1.22.

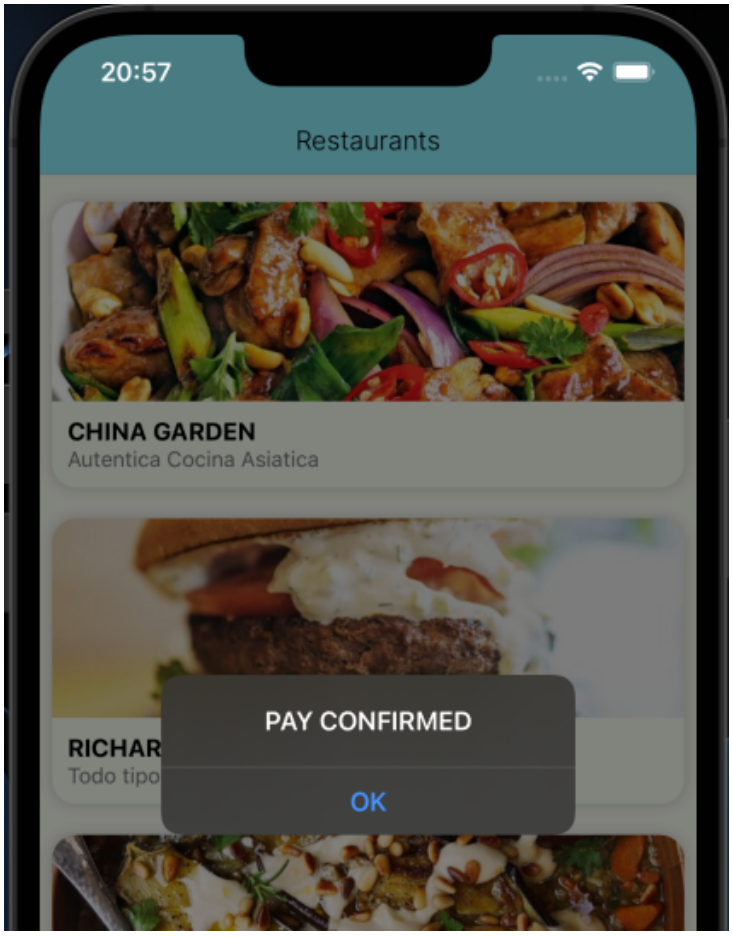


Fig. 6.3.1.22. Confirmación de pago.

- También, una vez realizado el pago se nos desvincula la mesa en la que estábamos sentados, por lo que si volvemos a la pantalla de mesa no estaremos unidos a ninguna mesa, como podemos observar en la Fig 6.3.1.23.

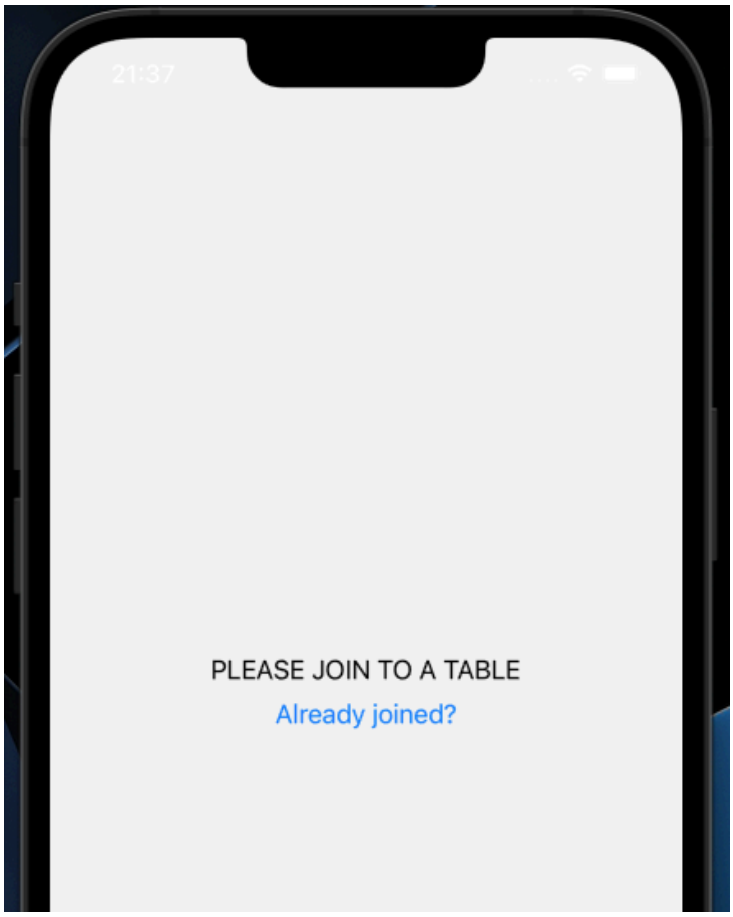


Fig. 6.3.1.23. Pantalla de mesa vacía

- Por último, en el menú inferior encontramos la pantalla de pedidos, aquí encontramos un listado de todo el historial de pedidos que hemos realizado desde nuestra cuenta, como podemos observar en la Fig 6.3.1.24.
  Esta pantalla es actualizable.

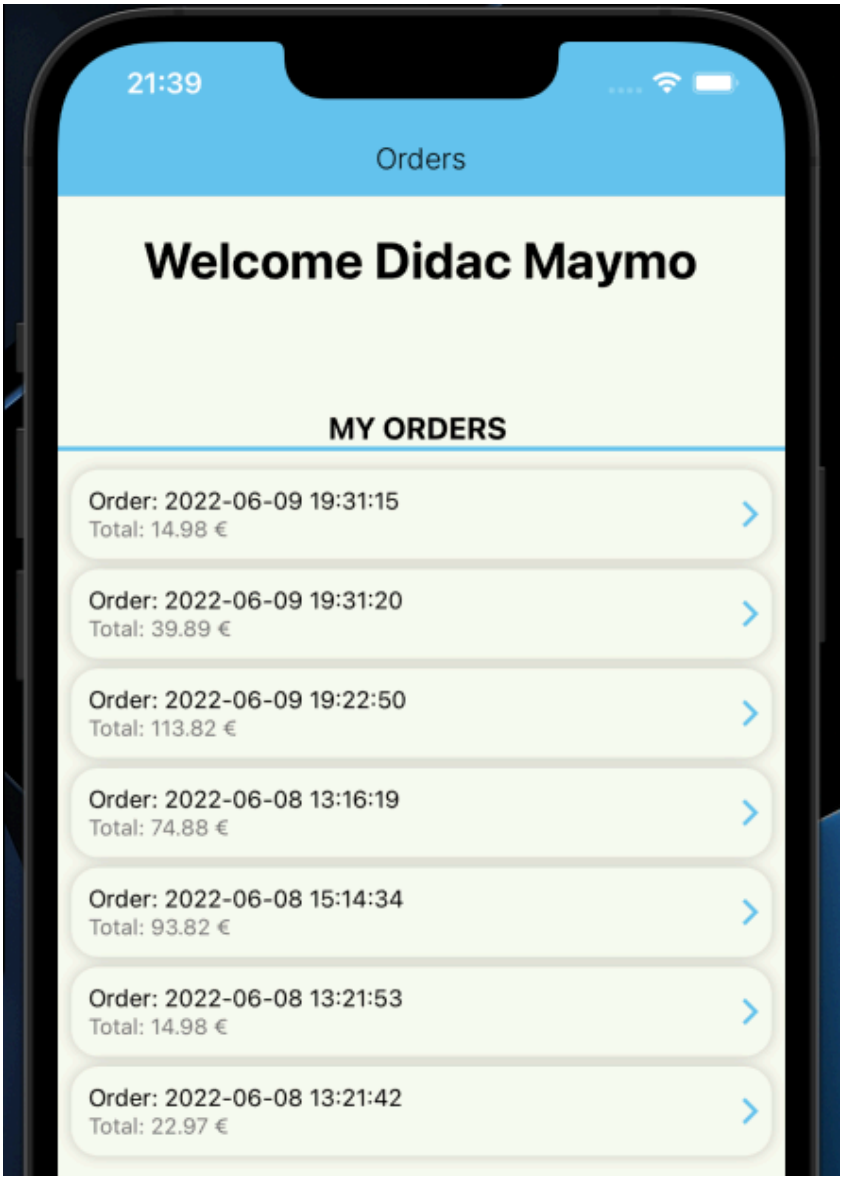


Fig. 6.3.1.24. Pantalla de historial de mis pedidos.

- En la pantalla de detalles de cada pedido encontramos la cabecera del restaurante e información del pedido, como podemos observar en la Fig 6.3.1.25.

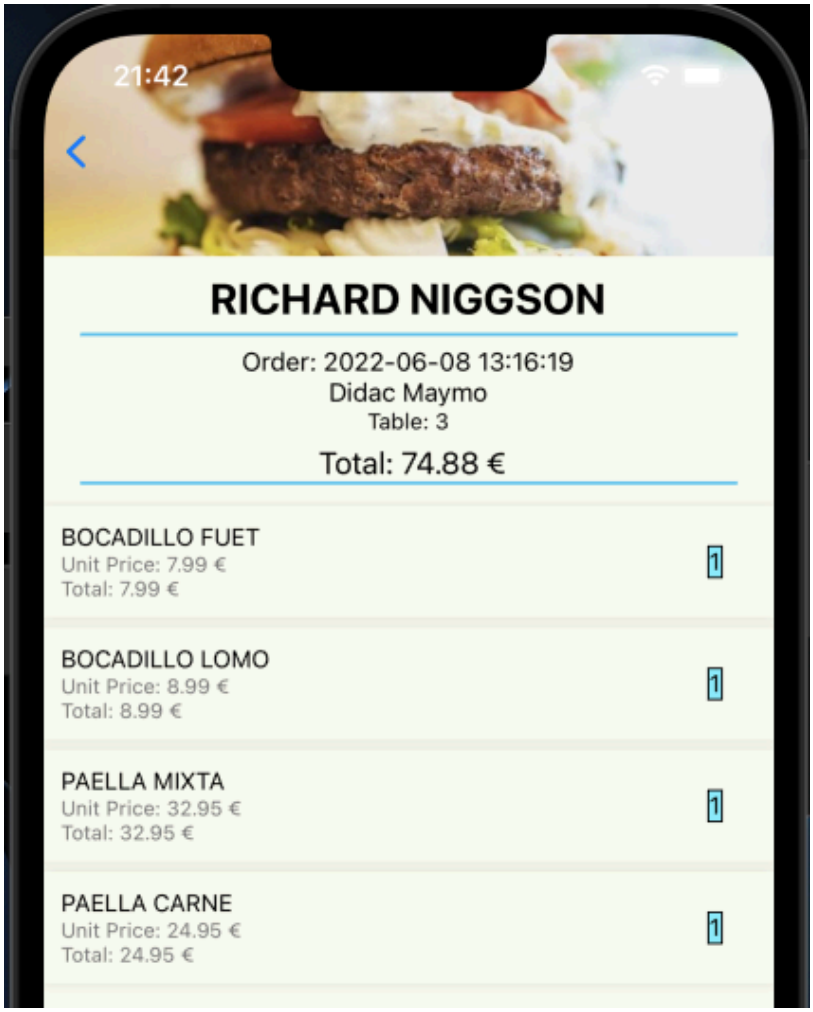


Fig. 6.3.1.25. Pantalla de detalle de pedido.

- La pantalla de bienvenida del usuario gestor de un restaurante está formada por la pantalla principal de nuevos pedidos realizado en ese restaurante, dividido por mesas, como podemos observar en la Fig 6.3.1.26.
  Esta pantalla es actualizable.

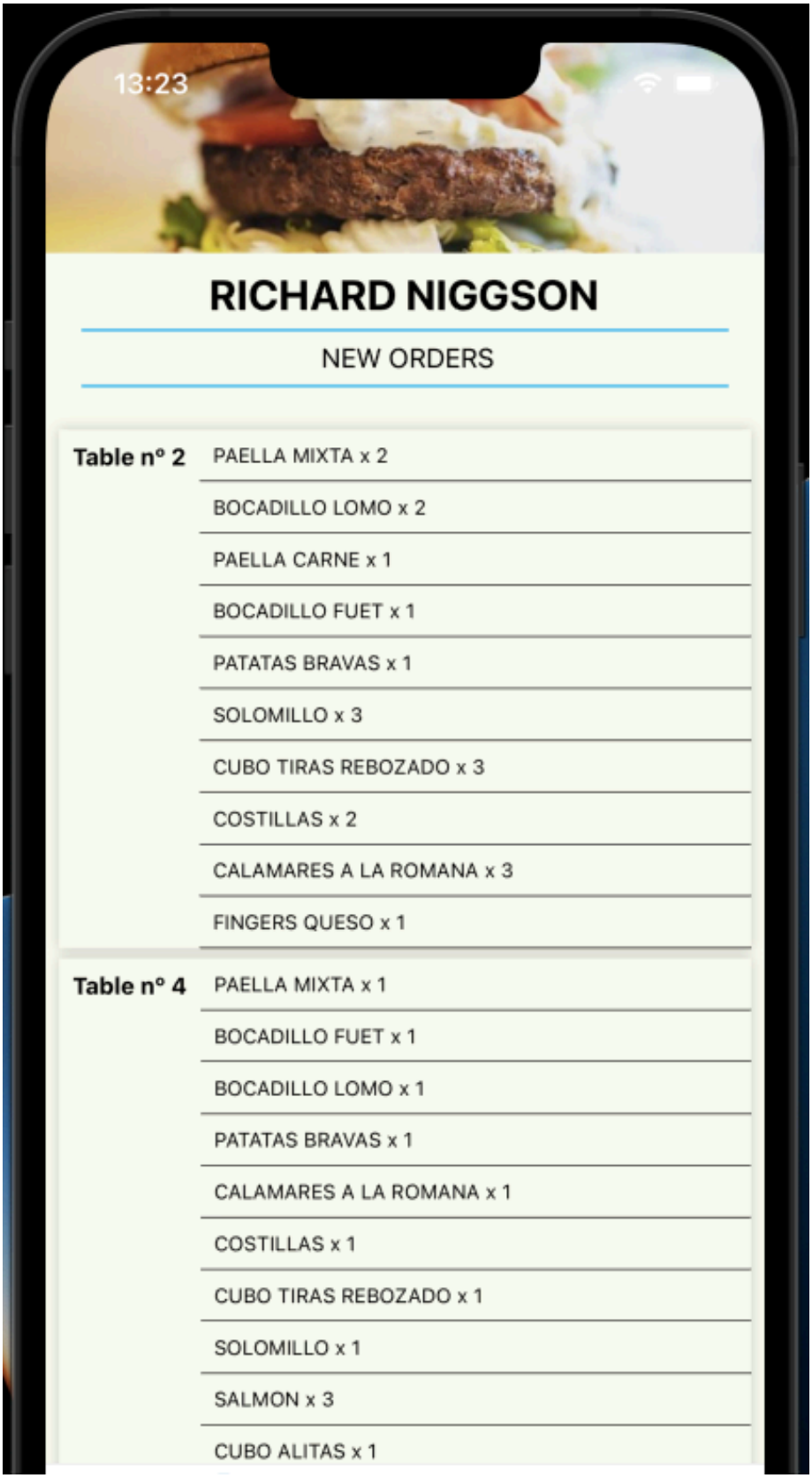


Fig. 6.3.1.26. Pantalla bienvenida de gestor de restaurante.

- También encontramos un menú inferior navegable con la pantalla principal de nuevos pedidos y la pantalla de gestión de carta, como podemos observar en la Fig 6.3.1.27.

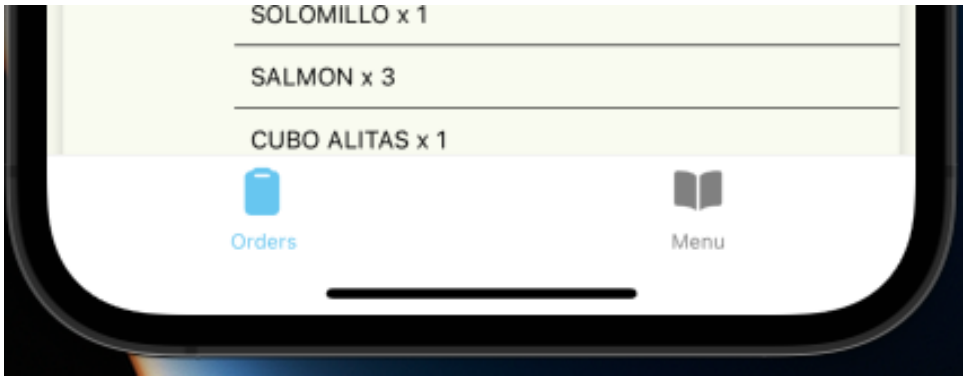


Fig. 6.3.1.27. Menú inferior.

- En la pantalla de gestión de carta encontramos un listado de todos los platos que tiene el restaurante, todos estos platos se pueden pulsar para acceder a la pantalla de edición de plato. También encontramos un botón inferior para añadir platos a la carta, como podemos observar en la Fig 6.3.1.28.
  Esta pantalla es actualizable.

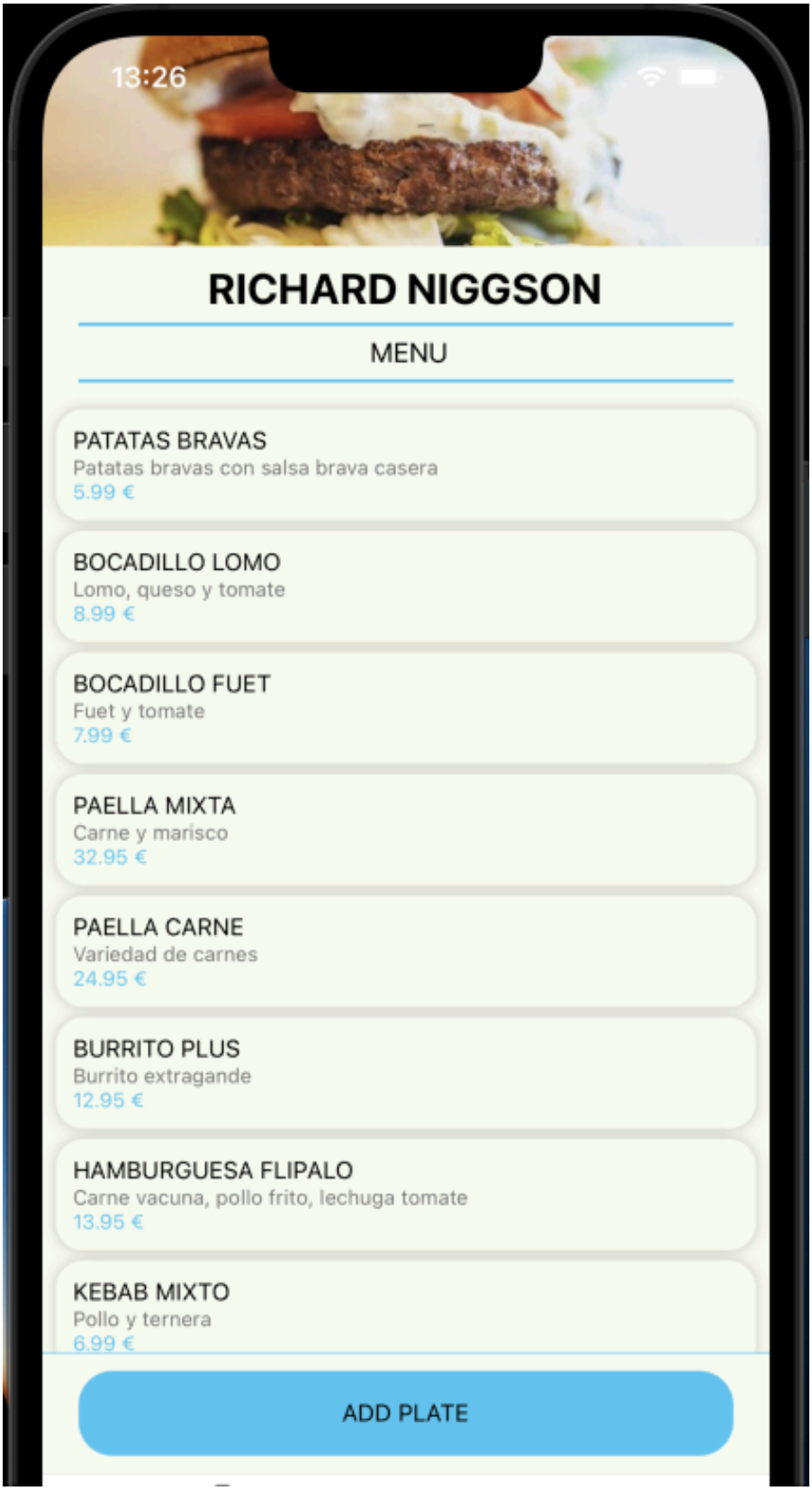


Fig. 6.3.1.28. Gestión de carta.

- En la pantalla de edición de platos encontramos el nombre del plato, la descripción del plato y el precio del plato, todos estos campos son editables, como podemos observar en la Fig 6.3.1.29.

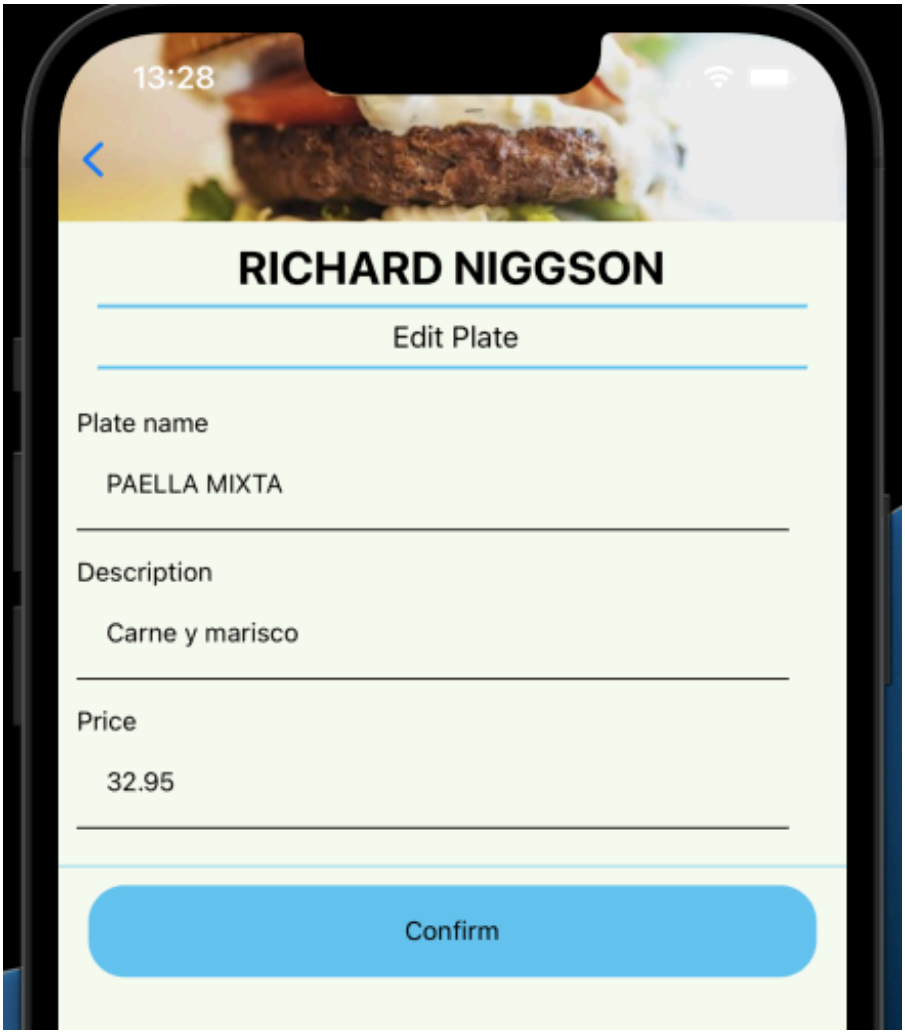


Fig. 6.3.1.29. Editar plato.

En el precio del plato se ha usado un teclado numérico, como podemos observar en la Fig 6.3.1.30

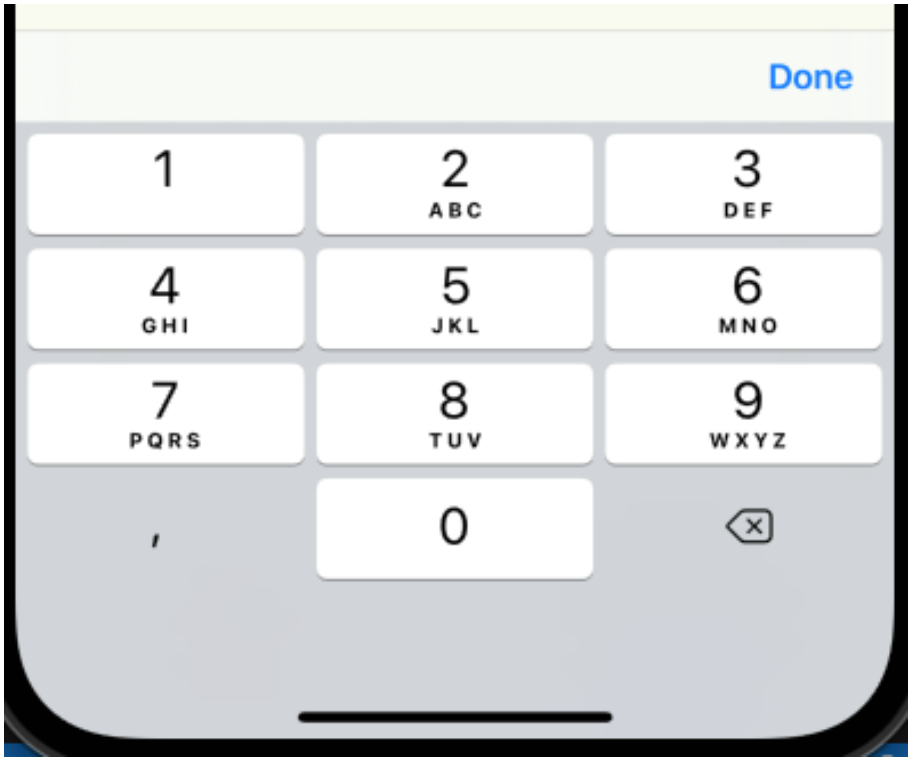


Fig. 6.3.1.30. Teclado numérico.

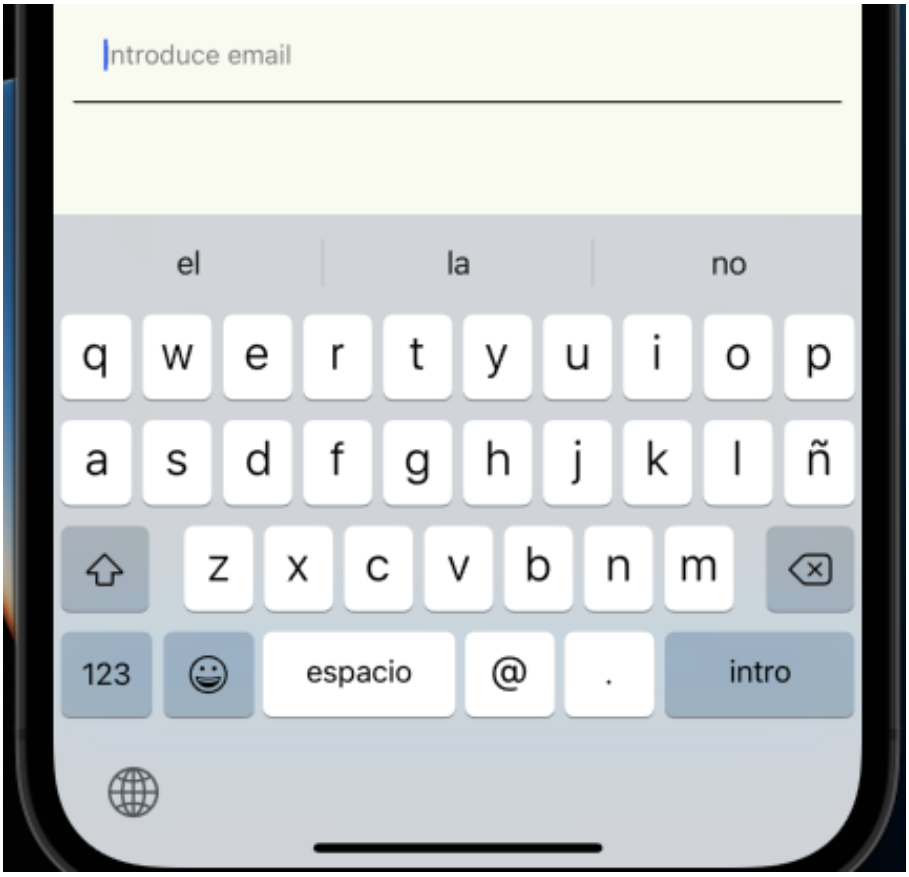


Fig. 6.3.1.31. Teclado correo.

- En la pantalla de añadir plato encontramos tres campos rellenables, nombre de plato, descripción de plato y precio del plato, como podemos observar en la Fig 6.3.1.32.
  En el precio del plato se ha usado un teclado numérico, como podemos observar en la Fig 6.3.1.30.

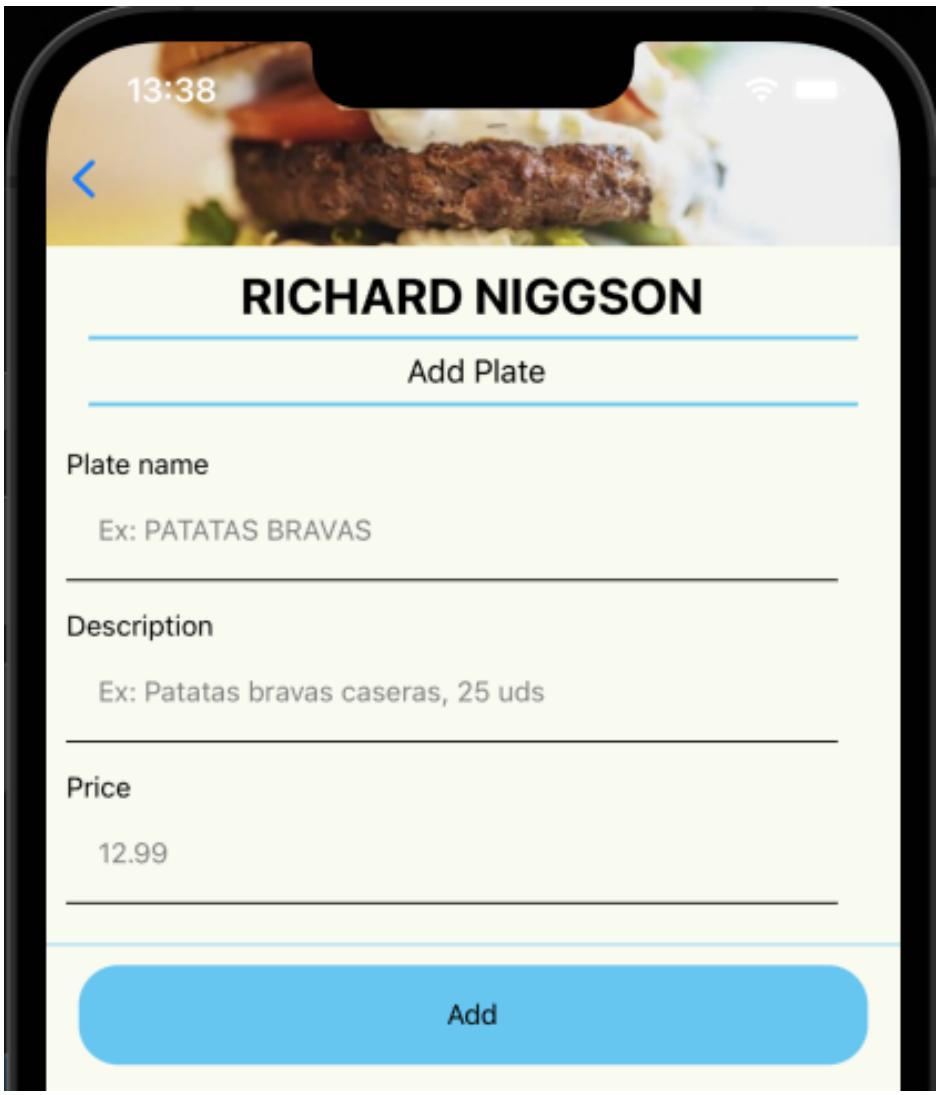


Fig. 6.3.1.32. Añadir plato.

6.4. Tecnologías usadas

En este apartado se van a explicar las tecnologías que se han usado para el desarrollo de la aplicación.

6.4.1. Spring Boot

Spring Boot es una de las principales herramientas del desarrollo de aplicaciones back end con java, que proporciona facilidad en el despliegue de dichas aplicaciones en contenedores web.

El servicio back end se ha desarrollado usando Spring Boot, la principal elección de esta tecnología se debe al conocimiento adquirido sobre esta durante el transcurso de la carrera.

También Spring Boot nos ofrece todas las funcionalidades necesarias para el desarrollo de nuestra aplicación, en concreto desplegar nuestros servicios REST creando nuestras propias API, para así crear, modificar o editar datos de nuestra base de datos.

6.4.2. MySQL

MySQL es un sistema de gestión de base de datos relacional de código abierto gratuito.

Debido a la facilidad de instalación y que permite interactuar con nuestro servicio back end que este software se ha elegido como gestor de base de datos de nuestra aplicación.

6.4.3. Ract Native

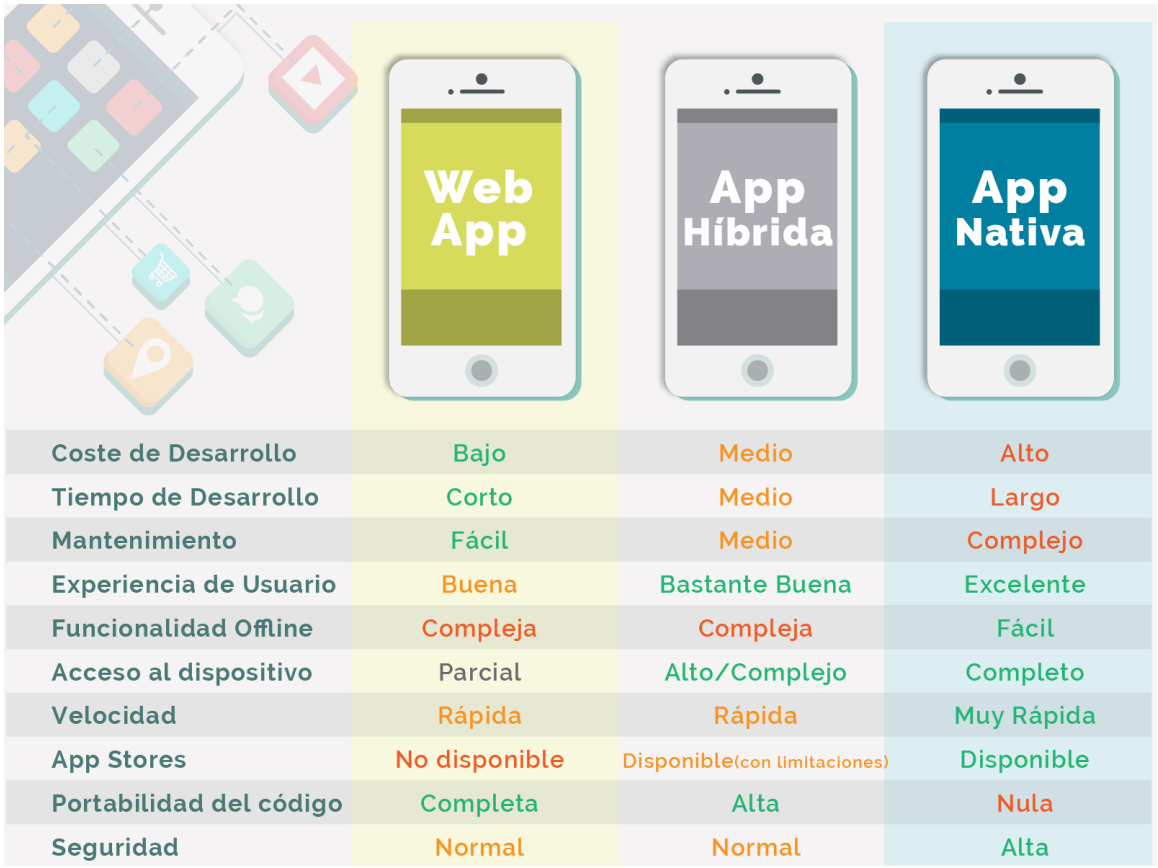
React Native es un framework de JavaScript para crear aplicaciones nativas de Android e IOS usando JavaScript y React. Este framework creado por Facebook nos permite obtener una aplicación nativa para Android y otra para IOS a partir del mismo código base.

Para ello usa componentes de React, son piezas para usarse en la interfaz, son reusables e independientes, por ejemplo un botón personalizado es un componente React.

Debido a la popularidad que ha obtenido en los últimos años y las funcionalidades que ofrece como, multi plataforma, actualización instantánea en desarrollo, aplicación nativa, entre muchas otras, se ha elegido React Native para desarrollar el Front End de nuestra aplicación.

Una aplicación nativa es aquella que se ha desarrollado exclusivamente para un sistema operativo, para así sacar el máximo potencial de un dispositivo. También podemos encontrar otro tipo de aplicaciones para móvil, aplicación web o híbrida, cada cual, con sus ventajas y

desventajas, a continuación, en la Fig 6.4.3.1 podemos observar una tabla comparativa entre estas tres.



	Web App	App Híbrida	App Nativa
Coste de Desarrollo	Bajo	Medio	Alto
Tiempo de Desarrollo	Corto	Medio	Largo
Mantenimiento	Fácil	Medio	Complejo
Experiencia de Usuario	Buena	Bastante Buena	Excelente
Funcionalidad Offline	Compleja	Compleja	Fácil
Acceso al dispositivo	Parcial	Alto/Complejo	Completo
Velocidad	Rápida	Rápida	Muy Rápida
App Stores	No disponible	Disponible(con limitaciones)	Disponible
Portabilidad del código	Completa	Alta	Nula
Seguridad	Normal	Normal	Alta

Fig. 6.4.3.1. Tabla comparativa aplicaciones móvil.

Con React Native bajamos el coste de desarrollo y tiempo de desarrollo de una aplicación Nativa, debido al hecho de no tener que desarrollar por separado en diferentes equipos una aplicación para Android y otra para IOS, esto también conlleva a reducir el tiempo de desarrollo, ya que las dos aplicaciones prácticamente se acabarán de desarrollar al mismo tiempo.

A pesar de todos los beneficios que nos proporciona React Native, en ciertos casos necesitamos depender de desarrolladores de cada plataforma, sobre todo en casos de querer añadir funcionalidades específicas que no se puedan lograr con React Native.

6.5. Despliegue

MySQL se ha instalado y desplegado localmente en el ordenador personal, donde se ha creado la base de datos y las tablas correspondientes.
Dicha base de datos es accesible localmente desde el puerto 3306 y el usuario “root”

El servicio back end se ha desplegado localmente desde el ordenador personal mediante IntelliJId y Spring Boot en el puerto 8080, las peticiones API son accesibles mediante http://localhost:8080 o sustituyendo localhost por la Ip de nuestra máquina.

La aplicación se ha desplegado usando Expo.
Expo es un conjunto de herramientas y servicios que permiten desarrollar, construir e implementar rápidamente aplicaciones IOS y Android. A parte de proporcionar herramientas para el desarrollo de la aplicación también nos permite subir el propio proyecto a nuestra cuenta de Expo y acceder a esta desde cualquier dispositivo a través de la aplicación de Expo, como podemos observar en la Fig 6.5.1.

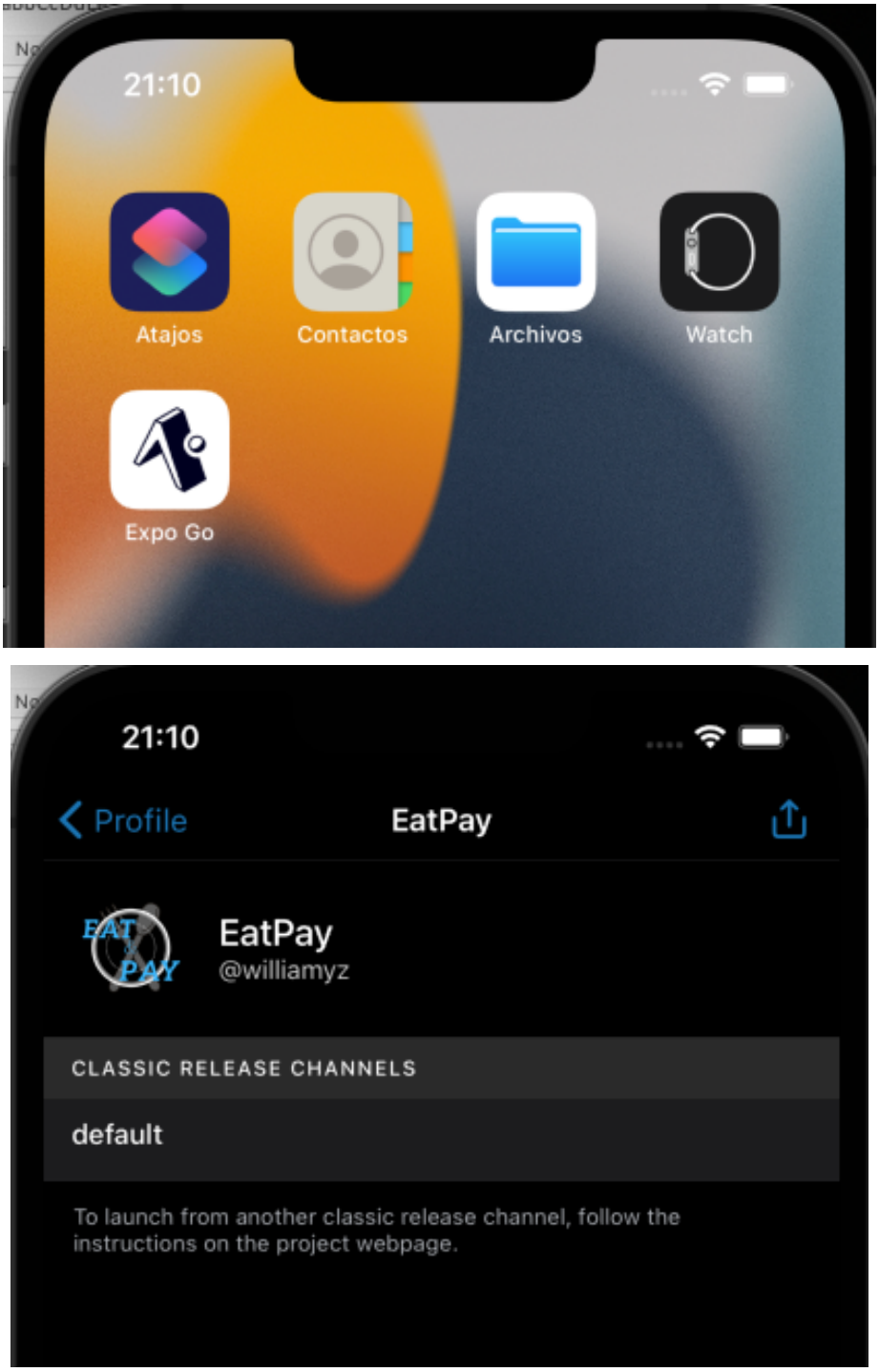


Fig. 6.5.1. Expo

Cabe recalcar que todas las peticiones api realizadas desde el front end tienen como dirección la Ip local de nuestra máquina.

7. Conclusiones

En este proyecto hemos podido observar una solución para el mercado de la hostelería, enfocado en dar una mayor flexibilidad al cliente y una mejora de gestión para el restaurante, donde el cliente puede realizar todas las gestiones desde la propia aplicación, como pagar o realizar el pedido y el propio restaurante puede gestionar los pedidos y su carta desde la propia aplicación.

En este proyecto hemos podido ampliar los conocimientos a partir de la base adquirida a lo largo de la carrera, donde todo lo aprendido ha estado más enfocado al desarrollo en back end.

Gracias a esto hemos conseguido aprender del desarrollo full stack, aprendiendo un nuevo lenguaje para el manejo de front end,, React Native, capaz de desplegarse en los dos sistemas operativos principales del mercado actual.

Con el desarrollo full stack, además hemos aprendido a crear desde cero y desplegar nuestro propio servicio back end con nuestra propia base de datos, también hemos adquirido una base de desarrollo de aplicaciones móvil, consiguiendo conectar la aplicación con nuestro servicio back end y desplegarla para probar en un dispositivo móvil físico.

8. Futuras ampliaciones

- Testing con usuarios reales:
    - Desplegar back end en un servidor, por ejemplo, Heroku
    - Instalar la aplicación en varios dispositivos móviles, a través de Expo.
    - Contactar con algún restaurante para realizar el testing.
    - Seleccionar un grupo de comensales aleatoriamente de ese restaurante.
    - Proporcionar un dispositivo a cada comensal.
    - Se les darán las indicaciones sobre el funcionamiento y la idea de la aplicación a todos los comensales, a partir de aquí realizaran los pedidos desde la aplicación, el restaurante podrá ver los pedidos realizados desde el dispositivo móvil que se facilita al restaurante.
    - A la hora de pagar cada comensal realizará el pago ficticio desde la aplicación, para posteriormente ser cobrados con el precio final del cobro ficticio.

- Ampliar las funcionalidades para el restaurante, para así convertirlo en un nuevo TPV digital para el restaurante:
    - La aplicación ha de permitir al restaurante gestionar los pedidos, aceptarlos, rechazarlos, marcarlos como realizados o entregados.
    - La aplicación ha de permitir al restaurante gestionar una mesa, añadir comensales, quitar comensales, añadir pedidos y quitar pedidos.
    - La aplicación ha de permitir al restaurante realizar pedidos desde la app.
    - La aplicación ha de permitir al restaurante visualizar los pedidos pagados.
    - La aplicación ha de permitir al restaurante aceptar reservas.
    - La aplicación ha de permitir al restaurante gestionar reservas.
    - La aplicación ha de permitir al restaurante recibir pedidos previos de reservas.
    - La aplicación ha de permitir al restaurante recibir pagos.
    - La aplicación ha de permitir al restaurante eliminar platos de la carta.

- Ampliar funcionalidades para el cliente:
    - La aplicación ha de permitir al cliente realizar reservas.
    - La aplicación ha de permitir al cliente realizar pedidos previos para una reserva.
    - La aplicación ha de permitir al cliente pagar toda la mesa.
    - La aplicación ha de permitir al cliente pagar la mesa entre todos.

- La aplicación ha de permitir al cliente seleccionar platos para compartir.
- La aplicación ha de permitir al cliente seleccionar con quien compartir un plato compartido.
- La aplicación ha de permitir al cliente mostrar un seguimiento de los pedidos, si están aceptados o rechazados, en preparación o servidos.
- La aplicación ha de permitir al cliente aceptar o rechazar un plato compartido.
- La aplicación ha de permitir al cliente gestionar una reserva.
- La aplicación ha de permitir al cliente escanear un QR para unirse a una mesa.
- La aplicación ha de permitir al cliente gestionar su cuenta.
- La aplicación ha de permitir al cliente cerrar sesión.
- La aplicación ha de permitir al cliente abandonar una mesa si no ha realizado ningún pedido.
- La aplicación solo ha de permitir al cliente cambiarse de mesa no tiene ningún pedido pendiente de pagar.
- La aplicación ha de permitir al cliente introducir un método de pago.
- La aplicación ha de permitir al cliente acceder a una pasarela de pago.
- La aplicación ha de permitir al cliente realizar un pago con su método de pago seleccionado.

- Proceso de comercialización de la app:
  - La aplicación necesita atraer clientes para que los restaurantes acepten nuestro servicio con comisión por uso, para ello la atracción de clientes se realizará mediante códigos de descuento.
  - La aplicación se promocionará a través de redes sociales y anuncios de directos, como Instagram, YouTube, o Twitch.
  - Los restaurantes iniciales no recibirán ningún cobro por uso, la implantación será totalmente gratuita para el restaurante, ofreceremos un servicio con la promesa de aumentar los clientes de manera gratuita, de esta manera conseguiremos restaurantes para nuestra aplicación.
  - La aplicación se subirá a Google Play y en App Store de manera gratuita.
  - La captación de clientes y de restaurantes supondrá costes sin beneficios, debido a que la atracción de restaurantes y de clientes se necesitan entre ambas, si no hay clientes no hay restaurantes y si no hay restaurantes no hay clientes. Por lo

que ambas son gratuitas y con beneficios para el cliente con códigos descuento ofrecidas por nuestra empresa, sin afectar al restaurante.

- Una vez establecido un margen de clientes considerable se empezará a comercializar la aplicación cobrando por uso, con su correspondiente comisión.

9. Bibliografía

[3] Aplicación paycui.

Disponible a https://paycui.com

[3] El referente artículo sobre paycui.

Disponible a https://elreferente.es/innovadores/paycui-la-app-con-la-que-crear-y-pagar-tu-reserva-en-un-restaurante/

[3] Aplicación sundayapp

Disponible a https://sundayapp.com/es/como-funciona/

[3] Comisiones sobre pedidos a domicilio

Disponible a https://www.banzzu.com/es/pedidos-domicilio-comisiones/

[3] Comisiones glovo

Disponible a https://blog.monouso.es/cuanto-cobra-glovo-a-los-restaurantes/

[3] Sobre Spring Boot

Disponible a https://platzi.com/blog/que-es-spring-boot/?utm_source=google&utm_medium=paid&utm_campaign=17446514363&utm_adgroup=&utm_content=&gclid=CjwKCAjwnZaVBhA6EiwAVVyv9HfCALD9LJwvJd6N-jwQJtJD9pOHAk-3eZ9d1C3i0sHfO0ErXr--LhoCpgkQAvD_BwE&gclsrc=aw.ds

[3] Spring Boot

Disponible a https://spring.io/projects/spring-boot

[3] Que es Spring Boot

Disponible a https://www.arquitecturajava.com/que-es-spring-boot/

[3] Sobre MySQL

Disponible a https://es.wikipedia.org/wiki/MySQL

[3] Diferencias entre SQL y MySQL

Disponible a https://agenciab12.com/noticia/que-son-sql-mysql-diferencia

[3] Que es React Native

Disponible a https://www2.deloitte.com/es/es/pages/technology/articles/que-es-react-native.html

[3] React Native

Disponible a https://reactnative.dev