UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Diseño e implementación de una plataforma de pedidos para restaurantes

Grado de ingeniería informática

Convocatoria de defensa: Junio de 2020

MEMORIA PRESENTADA POR: Carlos Hinojosa Pinto

Tutor: Toni Molina Picó

Resumen

En este proyecto se plantea diseñar y desarrollar una plataforma para facilitar a los clientes de un restaurante, la realización de pedidos desde su mesa y a su vez hacer accesibles los pedidos para el personal del establecimiento. Los clientes podrán acceder a la plataforma web desde su dispositivo móvil e iniciarán sesión escaneando un código QR situado en la mesa. Una vez los clientes hayan accedido a la plataforma, podrán realizar los pedidos, solicitar la atención de un camarero o pedir la cuenta.

El personal del establecimiento podrá ver en todo momento el estado de los pedidos de todas las mesas del local. El personal también podrá conectarse a la plataforma desde su dispositivo móvil o cualquier otro dispositivo con navegador web y conexión a internet. Además de esto, se implementará un apartado con estadísticas e historial de pedidos a partir de los datos de pedidos anteriores. Con esto se podrá llevar la cuenta de todo lo que ocurra en el establecimiento y así mejorar la experiencia final del usuario.

Palabras clave: Web, Cliente-Servidor, Hostelería, Php, Codigo QR

Summary

In this project is proposed to design and develop a platform that helps the clients of a restaurant to realize orders from their table, and also make accessible this orders to the staff of the local. Clients would access to the web platform from their smartphone and login scanning a QR code located in their table. Inside the platform, they could make orders, request a waiter or demand the bill.

The staff would be able to see the state of the orders of all the tables in each moment. They may be able to access the platform from their smartphones or from another device with an internet connection and a web browser. It will be developed a section with statistics and a history of the orders obtained from the data of the previous orders. With this it will be able to keep count of what happens in the local and improve the experience of the final user.

Key words: Web, Client-Server, hostelry, Php, QR code

Listado de figuras

Contenido

1 Introducción

1.1 Propósitos

El propósito de este trabajo es el diseño y la realización de una plataforma web que permita a los clientes de un restaurante realizar pedidos desde sus dispositivos móviles y que los empleados del restaurante tengan acceso a la información para así saber qué platos tienen que servir, a que mesas tienen que llevar la cuenta, que tienen que cocinar en cada momento, etc.

El modo en el que acceden al sistema los clientes es mediante un código que se asigna a la mesa y una clave la cual cambiará cada vez que la mesa deje de estar ocupada. Los clientes tendrán la opción de entrar directamente a la mesa mediante el escaneo de un código QR.

Todo esto con el objetivo de ayudar al restaurante en su servicio y de mejorar la experiencia del consumidor para que así este sea más propenso a volver la restaurante.

1.2 Ámbito del sistema

La plataforma a desarrollar está planteada para que pueda ser utilizada por cualquier restaurante. Como veremos en los siguientes capítulos tenemos una parte que solo verán los empleados en la cual aparecerá todo la información de los pedidos realizados por las diferentes mesas así como la lista de los productos disponibles con la posibilidad de añadir, eliminar o editar cualquiera de estos. La parte pública de la plataforma será a la que accederán los clientes mediante un código y una clave de la mesa en la cual se encuentran.

1.3 Visión general del documento

En este primer capítulo de "**Introducción"** hemos visto las funcionalidades generales del proyecto.

En el siguiente capítulo de "**Estado del arte"** veremos cómo está el sector de la restauración en la actualidad y las oportunidades que hay para entrar en el mercado.

En el capítulo titulado "**Diseño general del proyecto**" vamos a analizar que necesitamos para la realización de nuestro proyecto así como el diseño de las partes más importantes, como son la base de datos o la API a implementar por parte del servidor.

Y en el último capítulo titulado "**Implementación**" vamos a ver más en profundidad el desarrollo del proyecto, estudiando tanto para el backend como para el frontend las tecnologías utilizadas en el desarrollo. Por último veremos cuál ha sido el resultado de la implementación.

2 Estado del arte

El sector de la restauración es uno de los más importantes en nuestro país, en 2018 facturó 94.000 millones de euros, lo que equivale a un 4,7% del PIB de ese mismo año. Y da empleo a más de 1.3 millones de trabajadores en más de 250.000 locales (datos de hosteltur.com).

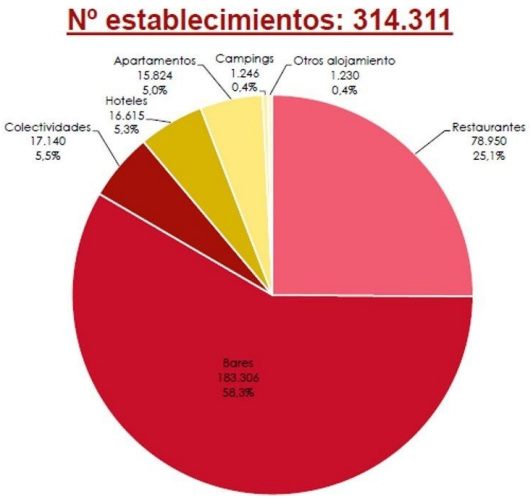

Figura 1.1 Establecimientos de hostelería 2018 en España

Dentro de este sector, una parte importante es la comida a domicilio, esta es cada vez más popular, facturando 4.035 millones de euros en 2019, un 4,7% más que el año anterior (datos de foodretail.es). Como consecuencia de esto han aparecido muchas plataformas de para realizar pedidos a domicilio, algunas de las más populares son Just Eat, Glovo o Deliveroo. Esto ha hecho que cada vez haya más personas que piden comida desde sus dispositivos móviles para que se la lleven a casa, el 36% de los españoles entre 18 y 69 años los hacen.

A pesar de esta mejora tecnológica que ha experimentado el reparto a domicilio es muy difícil encontrar un restaurante en el que puedas pedir desde tu dispositivo móvil para que te entreguen la comida a tu mesa. Existen diferentes propuestas para hacerlo, como son Maître Serie 4 Menú de la empresa Astarté Informática S.L o la plataforma Entrecartas. Pero ninguna de ellas tiene de momento la popularidad de las aplicaciones de pedidos a domicilio mencionadas anteriormente.

Las plataformas mencionadas tienen un modelo de Marketplace, es decir, en su plataforma se encuentran todos los restaurantes que hayan contratado el servicio. A diferencia de esto, en nuestro proyecto se plantea una plataforma personalizada para el restaurante. La cual se adapte a las necesidades del cliente y presente los diseños y estilos del establecimiento.

3 Diseño general del proyecto

3.1 Análisis de requisitos

A continuación vamos a describir los requerimientos de la plataforma. Los vamos a dividir en dos grupos, por un lado los requerimientos relacionados con la parte visible por los empleados, y por otro lado los de la parte visible por los clientes. En el capítulo **2.2 Casos de uso** veremos ampliados estos conceptos.

3.1.1 Empleados

Como hemos visto anteriormente, vamos a detallar los requerimientos necesarios para la parte de administración.

1 Iniciar sesión: Los usuarios podrán iniciar sesión en la plataforma de administración con el uso de unas credenciales previamente almacenadas en una base de datos.

2 Visualizar estadísticas: Los usuarios podrán visualizar estadísticas relacionadas con los pedidos del restaurante.

3 Administrar productos: Los usuarios podrán visualizar, editar, crear y eliminar productos del menú.

4 Administrar mesas: Los usuarios podrán visualizar, editar, crear y eliminar las mesas del restaurante.

5 Administrar usuarios: Los usuarios podrán visualizar, editar, crear y eliminar los usuarios de la plataforma de administración.

6 Administrar pedidos: Los usuarios podrán visualizar, editar y eliminar los pedidos realizados por los clientes.

3.1.2 Clientes

Como hemos visto anteriormente, vamos a detallar los requerimientos necesarios para la parte de administración.

1 Iniciar sesión: El cliente podrá iniciar sesión en la plataforma mediante una clave y un código asignado a una mesa.

2 Visualizar menú: El cliente podrá ver la carta con todos los platos disponibles para pedir.

3 Administrar pedidos: El cliente podrá realizar pedidos para que el camarero los traiga directamente a la mesa, en el caso de que el pedido no haya pasado a la cocina este podrá cancelarlo.

4 Solicitar camarero: El cliente podrá solicitar que un camarero se acerque a la mesa para realizar consultas.

5 Visualizar cuenta: El cliente podrá ver en todo momento la cuenta asociada a su mesa con los precios asignados.

6 Solicitar la cuenta: El cliente podrá solicitar desde la plataforma que le traigan la cuenta, tanto para pagar en efectivo o con tarjeta.

3.2 Casos de uso

Al igual que en el apartado anterior vamos a dividir los casos de uso en dos partes, por un lado la de los empleados y por el otro la de los clientes.

3.2.1 Empleados

A continuación vamos a ver un diagrama en el cual se resume los casos de uso relacionados con la administración de la plataforma.

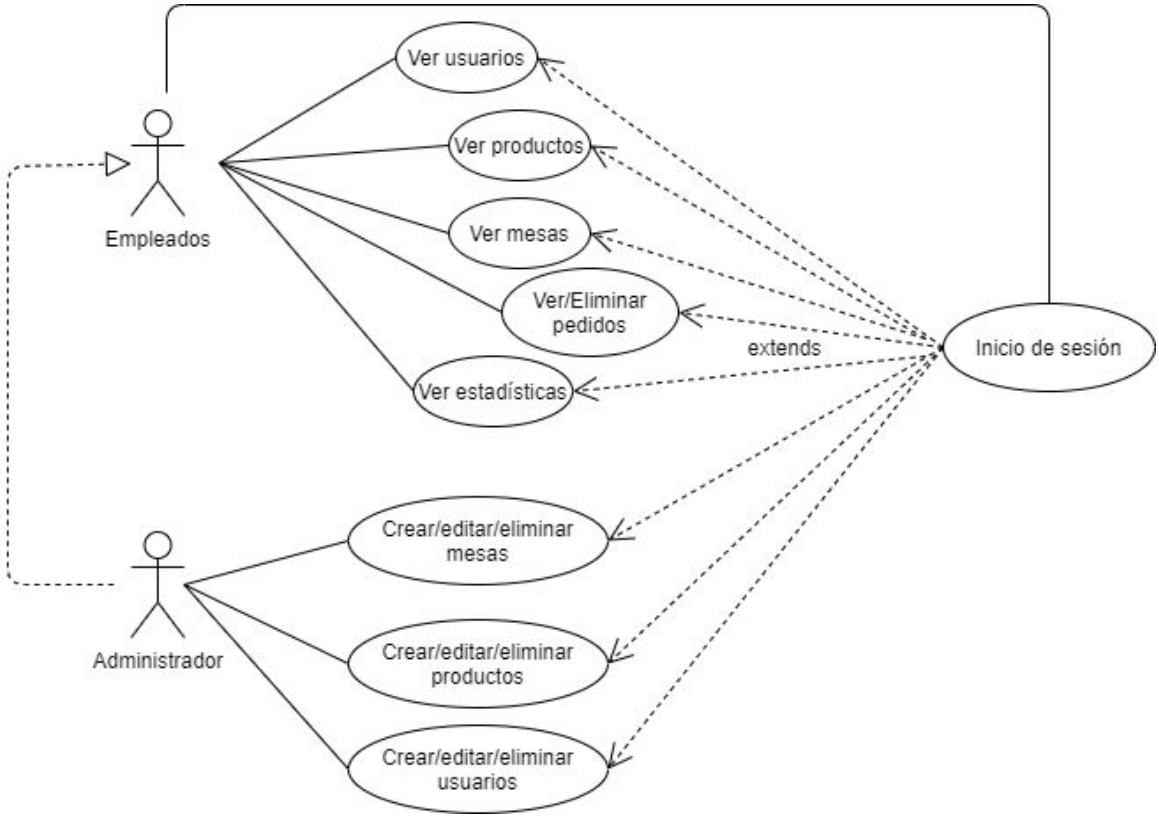

Figura 3.1 Casos de uso de empleados

Como podemos observar a parte de los empleados normales encontramos a los administradores, los cuales manipular los datos de las mesas, productos y usuarios de la plataforma. Por otro lado vemos que para realizar cualquiera de las acciones señaladas

hay que primero iniciar sesión en la plataforma. A continuación vamos a ver con detalle cada caso de uso.

Inicio de sesión

Evento: El usuario introduce un nombre de usuario y una contraseña en el formulario de inicio de sesión.
Escenario principal de éxito : El usuario accede a la plataforma, donde aparecerán diferentes acciones a realizar.
Extensiones: En el caso de que las credenciales no sean correctas se mostrará un error y se volverá al formulario de inicio.

Ver usuarios

Evento: El usuario selecciona el apartado usuarios en el menú principal de la plataforma de administradores
Precondiciones: El usuario debe de haber iniciado sesión.
Escenario principal de éxito : Se muestra un listado de los usuarios registrados.

Crear usuarios

Evento: El usuario rellena los datos y pulsa el botón añadir usuario
Precondiciones: El usuario debe de haber iniciado sesión con un usuario de tipo administrador.
Escenario principal de éxito : Se crea un usuario.
Extensiones: En el caso de no ser un usuario de tipo administrador no se podrá crear y se mostrará un mensaje de error.

Editar usuarios

Evento: El usuario rellena los datos para editar su usuario.
Precondiciones: El usuario debe de haber iniciado sesión.
Escenario principal de éxito : Se modifican los datos del usuario.

Eliminar usuarios

Evento: El usuario pulsa el botón de eliminar dentro de un usuario registrado.
Precondiciones: El usuario debe de haber iniciado sesión con un usuario de tipo administrador.
Escenario principal de éxito : Se elimina el usuario de la base de datos.

Ver productos

Evento: El usuario selecciona el apartado productos en el menú principal de la plataforma de administradores.
Precondiciones: El usuario debe de haber iniciado sesión.
Escenario principal de éxito : Se muestran la lista con los productos.

Crear productos

Evento: El usuario rellena los datos del nuevo producto y pulsa el botón de añadir
Precondiciones: El usuario debe de haber iniciado sesión con un usuario de tipo administrador.
Escenario principal de éxito: Se añade un producto a la base de datos.

Editar productos

Evento: El usuario modifica algún dato del producto seleccionado.
Precondiciones: El usuario debe de haber iniciado sesión con un usuario de tipo administrador.
Escenario principal de éxito : Se edita el producto a la base de datos.

Eliminar productos

Evento: El usuario pulsa el botón de eliminar sobre el producto.
Precondiciones: El usuario debe de haber iniciado sesión con un usuario de tipo administrador.
Escenario principal de éxito : Se elimina el producto de la base de datos.

Ver mesas

Evento: El usuario selecciona el apartado mesas en el menú principal de la plataforma de administradores
Precondiciones: El usuario debe de haber iniciado sesión.
Escenario principal de éxito : Se muestran la lista con las mesas.

Crear mesas

Evento: El usuario introduce la información de la nueva mesa y pulsa el botón de añadir.
Precondiciones: El usuario debe de haber iniciado sesión con un usuario de tipo administrador.
Escenario principal de éxito : Se crea una nueva mesa en la base de datos.

Editar mesas

Evento: El usuario modifica algún dato de la mesa seleccionada.
Precondiciones: El usuario debe de haber iniciado sesión con un usuario de tipo administrador.
Escenario principal de éxito : Se modifica la mesa de la base de datos.

Eliminar mesas

Evento: El usuario pulsa el botón de eliminar sobre la mesa.
Precondiciones: El usuario debe de haber iniciado sesión con un usuario de tipo administrador.
Escenario principal de éxito : Se elimina la mesa de la base de datos.

Ver estadísticas

Evento: El usuario selecciona el apartado estadísticas en el menú principal de la plataforma de administradores
Precondiciones: El usuario debe de haber iniciado sesión.
Escenario principal de éxito : Se muestran las estadísticas sacadas de los pedidos del restaurante.
Extensiones: Dentro de este apartado se encontrarán varios tipos de estadísticas que se mostrarán en los próximos apartados.

Ver pedidos

Evento: El usuario selecciona el apartado pedidos en el menú principal de la plataforma de administradores
Precondiciones: El usuario debe de haber iniciado sesión.
Escenario principal de éxito : Se muestran los pedidos en tiempo real de los clientes del restaurante.

Editar pedidos

Evento: El usuario modifica algún dato del pedido seleccionado.
Precondiciones: El usuario debe de haber iniciado sesión.
Escenario principal de éxito : Se modifica el pedido de la base de datos.

3.2.2 Clientes

A continuación veremos un diagrama en el cual se resume los casos de uso relacionados con la plataforma para realizar pedidos a la cual accederán los clientes del restaurante.

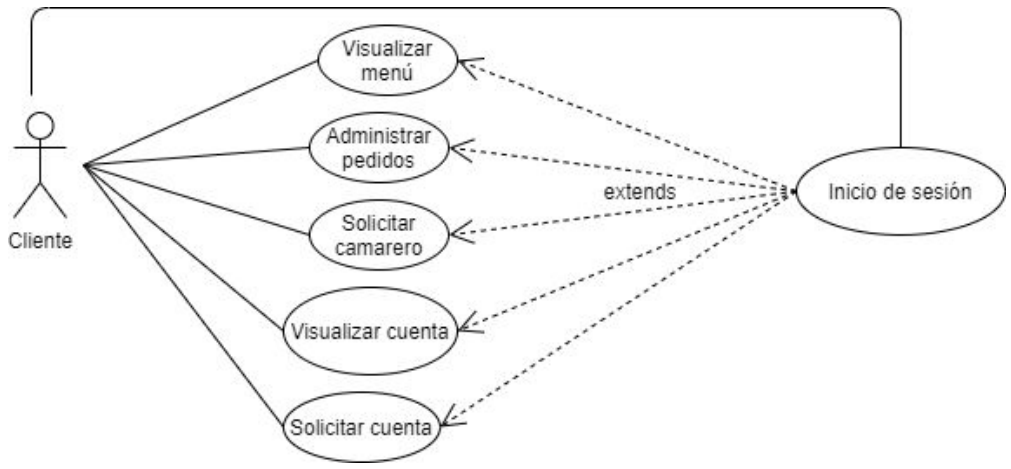

Figura 3.2 Casos de uso de clientes

Al igual que en la parte de administración, todas las funcionalidades dependen del inicio de sesión, en este caso, con el código y la clave de las mesa.

Inicio de sesión

Evento: El cliente escribe la clave y el código de la mesa.
Escenario principal de éxito : Se accede a la plataforma del cliente.
Extensiones: Si las credenciales no son correctas se muestra un error.

Visualizar menú

Evento: El cliente puede visualizar todos los productos disponibles del restaurante.
Precondiciones: El cliente debe de haber iniciado sesión.

Realizar pedidos

Evento: El cliente selecciona un producto y le da al botón de pedir
Precondiciones: El cliente debe de haber iniciado sesión.
Escenario principal de éxito : Se añade el pedido a la base de datos asociado a la mesa en la que se encuentra**.**

Cancelar pedidos

Evento: El cliente selecciona uno de sus pedido y pulsa el botón de cancelar.
Precondiciones: El cliente debe de haber iniciado sesión.
Escenario principal de éxito : Se marcará el pedido como cancelado.
Extensiones: En el caso de que el pedido ya haya pasado a la cocina no saldrá la opción de cancelar.

Solicitar camarero

Evento: El cliente pulsa el botón de solicitar camarero.
Precondiciones: El cliente debe de haber iniciado sesión.
Escenario principal de éxito : Se envía una alerta al camarero para que vaya a atender a la mesa

Visualizar cuenta

Evento: El cliente pulsa el botón de cuenta.
Precondiciones: El cliente debe de haber iniciado sesión.
Escenario principal de éxito : Se muestran los pedidos realizados con sus importes correspondientes.

Solicitar cuenta

Evento: El cliente pulsa el botón de solicitar la cuenta
Precondiciones: El cliente debe de haber iniciado sesión.
Escenario principal de éxito : Se envía una alerta al camarero para que vaya a entregar la cuenta.
Extensiones: Se podrá solicitar el pago en efectivo o con tarjeta.

3.3 Arquitectura a utilizar

Para el desarrollo del proyecto se va a realizar una aplicación distribuida de tipo cliente-servidor. El nombre “cliente” hace referencia a el dispositivo el cual hace peticiones para mostrar la información al usuario final, no confundir con el cliente del restaurante, ya que los dispositivos de los empleados y administradores de la plataforma también serán “clientes” visto desde la perspectiva de la arquitectura. A continuación vamos a ver cómo sería una aplicación cliente-servidor normal:

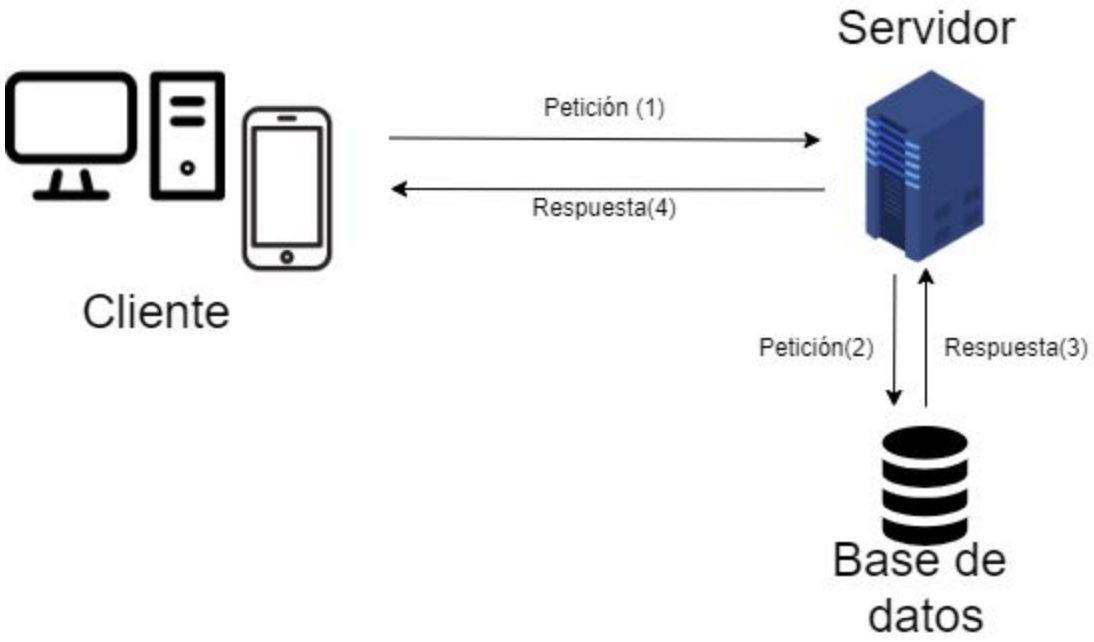

Figura 3.3 arquitectura cliente-servidor

Como vemos, bajo esta arquitectura realizamos peticiones al servidor desde el cliente. Y el servidor su vez realiza una petición a la base de datos (esta puede estar alojada en el mismo servidor o en otro). Después de esto el servidor procesa los datos para dar la respuesta al cliente.

Mediante este proceso podemos obtener cualquier dato de la base de datos, así como modificarlos. El problema está cuando queremos tener información en tiempo real. En el caso de nuestro proyecto se daría en el momento de querer visualizar los pedidos realizados por los clientes, para poder verlos continuamente tendríamos que realizar muchas peticiones completas al servidor y en ellas procesar todo la información así como montar la página necesaria para la correcta visualización del empleado en su dispositivo. Para evitar este fenómeno podemos realizar la siguiente modificación.

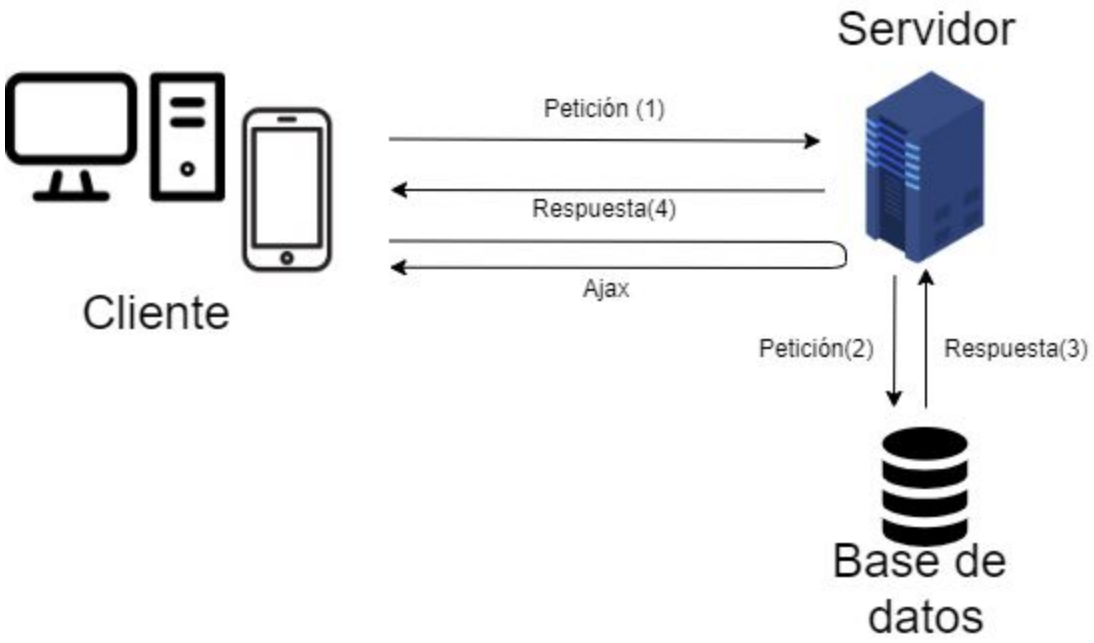

Figura 3.4 Arquitectura cliente-servidor con Ajax

El la figura 3.4 vemos que además de realizar la petición inicial en la cual obtenemos la página, también realizamos peticiones AJAX. AJAX, que significa Asynchronous JavaScript And XML está pensado para poder realizar peticiones al servidor sin recargar la página. Por lo tanto, el cliente pedirá en un principio los datos necesarios para montar la página y más adelante podrá pedir más datos, los cuales pueden ir cambiando con el paso del tiempo. Como hemos visto antes esto se puede dar en el caso de querer mostrar los pedidos de los clientes en tiempo real, ya que estos pedidos se irán generando continuamente y queremos verlos siempre actualizados en nuestra página de administración.

En el apartado 3.5 API veremos las diferentes tipos de peticiones que se van a poder efectuar tanto desde la plataforma de administración como desde la de clientes.

3.4 Base de datos

Para guardar todos los datos es necesario tener una base de datos que permita estructurar toda la información para así poder acceder a los datos, añadir más, modificarlos o eliminar los que ya no sean necesarios. Para el proyecto se va a utilizar una base de datos relacional. Esta se basa en tablas, y cada uno de los componentes que van a ser nombrados será representado por una. En el apartado de implementación veremos más acerca de la tecnología utilizada para mantener la base de datos.

La base de datos diseñada presenta 4 partes diferenciadas, a continuación vamos a verlas individualmente.

3.4.1 Productos

Los productos son la base del proyecto presentado, este, como su nombre indica, representa un producto de un restaurante. Un ejemplo de este podría ser un plato o una bebida. Vamos a ver qué información se necesita guardar para representar un producto.

Tabla Producto

- Id: Identificador único del producto.
- Nombre: Nombre del producto.
- Descripción: Descripción del producto.
- Precio: Precio del producto.
- Ingredientes: Lista de ingredientes.
- Imagen: Url de la imagen.

A parte de los campos anteriormente también se plantea que el producto esté clasificado en una categoría, como estas categorías se repetirán vamos a crear una tabla en la base de datos separada, esta contendrá los siguientes campos:

Tabla Categoría

- id: Identificador único de la categoría.
- Nombre: Nombre de la categoría.
-

Por otro lado también se desea que los productos puedan tener extras, por lo que se plantean otras dos tablas llamadas Extra y Opciones.

Tabla Extra

- id: Identificador único del extra.
- Nombre: Nombre del extra.
- Múltiple: Indica si se pueden seleccionar varias opciones.

Tabla Opción

- id: Identificador único de la opción.
- Nombre: Nombre de la opción.
- inc_dec: Indica si el precio de la opción es descontado o añadido al precio inicial del producto.
- Precio: Precio de la opción.

Las tablas vistas anteriormente están relacionadas de la siguiente manera:

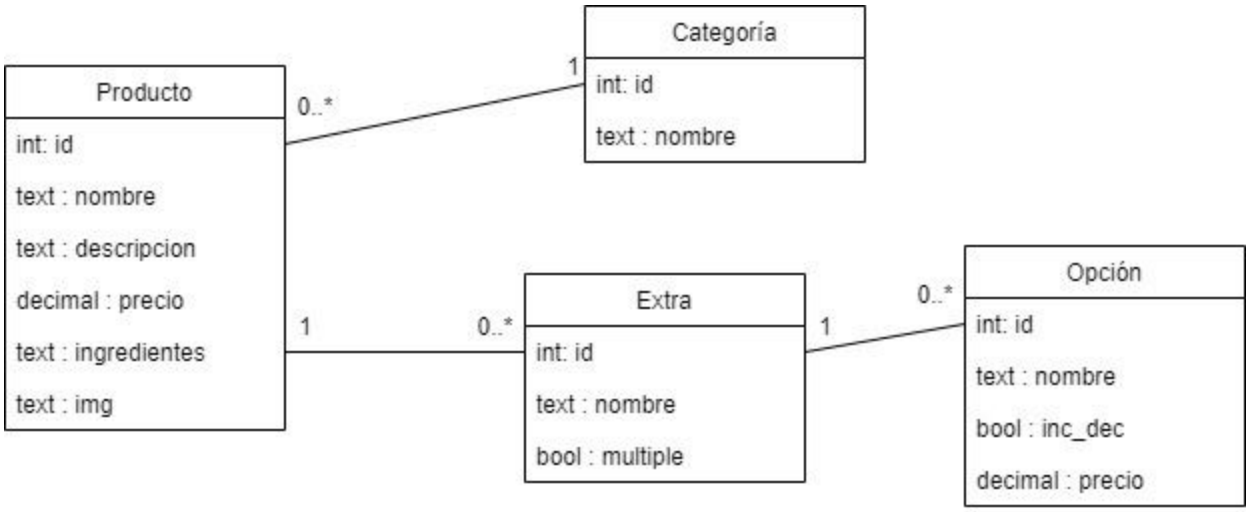

Figura 3.5 Tabla productos y sus relaciones

En el diagrama anterior vemos cómo se relacionan las tablas anteriormente explicadas. Vamos como la tabla producto es la parte central la cual está relacionada de manera directa o indirecta con todas las demás.

3.4.2 Pedidos

Otro elemento fundamental en la plataforma son los pedidos. A parte de esta, habrán varias tablas auxiliares que nos permitirán guardar los pedidos de manera ordenada para después poder acceder a estos en función de la mesa en la que va asociada, el producto que se ha solicitado, etc. A continuación veremos los campos de la tabla de pedidos.

Tabla Pedido

- Id: Identificador único del pedido.
- Estado: Estado el pedido, explicado más abajo.
- Notas: Notas extras para especificar pedido.
- Fecha inicio: Fecha y hora a la que se ha enviado el pedido.
- Fecha fin: Fecha y hora a la que se ha entregado el pedido a la mesa.
- Extras: Lista de extras y sus respectivas opciones.
- Precio : Precio final del pedido, sumando extras al precio inicial del producto.

En la tabla vemos como hay un apartado de estado, este representa en qué momento del ciclo de vida del pedido se encuentra nuestro pedido. A continuación veremos que estados existen.

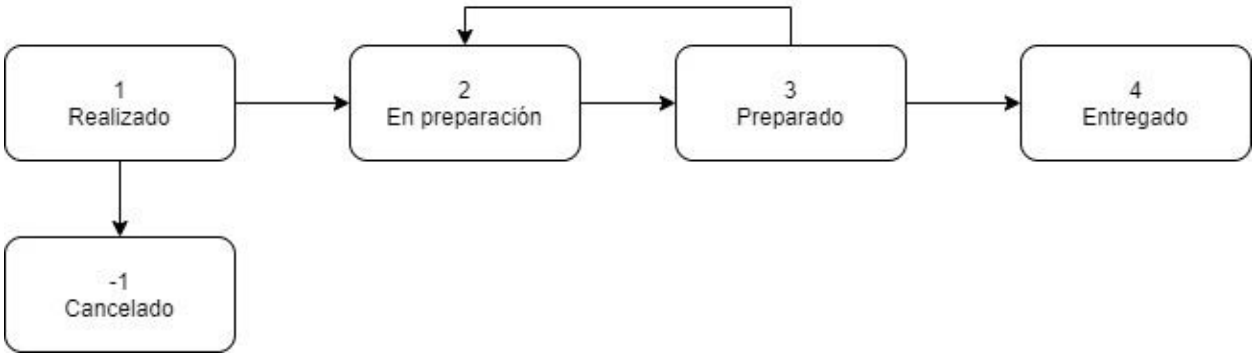

Figura 3.6 Estados de pedidos

En la imagen anterior vemos como un pedido puede estar en 5 estado distintos, y a que estado puede pasar en función de las acciones del personal del restaurante.

3.4.3 Mesas

En este apartado veremos los campos necesarios para hacer referencia a una mesa. Esta es un elemento indispensable en nuestra plataforma, ya que cada vez que se realice un pedido este debe ser asociado a una mesa.

Tabla Mesa

- Id: Identificador único de la mesa
- Código: Identificador único de la mesa que verá el cliente
- Nombre: Nombre de la mesa
- Estado: Estado de la mesa, se verá más adelante.
- Clave: contraseña de la mesa para poder realizar pedidos desde ella.

Los estados en los que puede estar una mesa son los siguientes:

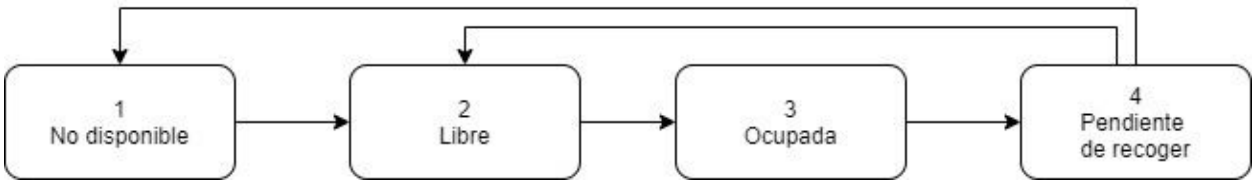

Figura 3.7 estado de mesa

3.4.4 Usuarios

Cada empleado deberá tener su usuario para poder acceder a la plataforma. Esta tabla no tiene relación con las anteriormente vistas.Vamos a ver que campos tendremos en cuenta para guardarlo.

Tabla Usuario

- Id: Identificador único del usuario
- Nombre: Nombre del usuario
- Password: Contraseña del usuario.
- Permisos: Permisos del usuario. 1→ Administrador, 2→ Empleado.

3.4.5 Relacionando tablas

Como hemos visto antes la tabla de pedidos debe ser relacionada con la tabla de productos por un lado y por otro con la de mesas, para tener guardada la información de que producto se ha solicitado y a que mesa hay que entregarlo.

Para el caso de las mesas se debe crear una tabla extra, ya que queremos ver para cada cliente que pedidos se han hecho, de realizar esta relación de manera directa cada mesa tendría todos los pedidos realizados y no se podrían analizar los datos de manera efectiva. Por ello se ha creado la tabla de sesión, vamos a ver que campos tiene.

Tabla Sesión

- Id: Identificador único de la sesión.
- Fecha inicio: Fecha y hora a la que se ha iniciado la sesión.
- Fecha fin: Fecha y hora a la que se ha finalizado la sesión.
- Estado: Estado de la sesión, 1→ Activa, 2→ Finalizada.

Vista la tabla de sesión vamos a ver cómo se relaciona esta con las mesa y con la de pedidos.

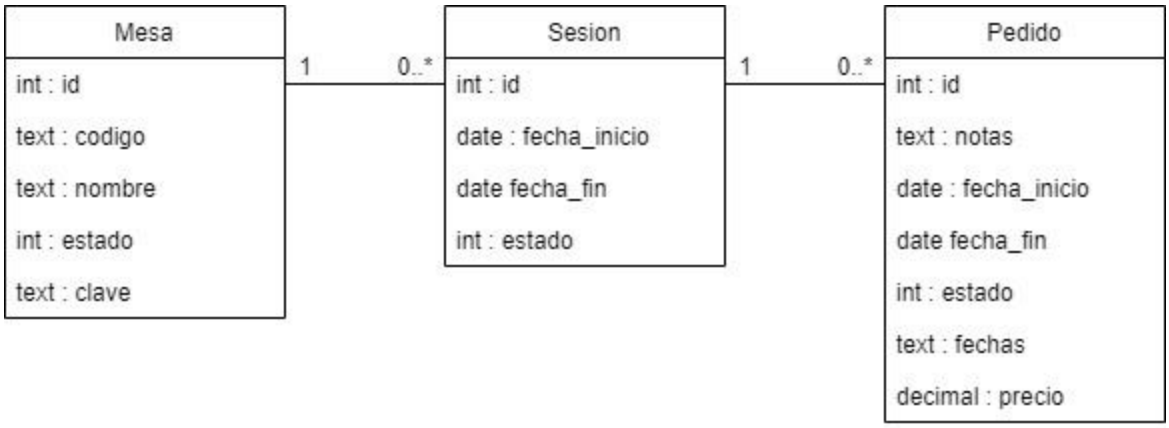

Figura 3.8 Relación tabla pedido y mesa

En la imagen anterior vemos como la tabla de sesión relaciona los pedidos con las mesas, permitiendo así tener un registro de todos los pedidos de cada mesa separados en los diferentes clientes que lo han solicitado.

Y como hemos visto anteriormente también se debe relacionar la tabla de pedidos con la de productos. Vamos a ver cómo se realiza esto.

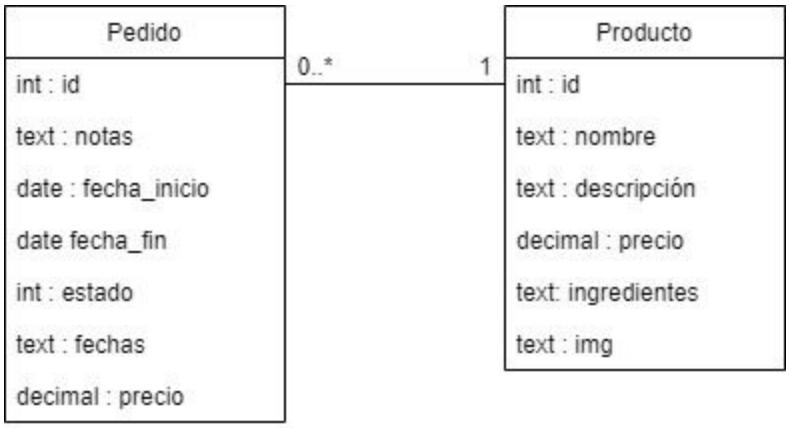

Figura 3.9 Relación tabla pedido y producto

Como vemos en la imagen, cada pedido debe de estar asociado a un producto, y cada producto puede tener muchos pedidos asociados.

Después de haber visto todas las tablas vamos a ver como quedaría la base de datos completa.

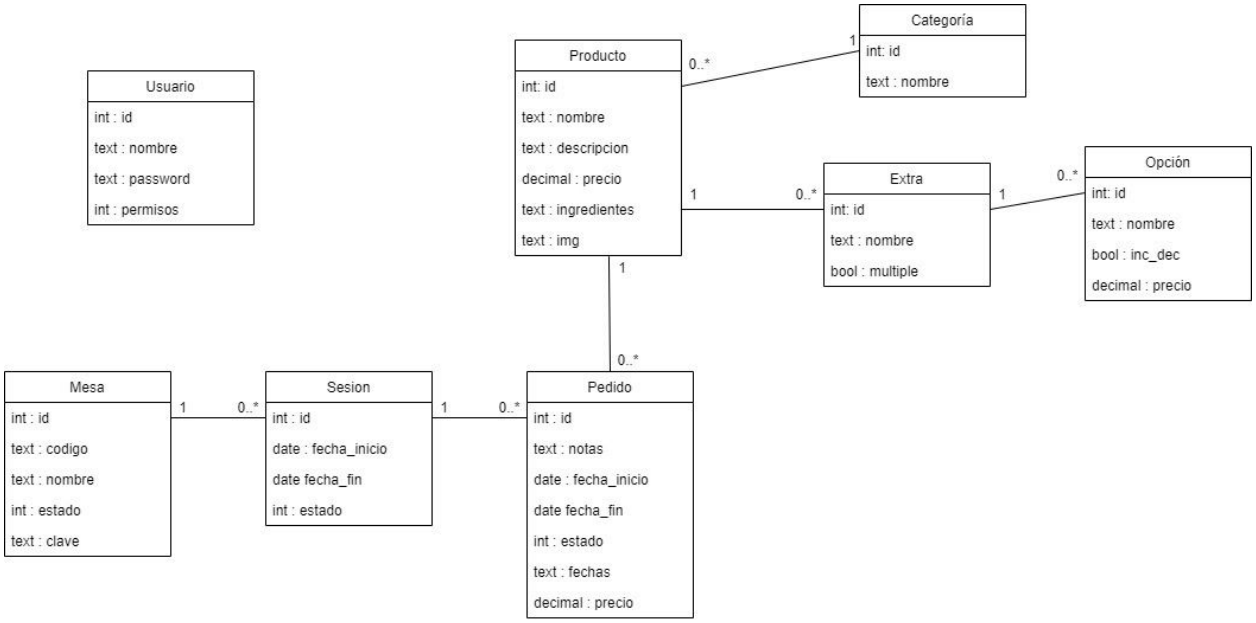

Figura 3.10 Base de datos completa

3.5 API

En este apartado vamos a ver documentación de la API desarrollada, esta está dividida en dos. Por un lado tenemos la API que utiliza la plataforma de administración para que los empleados puedan modificar y consultar datos de los pedidos o las mesas, y por otra parte la API del cliente, la cual permite la visualización de los pedidos realizados por su mesa así como los productos disponibles.

3.5.1 API plataforma administrador

Los archivos de esta API se encuentran en **/admin/api/** y son llamados desde la plataforma de empleados para poder obtener o modificar información acerca de los pedidos, mesas, etc. Para poder utilizar dichas funcionalidades previamente habrá que iniciar sesión con unas credenciales previamente guardadas en la base de datos.

Nombre	**Descripción**	**Parámetros**
obtener_pedidos	Obtiene los pedidos realizados por los clientes.	**id_empleado**: int → Filtra por empleado
obtener_mesas	Obtiene la información de las mesas	
obtener_productos_count	Obtiene la cantidad de pedidos realizados de cada producto	
obtener_sesiones	Obtiene todas la sesiones realizadas de los clientes	
obtener_facturacion	Obtiene la facturación entre las fecha proporcionadas. Si no lo especificas irá desde el primer pedido hasta el último.	**fecha_inicio**: date → Desde donde se cuenta **fecha_fin** : date → Hasta donde se cuenta **opcion** : int → Intervalo entre mediciones(0:Un día, 1: Una semana, 2: Un mes, 3: Un año)
add_extra	Añade un extra y opciones a un producto	**nombre**: string → Nombre del extra **tipo**: int → Tipo de extra (0: Sencillo ,1: multiple) **id**: int → Id del producto **lista_opciones**: json → Lista de opciones para este extra

editar_estado_mesa	Edita el estado de una mesa.	**id_mesa**: int → Id de la mesa a editar **estado**: int → Nuevo estado de la mesa **clave**: int → Clave de la mesa(en caso de que el nuevo estado sea 3)
editar_estado_pedido	Edita el estado de un pedido.	**id_pedido**:int → Id del empleado **estado**: int → nuevo estado del pedido
editar_usuario_mesa	Edita el usuario asignado a una mesa	**id_mesa**: int → Id de la mesa a modificar **id_empleado**: int → Id del nuevo usuario

3.5.2 API Cliente

Los archivos de esta API se encuentran en **/app/api/** y son llamados desde la plataforma del cliente para poder obtener o modificar información acerca de los pedidos asociados a su mesa, por lo que para ello primero tendrá que iniciar sesión con una clave válida.

Nombre	**Descripción**	**Parámetros**
cancelar_cuenta	Cancela la petición de cuanta previamente realizada	
cancelar_pedido	Cancela uno de los pedidos realizados por la mesa.	**id_pedido**:int → Id del pedido a cancelar
obtener_carta	Obtiene la carta disponible del restaurante.	
obtener_pedidos	Obtiene los pedidos realizados por la mesa.	
petición especial	Realiza una petición de llamar a camarero o pedir la cuenta	**opcion**: int → tipo de petición(-1: petición de camarero, -2: solicitar cuenta en efectivo, -3: solicitar cuenta en tarjeta)
realizar_pedido	Realiza un pedido para la mesa asociada.	**pedido**: json con los siguientes datos: **id**: int, **notas**: string, **extras**: json

3.6 Estructura de ficheros

A continuación vamos a ver cómo se han estructurado los ficheros para la implementación del proyecto. Vemos como las carpetas admin y app presentan la misma estructura interna.

/admin/ → Archivos relacionados con la plataforma de administración.

- **api/** → API de la plataforma de empleados
- **css/** → Archivos de estilos
- **js/** → Archivos de javaScript
- **vistas/** → Vistas que utilizan los ficheros php
- **funciones/** → Funciones que utilizan los ficheros php

/app/ → Archivos relacionados con la plataforma de cliente.

- **api/** → API de la plataforma de clientes
- **css/** → Archivos de estilos
- **js/** → Archivos de javaScript
- **vistas/** → Vistas que utilizan los ficheros php
- **funciones/** → Funciones que utilizan los ficheros php

index.php → Web principal del restaurante

carta.php → Web para visualizar la carta sin necesidad de iniciar sesión

4 Implementación

En este capítulo veremos como se ha implementado la plataforma descrita en los capítulos anteriores. En primer lugar se explicarán las tecnologías utilizadas, en segunda parte veremos algunas de las partes más importantes del código, y por último veremos el resultado obtenido.

4.1 Tecnologías utilizadas

En este apartado vamos a ver con qué tecnologías vamos desarrollar el proyecto planteado. Vamos a clasificar estas en dos categorías, Backend y Frontend. A continuación vamos a ver que es cada una de ellas.

4.1.1 Backend

Backend es la capa de la aplicación que corre en el servidor, esta es la encargada de acceder a los datos guardados y a modificarlos. Es esencial que el acceso a los datos se haga en el servidor, ya que de hacerlo desde el dispositivo del cliente habría brechas de seguridad al permitir al usuario final entrar en nuestra base de datos. Para que el cliente pueda obtener datos y verlos en su dispositivos se crean unas directivas que devuelven los datos que pide el cliente si este se ha autentificado correctamente en la plataforma. Estas directivas se han mostrado en el apartado API del capítulo anterior.

Una vez introducido el concepto de backend vamos a ver las tecnologías escogidas para implementar la plataforma. Existen multitud de tecnologías que podemos utilizar para el desarrollo, en este se ha decidido escoger el conjunto de tecnologías LAMP. LAMP es un acrónimo de Linux, Apache, Mysql (o MariaDB) y Php (También Perl o Python en algunas ocasiones). Se ha decidido utilizar esta infraestructura debido a que es de código abierto y a que hay multitud de documentación disponible. A continuación vamos a analizar cada una de las tecnologías implicadas.

Linux

Linux es un sistema operativo de tipo Unix. El desarrollo del núcleo de este fue encabezado por Linus Torvalds en la década de los noventa. Es el sistema operativo por excelencia en los servidores de la actualidad, según el portal W3techs se calcula que el 70.3% de los servidores utilizan Linux en la actualidad (A fecha de Marzo de 2020).

Existen múltiples distribuciones de linux las cuales servirían para la realización del proyecto. En este caso se ha utilizado la distribución Debian en su versión 8.1. Debian es una de las distribuciones Linux populares y tiene una comunidad de desarrolladores muy grande que se encarga de dar soporte y seguir desarrollando este.

Apache

Apache es un servidor Web HTTP desarrollado por Robert McCool para plataformas Unix. Actualmente el desarrollo y el mantenimiento de esta a cargo de la Apache Software Foundation.

Este servicio que instalaremos en nuestro servidor nos permitirá recibir las peticiones http y responder a estas con la respuesta adecuada para que el cliente pueda visualizar la información que requiera.

Php

Php es uno de los lenguajes más utilizados para el desarrollo web en el lado del servidor. Este se desarrolló para ser utilizado como preprocesador de hipertexto, pero posteriormente se acabaría utilizando para el desarrollo web. Es lenguaje es utilizado para la generación de páginas HTML. Fue diseñado por Rasmus Lerdorf en el año 1995. W3Techs afirma que el 78.5% de los servidores web utilizan Php a fecha de marzo de 2020.

Nuestro servidor Apache será el encargado de ejecutar los ficheros Php en el momento en el que un cliente haga la petición.

MariaDB

MariaDB es un sistema gestor de base de datos desarrollado a partir de MySql. Después de la compra de Sun Microsystems por parte de Oracle se desarrolló MariaDB con licencia GPL (general public license).

MariaDB será el sistema que permita crea la base de datos diseñada anteriormente y poder modificar los datos de esta así como acceder a ellos.

A continuación vamos a ver un diagrama de las anteriores tecnologías backend unificadas.

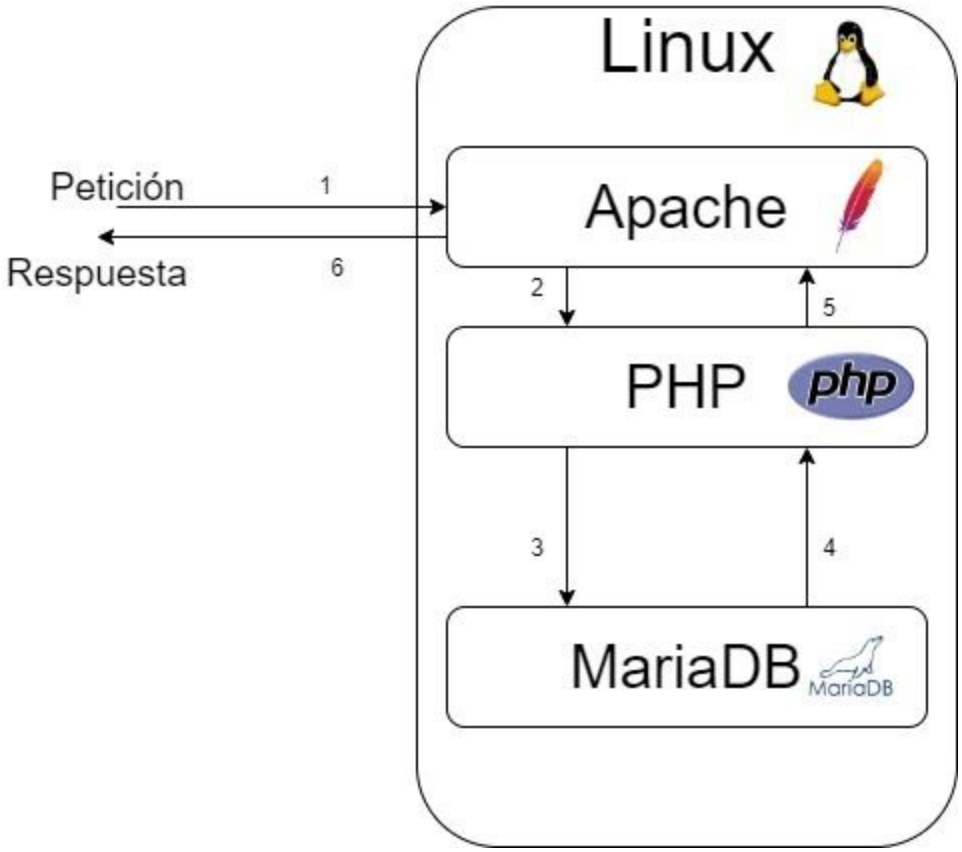

Figura 4.1 Tecnologías Backend

En la imagen anterior vemos como dentro de nuestro servidor Linux encontramos las anteriores tecnologías mencionadas. Cuando llega una petición Apache es el encargado de ejecutar el fichero Php necesario. Por otro lado, Php accede a la nuestra base de datos y genera la información que se enviará al cliente como respuesta. A partir de aquí es el dispositivo del cliente el encargado de presentar los datos de una manera ordenada, a continuación vamos a ver qué tecnologías están implicadas en ese proceso.

4.1.2 Frontend

Como hemos visto anteriormente, el Backend es la parte de nuestra aplicación que se ejecuta en el servidor. Pero por otro lado tenemos el Frontend. El Frontend engloba todo lo relacionado con la ejecución de la plataforma en el dispositivo del cliente.

Los navegadores web de la actualidad entienden únicamente tres lenguajes de programación, estos son HTML, CSS y JavaScript. A continuación veremos cual es la función de cada uno de ellos y que librerías y frameworks externos se han utilizado para facilitar el desarrollo.

HTML

HTML (HyperText Markup Language) es un lenguaje de marcado para el desarrollo de páginas web. Fue diseñado por Tim Berners-Lee en 1991 como una ampliación a SGML. En la actualidad existen varios lenguajes de marcados, como son XML o XHTML. Pero HTML es el más popular y el que se utilizada para el desarrollo de aplicaciones web.

HTML tiene una serie de etiquetas que describen la estructura de la página web. A parte de la estructura, también deseamos que nuestra web tenga estilos, ya sean los colores, estilos de las textos o la alineación de los elementos. Para ellos se utiliza CSS.

CSS

CSS (Cascading Style Sheets) fue propuesto por Håkon Wium Lie en 1994 para dar estilos a las páginas web. CSS es un lenguaje que describe los estilos de la página, con este podremos hacer que nuestra página web luzca como queramos.

Para facilitar el desarrollo existen multitud de librerías y framework con estilos predefinidos que podemos utilizar. En este caso se ha decidido utilizar Bootstrap. Bootstrap es un conjunto de herramientas las cuales nos permitirán entre otras cosas utilizar estilos de botones predefinidos y hacer que nuestra plataforma sea responsive (Se adapte a la resolución del dispositivo).

Después de aplicar HTML y CSS tendremos un web con estructura y estilo, pero falta la funcionalidad, para ello se utilizará JavaScript.

JavaScript

JavaScript es un lenguaje de programación interpretado que se ejecutará en el navegador. Fue desarrollado por Brendan Eich para el navegador Netscape y actualmente es compatible en todos los navegadores modernos.

JavaScript es el lenguaje que va a dar funcionalidad a nuestra página. Hasta ahora hemos visto qué se utiliza para crear la estructura y dar estilos, pero carecía de funcionalidad alguna. Con JavaScript podremos detectar eventos que realice el usuario, como apretar un botón, y realizar peticiones al servidor para que este nos envíe información que mostraremos en pantalla, como hemos visto en el apartado de "Arquitectura a utilizar".

En la actualidad hay muchas librerías que nos permiten agilizar el proceso de desarrollo, las más importantes en la actualidad son: Angular, Vue JS y React. Con todas ellas se podría desarrollar la plataforma planteada. En este caso se ha decidido utilizar React por la experiencia previa de uso.

React es una librería desarrollada por los ingenieros de Facebook. Esta permite separar nuestra aplicación en componentes reutilizables que podrán ser utilizados por representar elementos. En nuestro caso un componente podría ser un pedido, un producto o una lista que contenga cualquiera de estos.

Ya hemos visto las tecnologías implicadas en el frontend, ahora veremos como se ha desarrollado la plataforma.

4.2 Desarrollo

En este apartado veremos como se ha desarrollado las funcionalidades más importantes de nuestro proyecto. Veremos configuraciones en el servidor, e implementaciones de código tanto en la parte del servidor (backend) como en la parte del cliente (frontend).

4.2.1 Configuración en el servidor

En primer lugar se ha de configurar el servidor Apache para que responda a las peticiones como nosotros deseamos, la configuración utilizada ha sido la siguiente.

```
<VirtualHost *:80>

        ServerAdmin carlos@carlosh.es
        ServerAlias *tfg.carlosh.es
        ServerName www.tfg.carlosh.es

        DocumentRoot /www/tfg/html

        ErrorLog /www/tfg/logs/www/error.log
        LogLevel warn

        CustomLog /www/tfg/logs/www/access.log combined

        ErrorDocument 404 /error.php
        ErrorDocument 403 /error.php

RewriteEngine on
RewriteCond %{SERVER_NAME} =www.tfg.carlosh.es [OR]
RewriteCond %{SERVER_NAME} =*tfg.carlosh.es
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
```

Figura 4.2 Configuración de Apache

En la imagen anterior vemos como se ha establecido tfg.carlosh.es como dominio principal. El dominio de carlosh.es es un dominio que se tenía registrado anteriormente, de no tenerlo se podría especificar directamente la Ip del servidor donde se vaya a realizar la implementación. En la configuración se especifica que la ruta donde irán nuestro fichero que van a ser servidos al cliente se encontrarán en **/www/tfg/html/** mientras que los errores se registrarán en **/www/tfg/logs/www/error.log**

Esta configuración se podrán hacer peticiones HTTP al servidor y este dará respuestas a partir de los fichero que contenga en el directorio anteriormente nombrado. A pesar de que esta configuración realizaría la tarea que queremos carece de seguridad, ya que la comunicación entre el cliente y el servidor no estaría cifrada. Para solucionar esto habría de decir Apache que utilizara peticiones HTTPS. Para hacer esto de manera simplificada se ha utilizado la herramienta Certbot, la cual convierte nuestras rutas de apache HTTP directamente a HTTPS, además de instalar un certificado SSL y renovarlo

automáticamente cuando sea necesario. Después de ejecutar esta herramienta se ha generado automaticamente el siguiente fichero de configuración.

```
<IfModule mod_ssl.c>
<VirtualHost *:443>

        ServerAdmin carlos@carlosh.es
        ServerAlias *tfg.carlosh.es
        ServerName www.tfg.carlosh.es

        DocumentRoot /www/tfg/html

        ErrorLog /www/tfg/logs/www/error.log
        LogLevel warn

        CustomLog /www/tfg/logs/www/access.log combined

        ErrorDocument 404 /error.php
        ErrorDocument 403 /error.php

SSLCertificateFile /etc/letsencrypt/live/www.tfg.carlosh.es/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/www.tfg.carlosh.es/privkey.pem
Include /etc/letsencrypt/options-ssl-apache.conf
</VirtualHost>
</IfModule>
```

Figura 4.3 Configuración de Apache segura

Vemos como el puerto 80 (HTTP) por el 443 (HTTPS) y se han añadido algunas líneas que hacer referencia al certificado SSL instalado.

4.2.2 Conexión a base de datos

Para acceder la conexión a la base de datos diseñada con anterioridad se ha creado un pequeño programa en php que será utilizado después por los demás ficheros que accedan a la base de datos.

```php
function sqli($query) {
  GLOBAL $mysqli_local;
  if (!$mysqli_local) {
   if (!$mysqli_local = mysqli_connect("localhost","root","","tfg")) {
      echo "Error al conectar a la base de datos" ;
      exit;
   }
  }
  $mysqli_local->set_charset("utf8");
  $r=$mysqli_local->query($query);
  if (!$r) {
     echo "Ha ocurrido un error al generar la pagina. Por favor, intentalo de nuevo<br>";
     echo "SQL: $query<br>";
     exit;
  }
  return $r;
}
```

Figura 4.4 Conexión a la base de datos

Vemos como en la función presentada se crea una conexión a la base de datos y se hace la consulta que se ha pasado como parámetro. Vamos a ver cómo sería utilizada esta función en el programa.

```php
<?php
  include "sql.php";
  $consulta = sqli("SELECT id,nombre FROM productos");
  while($producto = mysqli_fetch_array($consulta)){
    echo "Id: $producto[id] Nombre: $producto[nombre]";
  }
?>
```

Figura 4.5 Ejemplo de acceso a la base de datos

En el fragmento de código se realiza una consulta a la base de datos en las que se solicitan los identificadores y los nombres de todos los productos.

A pesar de que el código anterior realizaría la funcionalidad deseada esta presenta una gran brecha de seguridad que veremos a continuación.

SQL Injections

Una SQL injection o inyección SQL es una vulnerabilidad que permite que un usuario malintencionado haga consultas a la base de datos que nosotros no deseamos, vamos a ver esto con un ejemplo. Imaginemos que se le pide a un usuario el nombre y la contraseña, este pone de nombre "**Carlos**" y de contraseña "**Hinojosa';drop table usuarios;**". El programa podría generar una consulta similar a la siguiente:

```
SELECT FROM usuarios WHERE nombre = 'Carlos' AND password = 'Hinojosa';DROP TABLE usuarios;'
```

Figura 4.6 Ejemplo de SQL Injection

Esta consulta eliminaría la tabla de usuarios, ya que MariaDB interpretaría que hay dos consultas separadas por **;**. Por otro lado esto también daría error cuando un usuario pusiera el carácter **'** dentro de un campo de texto, ya que estos en en SQL están separados por comilla simple. Una manera de solucionar este problema es utilizar la función **addslashes** de Php. Esta función pone el carácter "\" antes de cada carácter especial que encuentre, como son los punto y como o las comillas simples. Esta función será utilizada más adelante para convertir la peticiones de los usuarios.

4.2.3 Autenticación

Otras de las funcionalidades más importantes a implementar es la autenticación de los usuarios. Tanto la autenticación de los clientes como la de los empleados se basa en la misma estructura de código. Para esto utilizamos las sesiones de Php, estas nos permiten guardar un información de un determinado dispositivo en el servidor, y por lo tanto tener un registro de los usuarios que han iniciado sesión. Vamos a ver como se ha implementado esto para los empleados que utilicen la plataforma.

Figura 4.7 Autenticación de empleados

En la imagen anterior vemos cómo en primer lugar se utiliza la función addslashes en todos los datos enviados mediante HTTP Post. Más adelante rellena las sesiones si es que el usuario ha enviado su nombre y contraseña. Y por último se hace la consulta a la base de datos para ver si existe un usuario con las credenciales proporcionadas. Este código será ejecutado siempre que se acceda a una página de la plataforma de administración, y podrá comprobarse por la variable **$id_usuario** si se ha iniciado sesión o no. Con la variable de **$permisos_usuario** podremos conocer qué acciones puede realizar el usuario en la plataforma de administración.

De manera muy similar a la anterior, se ha implementado un inicio de sesión en la plataforma de pedidos, para que los clientes puedan acceder desde sus mesas.

```php
<?
header("Content-type: text/html; charset=utf8");
include "sql.php";
include "funciones.php";
session_start();
foreach ($_POST as $key => $value) {
  $_POST[$key] = addslashes($value);
}

if($_REQUEST["codigo"] != "" && $_REQUEST["clave"] != ""){
  $_SESSION["form_codigo_mesa"] = $_REQUEST["codigo"];
  $_SESSION["from_clave_mesa"] = $_REQUEST["clave"];
}

$id_mesa = -1;
$id_sesion = -1;
if(isset($_SESSION["form_codigo_mesa"]) && isset($_SESSION["form_clave_mesa"])){
  $inicioSesion = sqli("SELECT * FROM mesas WHERE codigo = '$_SESSION[form_codigo_mesa]'
                        AND clave= '$_SESSION[from_clave_mesa]';");
  if($mesa = mysqli_fetch_array($inicioSesion)){
    $id_mesa = $mesa["id"];
    sqli("UPDATE mesas SET estado = 3 WHERE id = $id_mesa AND estado = 2");
    $id_sesion = obtenerIdSesion($id_mesa);
  }
  else{
    $id_mesa = -2;
    session_unset();
  }
}
?>
```

Figura 4.8 Autenticación de clientes

Vemos como la autenticación de clientes es muy similar a la de empleados, pero que la de clientes realiza la petición en la tabla de mesas y no la de usuarios, por lo que los campos requeridos son el de código y el de clave en ver el de nombre y contraseña. En este caso se guardan dos variables, por un lado **$id_mesa**, que como su nombre indica identifica la mesa en la que se van a realizar pedidos. Y por otro lado **$id_sesion**, que asociará los pedidos con la sesión de compra.

Más adelante veremos como se puede hacer que el cliente entre directamente a la plataforma escaneando un código QR desde su dispositivo móvil.

4.3.4 Formularios de administradores

Otra de las funcionalidades más importantes del proyecto consiste en poder editar, visualizar, crear o eliminar los diferentes registros de la base de datos. A pesar de que cada uno de estos tiene diferentes campos la forma en la que se implementará será muy similar.

Para ello se utilizarán formularios web que permitirán enviar peticiones de tipo Post al servidor. Una vez que el servidor tiene la información accede a la base de datos para realizar la operación conveniente.

Vamos a ver esto con un ejemplo en la tabla de categorías, esta es la más simple, ya que cada categoría sólo contiene un campo identificador y un nombre.

Creación

En primer lugar vamos a ver cómo se realizará la operación de crear una nueva categoría mediante la funcionalidad INSERT de SQL.

```
if( isset($_POST["add_categoria"]) && isset($_POST["nombre"]) ) {
  if($permisos_usuario  != 1){
    $error  = "NO tienes permisos";
  }
  else if(!existeRegistro("SELECT id FROM categorias where nombre = '$_POST[nombre]';")){
      sqli("INSERT INTO categorias (nombre) values('$_POST[nombre]');");
  }
}
```

Figura 4.9 Creación de una categoría

En la imagen anterior podemos ver que si se recibe los parámetros **add_categoria** y **nombre** via POST se comprobará si el usuario que ha iniciado tiene los permisos para crear, y en segundo lugar si no existe ya ninguna categoría con el mismo nombre. En el caso de pasar la comprobaciones se guardará la nueva categoría en la base de datos.

Visualización

También será necesario visualizar todos los registros de la base de datos, para ello vamos a realizar una consulta a la base de datos de tipo SELECT.

```
<?
$consulta_categorias = sqli("select * from categorias;");
while($cat = mysqli_fetch_array($consulta_categorias)){
?>
  <div class="card text-left" style="padding:10px;">
    <div class="text-right">
      <button class="btn btn-warning"
       onclick="editar_categoria(<?= $cat["id"] ?>,'<?= $cat["nombre"] ?>')">
        <i class="fa fa-pencil"></i>
      </button>
      <button class="btn btn-danger"
       onclick="eliminar_categoria('<?= $cat["id"] ?>','<?= $cat["nombre"] ?>')">
        X
      </button>
    </div>
    <h3><?= $cat["nombre"] ?></h3>

  </div>
<? } ?>
```

Figura 4.10 Visualización de una categoría

Con la consulta a la base de datos obtendremos todas las categorías y podremos generar la página HTML para visualizarlas. En la imagen también podemos ver que se crean dos botones, a continuación vamos a ver para qué sirven estos.

Edición

Como hemos visto en el apartado anterior, al visualizar cada categoría se crean dos botones, al ser pulsado el botón de editar se ejecutará una función escrita en Javascript que permitirá el usuario introducir el nombre de la nueva categoría.

```
function editar_categoria(id, nombre){
  $.confirm({
    title: 'Editar categoria',
      content: `
      <form id="formulario_editar" action="categorias.php" class="formName" method="post">
        <div class="form-group">
        <input type="text" name="editar" value="${id}" class="form-control" required hidden/>
        Nombre: <br>
        <input type="text"  value="${nombre}"name="nombre" class="form-control" required />
        </div>
      </form>`,
    buttons: {
      formSubmit: {
        text: 'Editar',
        btnClass: 'btn-blue',
        action: function () {
          $("#formulario_editar").submit();
        }
      },
      Cancelar: function () {
          //close
      },
    }
  });
}
```

Figura 4.11 Modal para editar una categoría

En la función anterior se crea un modal que contiene un formulario el cual permitirá al usuario introducir el nombre de la nueva categoría. Al enviar el formulario será el servidor el encargado de editar el registro de la base de datos.

Para editar un registro de la base de datos se utiliza una consulta de tipo UPDATE, vamos a ver como se ha realizado en la tabla de categorías.

```
if(isset($_POST["editar"]) && isset($_POST["nombre"]) ) {
  if($permisos_usuario  != 1){
    $error  = "NO tienes permisos";
  }else{
      sqli("UPDATE categorias SET nombre = '$_POST[nombre]' WHERE id = $_POST[editar];");
  }
}
```

Figura 4.12 Edición de una categoría

Este fragmento de código es muy similar al visto en el apartado de creación, ya que ambos reciben los mismo parámetros. Pero, como hemos comentado anteriormente se utilizará UPDATE para modificar el registro, en vez de el INSERT utilizado para la creación.

Borrado

Por último queda analizar cómo se eliminan los registros de la base de datos. Al igual que en el editado, para eliminar una categoría el usuario pulsara un botón que ejecutará una función creada en JavaScript.

```
function eliminar_categoria(id, nombre){
  if(confirm("Seguro que quieres elimnar la categoria "+  nombre)){
    location.href = "categorias.php?eliminar="+id;

  }
}
```

Figura 4.13 Modal para eliminar una categoría

Esta función es mucho más simple que la de editado, ya que únicamente sirve para confirmar que el usuario quiere eliminar la categoría indicada.

Después de que se confirme se enviará al servidor el id de la categoría que se desea eliminar, esta vez se utilizara una consulta de tipo DELETE.

```
if(isset($_GET["eliminar"])) {
  if($permisos_usuario  != 1){
    $error  = "NO tienes permisos";
  }else{
      sqli("DELETE FROM categorias WHERE id = '$_GET[eliminar]'");
      sqli("UPDATE productos SET id_categoria =0 WHERE id_categoria = $_GET[eliminar];");
  }
}
```

Figura 4.14 Borrado de una categoría

Podemos ver que a parte de eliminar el registro de la base de datos de categorías también se editan los productos que pertenezcan a esa categoría.

Hemos podido ver cómo se crean y modifican los registros de la tabla de categorías, esto se realizaría de manera muy similar en las otras tablas, como productos, mesas y usuarios. La tabla de pedidos se realizará de forma diferente, ya que esta va a ser editada de manera muy recurrente y no queremos que se tenga que recargar entera la página cada vez que se modifique o se cree un pedido.

4.3.5 Pedidos

El núcleo de pedidos es uno de los más importantes de la plataforma, ya que los clientes deben poder solicitar nuevos pedidos y visualizar los realizados previamente y los empleados deben poder ver en tiempo real todos los pedidos realizados por los clientes. Vamos a ver estos dos casos por separado

Clientes

En primer lugar, para que el cliente pueda realizar un pedido, este debe primero poder visualizar todos los productos. Para ello se realiza una petición a la API para obtener un listado con todos los productos agrupados por categorías. Vamos a ver como es el código de la API de obtener la carta.

```php
<?php
include "../funciones/inicioSesion.php";
if($id_mesa <0){
      echo "-1";
      die;
}
$array = array();
$i = 0;
$consulta = sqli("SELECT id, nombre FROM categorias;");
while($categoria = mysqli_fetch_array($consulta)){

    $array[$i] =  new stdClass();
  $array[$i]->nombre = $categoria["nombre"];
  $array[$i]->indice = $i;
  $array[$i]->productos = array();
  $j = 0;
  $consulta2 = sqli("SELECT * FROM productos WHERE id_categoria = $categoria[id];");
  while($producto = mysqli_fetch_array($consulta2)){
    $array[$i]->productos[$j] =  new stdClass();
    $array[$i]->productos[$j]->id =  $producto["id"];
    $array[$i]->productos[$j]->nombre =  $producto["nombre"];
    $array[$i]->productos[$j]->descripcion =  $producto["descripcion"];
    $array[$i]->productos[$j]->precio =  $producto["precio"];
    $array[$i]->productos[$j]->ingredientes =  $producto["ingredientes"];
    $array[$i]->productos[$j]->img =  "/img/productos/".$producto["img"];
    $array[$i]->productos[$j]->extras =  array();
    $k = 0;

    $consulta3 = sqli("SELECT * FROM extras WHERE id_producto= $producto[id];");
    while($extra = mysqli_fetch_array($consulta3)){
      $array[$i]->productos[$j]->extras[$k] = new stdClass();
      $array[$i]->productos[$j]->extras[$k]->id = $extra["id"];
      $array[$i]->productos[$j]->extras[$k]->nombre = $extra["nombre"];
      $array[$i]->productos[$j]->extras[$k]->multiple = $extra["multiple"];
      $array[$i]->productos[$j]->extras[$k]->opciones = array();
      $l = 0;
      $consulta4 = sqli("SELECT * FROM opciones WHERE id_extra= $extra[id];");
      while($opcion = mysqli_fetch_array($consulta4)){
        $array[$i]->productos[$j]->extras[$k]->opciones[$l] =  new stdClass();
        $array[$i]->productos[$j]->extras[$k]->opciones[$l]->id =  $opcion["id"];
        $array[$i]->productos[$j]->extras[$k]->opciones[$l]->nombre =  $opcion["nombre"];
        $array[$i]->productos[$j]->extras[$k]->opciones[$l]->precio =  $opcion["precio"];
        $array[$i]->productos[$j]->extras[$k]->opciones[$l]->inc_dec =  $opcion["inc_dec"];
        $l++;
      }
      $k++;
    }
    $j++;
  }
  $i++;

}
$respuesta = new stdClass();
$respuesta->array = $array;
echo json_encode($respuesta);
```

Figura 4.15 API obtener carta para clientes

Vemos como en el código anterior se obtienen primero las categorías, después los productos, extras y opciones, todo esto se guarda en un array y es enviado al dispositivo cliente en formato JSON. Veamos un ejemplo de una respuesta de este tipo.

```
{
  "array": [
    {
      "nombre": "Pizzas",
      "indice": 0,
      "productos": [
        {
          "id": "19",
          "nombre": "Pizza super queso",
          "descripcion": "Pizza para los verdaderos amantes del queso",
          "precio": "8",
          "ingredientes": "Queso, harina, tomate y salsa de la casa",
          "img": "/img/productos/producto-Pizza super queso-615",
          "extras": [
            {
              "id": "32",
              "nombre": "Añadidos",
              "multiple": "1",
              "opciones": [
                {
                  "id": "49",
                  "nombre": "Extra de queso",
                  "precio": "1",
                  "inc_dec": "0"
                },
                {
                  "id": "50",
                  "nombre": "Bacon doble",
                  "precio": "2",
                  "inc_dec": "0"
                }
              ]
            }
          ]
        },
        {
          "id": "20",
          "nombre": "Pizza pepperoni",
          "descripcion": "Pizza pepperoni",
          "precio": "7",
          "ingredientes": "Pepperoni, harina, tomate y queso",
          "img": "/img/productos/producto-Pizza pepperoni-498",
          "extras": []
        }
      ]
    },
  ]
}
```


Figura 4.16 Json de ejemplo para obtener productos

Este es un fragmento de respuesta Json, vamos como está la categoría “Pizzas” y tiene dos productos asociados. Cada uno de estos productos tiene su propia información, como el nombre o la descripción y por otro lado una lista de extras que puede estar vacía, cada extra contendrá una lista de opciones.

Una vez el cliente tiene la lista de productos ya puede mostrarlos en la pantalla del navegador. Esto se realiza con la ayuda de React y de sus componentes. Vamos a ver como se ha implementado cada uno de los componentes necesarios para mostrar la carta.

```
class Carta extends React.Component {
  constructor(props) {
    super(props)


  }
  render() {
    const listaCategorias = lista_categorias.map((cat) =>
      <Categoria  info={cat}/>
    );
    let btn_cuenta= <button className="btn btn-light wow zoomIn"
                      onClick= {(e) => solicitarCuenta()}>
                      <i className="fa fa-edit"></i> <br/>
                      Solicitar la cuenta
                    </button>
    let btn_camarero= <button className="btn btn-light wow zoomIn"
                        onClick= {(e) => solicitarCamarero()}>
                        <i className="fa fa-user    "></i> <br/>
                        Solicitar Camarero
                      </button>

    return <div class="carta">
              {btn_camarero}
              {btn_cuenta}
            <div className="row">
            {listaCategorias}
            </div>
           </div>
    }
  }
```

```
class Categoria extends React.Component {
  constructor(props) {
    super(props)


  }
  render() {
    let listaProd = lista_categorias[this.props.info.indice].productos;
    const listaProductos = listaProd.map((prod) =>
    <Producto indice_categoria={this.props.info.indice} info={prod} />
    return  <div className="col-sm-6 col-12">
              <details class="categoria">
                <summary className="wow zoomIn">
                  <h1>{this.props.info.nombre}</h1>
                </summary>
                <div className=" container">
                  <div className="row">
                  {listaProductos}
                  </div>
                </div>
              </details>
            </div>

    }
}
```

```
class Producto extends React.Component {
  constructor(props) {
    super(props)
  }
  render() {
    return <div className="col-6">
              <div className="producto card shadow "
                onClick ={(e) => informacionProducto(this.props.info)}>
              <img className="card-img-top" src={this.props.info.img} alt="Card image cap" />
              <div className="card-body">
                <p className="card-text">{this.props.info.nombre}</p>
                {this.props.info.precio} € <br/>
              </div>
            </div>
          </div>

    }
  }
```

Figura 4.17 Componentes React para visualizar carta completa

En las anteriores imágenes podemos ver los componentes de Carta, Categoría y Producto. El componente de Carta se encarga de colocar los botones de solicitar cuenta y solicitar camarero y listar todas las categorías obtenidas en la petición a la API. Por otro lada, el de Categoría lista todos los productos asociados a ella. Y por último, el componente de Producto muestra la información de un determinado producto y espera el evento click sobre él para que el cliente pueda realizar un pedido.

Cuando el cliente selecciona el producto que desea pedir y añade los extras si los desea se realiza la petición al servidor para crear el registro del pedido en la base de datos. Para ello se utilizará la función **$.ajax()** de Jquery para facilitar la implementación.

```
$.ajax({
  type:"POST",
  data :{
    pedido : pedido_json
  },
  url:"https://www.tfg.carlosh.es/app/api/realizarPedido.php",
  success:function(datos){
    if(datos.error){
      alertError(datos.error)
    }

   },
   dataType : "json"
})
```

Figura 4.18 Petición Ajax para realizar pedido

En la imagen anterior vemos el ejemplo de cómo se realizaría la petición AJAX, vemos como a la función hay que pasarle como parámetro un objeto. Este contendrá los datos necesarios para realizar la petición, como la url, el tipo, los datos y una función que se ejecutará cuando se reciba la respuesta a la petición.

Una vez se realiza la petición AJAX al servidor para que los empleados puedan verlo desde la plataforma de administración. En el siguiente apartado veremos cómo se implementaría esa parte del proceso de pedidos.

Empleados

Ya hemos visto como un cliente puede realizar pedidos, ahora veremos cómo los empleados pueden ver estos y editar su estado. Al igual que los clientes con los productos, primero se debe listar todos los pedidos mediante una petición a la API, vamos a ver como se ha implementado esta.

```php
<?php
include "../funciones/inicioSesion.php";
if($id_usuario <0){
      echo "-1";
      die;
}
$array = array();
$i = 0;
$filtro_empleado = "";
if($_POST["id_empleado"] > 0){
  $filtro_empleado = " AND d.id_usuario = $_POST[id_empleado]";
}

$consulta = sqli("select a.id as id, b.nombre as producto, a.estado as estado, a.notas as notas,
a.fecha_inicio as fecha_inicio, a.extras as extras,
                  a.fecha_fin, a.id_producto, d.nombre as nombre_mesa, d.codigo as codigo_mesa, a.id_sesion
                  from pedidos a LEFT JOIN productos b ON a.id_producto = b.id
                  LEFT JOIN sesiones c ON a.id_sesion = c.id
                  LEFT JOIN mesas d ON c.id_mesa = d.id
                  WHERE c.estado = 1 AND c.estado = 1 $filtro_empleado order by id desc ");
while($pedido = mysqli_fetch_array($consulta)){
  $array[$i] = new stdClass();
    $array[$i]->id = $pedido['id'];
  $array[$i]->producto = $pedido['producto'];
  $array[$i]->estado = $pedido['estado'];
  $array[$i]->notas = $pedido['notas'];
  $array[$i]->id_sesion = $pedido['id_sesion'];

  $hoy = new DateTime("now");
  $datetime1 = new DateTime($pedido['fecha_inicio']);
  $interval = $datetime1->diff($hoy);
  $array[$i]->diff_minutos = $interval->format('%i');
  $array[$i]->diff_horas = $interval->format('%H');


  $array[$i]->fecha_inicio = $pedido['fecha_inicio'];
  $array[$i]->fech_fin = $pedido['fech_fin'];
  if($pedido['extras']) $array[$i]->extras = obtenerExtras($pedido['extras']);
  else $array[$i]->extras = "";
  $array[$i]->id_producto = $pedido['id_producto'];
  $array[$i]->nombre_mesa = $pedido['nombre_mesa'];
  $array[$i]->codigo_mesa = $pedido['codigo_mesa'];
  $i++;

}
$respuesta = new stdClass();
$respuesta->hash =md5(json_encode($array)) ;
if($respuesta->hash != $_POST["hash"]){
  $respuesta->array = $array;

}

echo json_encode($respuesta, JSON_UNESCAPED_UNICODE);
```

Figura 4.19 API para obtener pedidos

Al igual que al obtener productos, se obtienen todos los pedidos (con la posibilidad por filtrar por usuario) mediante una petición SELECT a la base de datos y se envía la respuesta en formato JSON.

```
{
  "array": [
    {
      "id": "361",
      "producto": "Spaghettis Carne",
      "estado": "1",
      "notas": "",
      "id_sesion": "91",
      "diff_minutos": "0",
      "diff_horas": "00",
      "fecha_inicio": "2020-04-14 13:58:05",
      "fech_fin": null,
      "extras": [],
      "id_producto": "23",
      "nombre_mesa": "Mesita",
      "codigo_mesa": "13"
    }
  ]
}
```


Figura 4.20 Json de ejemplo al obtener pedidos

Este es un ejemplo de respuesta con la lista de pedidos, en este caso solo hay un pedido. Analizando el pedido podemos ver que el cliente no ha pedido extras o que se ha realizado 14 minutos antes de la petición.

Al igual que en la plataforma de clientes, se utilizan componentes React para renderizar la información de los pedidos. Estos componentes son más complejos que los anteriores por eso se va a mostrar una versión simplificada, para verlos completos están situados en fichero **/admin/js/script_estado_pedidos.js** del proyecto.

```
class TablaPedidos extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      lista_pedidos : [],
      hash : "",
      estado : 0,
      id_empleado : 0
    }
  }


  render() {
    const listPedidos = lista_pedidos.map((mesa) =>
      <Pedido info={mesa} padre= {this}/>
    );
    return <div className="text-center">
              <div style={{width: "100%"}} >

              {listPedidos}
              </div>
            </div>
  }
}
```

```
class Pedido extends React.Component {
  constructor(props) {
    super(props)
  }

  render() {

      fila = <div className="card wow zoomIn"
                onClick ={(e) => acciones(this.props.info.id, this.props.info.estado)}>
              <div className="row" style={{padding:"10px"}}>
                <div className="col-6 text-left">
                  {this.props.info.nombre_mesa} - {this.props.info.codigo_mesa}
                </div>
                <div className="col-6 text-right">
                  <span className={clase_estado} >{estado} </span>
                </div>
                <div className="col-12 text-left">
                  <h3>{producto} <span style={{fontSize:"15px"}}> Hace {tiempo}</span></h3>
                </div>
                <div className="col-12 text-left" style={{paddingLeft:"100px"}}>
                  Notas: {this.props.info.notas}
                </div>
                <hr />
                <div className="col-12 text-left" style={{paddingLeft:"100px"}}>
                  Extras: {extras}
                </div>
                <div className="col-12 text-center" style={{paddingLeft:"100px"}}>
                  <a href={sesion_href} className="btn btn-secondary" >Mesa</a>
                </div>
              </div>
      </div>
    }
    return <div> {fila} </div>
  }
}
```

Figura 4.21 Componentes React para visualizar pedidos

Al igual que en el ejemplo de la carta tenemos un componente denominado TablaPedidos que se encarga de listar pedidos, y otro llamado Pedido, que se encarga de mostrar la información de un determinado pedido y permitir que un empleado pueda editar su estado. Recordemos que un pedido puede tener 5 estados distintos: realizado, en preparación, preparado, entregado o cancelado. Si el empleado modifica el estado se realiza una llamada a la API, es este caso a **editar_estado_pedido**.

```php
<?php
include "../funciones/inicioSesion.php";
if($id_usuario <0){
      echo "-1";
      die;
}

$consulta = sqli("select id, id_producto, id_sesion from pedidos WHERE id = $_POST[id_pedido];");
if($pedido = mysqli_fetch_array($consulta)){
    $cambioFecha = ", fecha_fin = NULL";
    if($_POST["estado"] == 4){
      $cambioFecha = ", fecha_fin = NOW()";
    }
    sqli("UPDATE pedidos SET estado = $_POST[estado] $cambioFecha
          WHERE id = $_POST[id_pedido];");

    if($pedido["id_producto"] < -1 && $_POST["estado"] == 4){
      // SE HA PAGADO LA CUENTA
      $consulta2 = sqli("select id, id_mesa  from sesiones WHERE id = $pedido[id_sesion];");
      if($sesion = mysqli_fetch_array($consulta2)){
        sqli("UPDATE sesiones SET estado = 2, fecha_fin = NOW()
        WHERE estado = 1 AND id = $pedido[id_sesion]");
        sqli("UPDATE mesas SET estado = 4 WHERE  id = $sesion[id_mesa]");
      }

    }
}
$respuesta = new stdClass();

echo json_encode($respuesta);
```

Figura 4.22 API para editar el estado de un pedido

En el fragmento de código podemos ver como se edita el estado del pedido solicitado. En el caso de que se marque como entregado se actualizará su fecha de entrega a la actual. Por otro lado, los pedidos con un identificador de producto menor a -1 serán solicitudes de cuenta, por lo que si estos finalizan la mesa deja de estar disponible.

4.3.6 Estadísticas

La última funcionalidad de la que hablaremos será la de mostrar estadísticas. Como se ha visto con anterioridad, una vez se obtienen los datos de los pedidos realizados deseamos ver estadísticas con el fin de analizar cómo va el establecimiento.

Vamos a ver tres partes diferentes en el apartado de estadísticas: Sesiones, Facturación y Productos más pedidos.

Sesiones

En este subapartado se mostrarán todas las sesiones. Como hemos visto antes, una sesión contiene todos los pedidos que realice un cliente en una mesa. Al igual que con los pedidos utilizaremos un componente de React para visualizar estas.

```
class Sesiones extends React.Component {
  constructor(props) {
    super(props)
  }
  render(){
      var lista = this.props.info.map((sesion) =>
      <div className="card" style={{paddingLeft:"10px"}}>
        <div className="row">
          <div className="col-8 text-left">
            <p>Fecha inicio:{sesion.fecha_inicio}</p>
            <p>Fecha fin:{sesion.fecha_fin}</p>
          </div>
          <div className="col-4 text-left">
            <p style={{bottom:"0px"}}>{sesion.precioTotal} €</p>
          </div>
        </div>
      </div>
    )
    return lista
  }
}
```

Figura 4.23 Componente React para visualizar una sesión

Este componente mostrará una lista de todas las sesiones con los datos que la caracterizan.

Facturación

Por otro lado se desea saber cuánto se ha facturado cada día, para ello se mostrará una gráfica con la suma de las sesiones de cada día, semana,mes o año (En función de lo que se desee).

Para realizar lo anteriormente mencionado primero se deberá hacer una petición a la API **obtener_facturación** con los parámetros que deseemos para indicar el intervalo de tiempo que deseamos . Esta se puede encontrar en /admin/api/obtener_facturacion.php

Una vez tenemos la información el componente de React mostrará los datos en pantalla. Veamos una versión simplificada de este.

```
class Facturacion extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      lista_valores :[],
      lista_fechas :[]

    }
  }
  render(){


    if(document.getElementById('myChart')){
      $('#myChart').remove();
      $('#chart-container').append('<canvas id="myChart"><canvas>');
      var ctx = document.getElementById('myChart').getContext('2d');

      var chart = new Chart(ctx, {
          type: 'line',
          data: {
              labels: this.state.lista_fechas,
              datasets: [{
                  label: '',
                  backgroundColor: 'rgb(255, 99, 132)',
                  borderColor: 'rgb(255, 99, 132)',
                  data:  this.state.lista_valores
              }]
          },
          // Configuration options go here
          options: {
            scales: {yAxes: [{ticks: {beginAtZero: true}}]},
            tooltips: {callbacks: {label: (item) => `${item.yLabel} €`,},}
          }
      });
    }

    return <div>
    Fecha Inicio:
    <input id="fecha_inicio" type="date" className="form-control" />
    Fecha Fin:
    <input id="fecha_fin" type="date" className="form-control" />
    <br / >
    Intervalo
    <select id="opcion" className="form-control" >
      <option value="0"> 1 Día</option>
      <option value="1"> 1 Semana</option>
      <option value="2"> 1 Mes</option>
      <option value="3"> 1 Año</option>
    </select>
    <input type="button" className="btn btn-info" value="Filtrar"
      onClick = {(e) => this.obtenerFacturacion()}/>
    <div id="chart-container">
    <canvas id="myChart"></canvas>
    </div>
    </div>
  }

}
```

Figura 4.24 Componente React para visualizar la gráfica de facturación

El componente que se muestra en la imagen anterior utiliza la librería de ChartJS para visualizar en pantalla la gráfica con la facturación de los períodos indicados.

Productos más vendidos

Por último, se desea ver cuales son los productos más vendidos, o lo que es lo mismo cuales son los productos que tienen asociados más pedidos. Para realizar esto también vamos a utilizar un componente React.

```
class Productos_count extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      lista_valores :[],
      lista_productos :[]

    }
  }
  render(){
    if(document.getElementById('myChart')){
      $('#myChart').remove();
      $('#chart-container').append('<canvas id="myChart"><canvas>');
      new Chart(document.getElementById("myChart"), {
      type: 'bar',
      data: {
        labels: this.state.lista_productos,
        datasets: [
          {
            label: "Número de pedidos",
            backgroundColor: ["#E5BB33", "#C0C0C0","#964B00"],
            data: this.state.lista_valores
          }
        ]
      },
      options: {
        legend: { display: false },
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true
                }
            }]
        }
      }
      });
    }
    return <div>
    <div id="chart-container">
    <canvas id="myChart"></canvas>
    </div>
    </div>
  }

}
```

Figura 4.25 Componente React para visualizar gráfica con los productos más solicitados

Vemos que, al igual que en el caso de la facturación, se utiliza la librería ChartJS para visualizar la gráfica que, esta vez contendrá una lista con los productos más pedidos

Una vez visto cómo se han desarrollado las funcionalidades más importantes vamos a ver el resultado de la implementación.

4.3 Resultado

Visto el desarrollo vamos a ver el resultado obtenido, en primer lugar veremos la plataforma que verán los clientes, y por otro la plataforma de administración que verán los empleados. Se ha creado una web basada en un restaurante italiano y se han creado productos para poder ver como funcionaría la plataforma. Se puede acceder a esta web mediante el dominio https://www.tfg.carlosh.es/

4.3.1 Plataforma de clientes

En primer lugar se ha creado una página web de presentación del restaurante. Esta no accederá a la base de datos, por lo que será una página web estática. Y por otro lado, otra página que mostrará la carta del restaurante sin necesidad de iniciar sesión en la plataforma de pedidos.

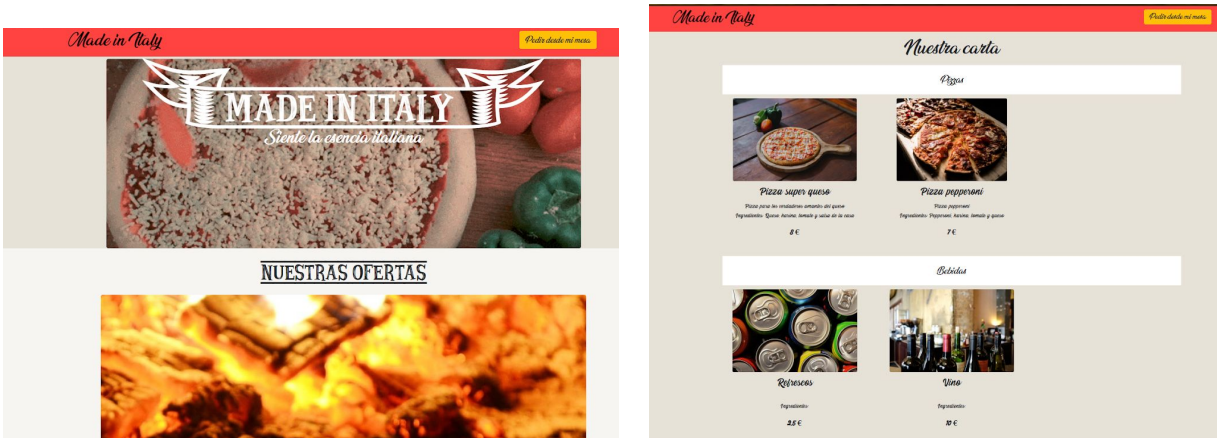

Figura 4.26 Página inicial del restaurante

En las imágenes anteriores vemos las páginas anteriores. Estas son las que vería un cliente antes de iniciar sesión. Se puede ver que en la parte superior derecha hay un botón para entrar en la plataforma. Al pulsar este llegaremos a la siguiente web.

Figura 4.27 Inicio de sesión a la plataforma de pedidos

Figura 4.28 Código QR que quivalente a: https://www.tfg.carlosh.es/app/index.php?codigo=13&clave=1YFUI

En la primera imagen podemos ver el formulario con el cual podemos entrar a la plataforma y así pedir desde la mesa. También en es posible entrar en la mesa utilizando un código QR y así iniciar sesión de manera más rápida. Al escanear el código se accederá a la plataforma de pedidos pasando como parametros GET el código y la clave de la mesa.

Una vez el cliente entre con sus credenciales verá lo siguiente:

Figura 4.29 Menú inicial de la plataforma de pedidos

Por un lado vemos la lista de productos disponibles para pedir y por otro un menú en el cual se recoge el historial de pedidos. Al seleccionar un producto aparecerá un formulario con el que el usuario podrá personalizar su pedido.

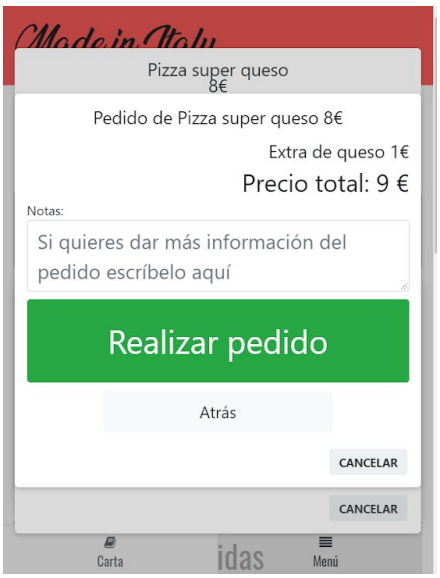

Figura 4.30 Modal para realizar pedido

Y, una vez realizado el pedido se mostrará en el menú, junto con su estado actual.

Figura 4.31 Evolución de estado de los pedidos

En las imágenes anteriores vemos la evolución del pedido desde que el cliente lo pide hasta que se le entrega a la mesa.

Una vez que el cliente ha realizado todos los pedidos este puede solicitar la cuenta.

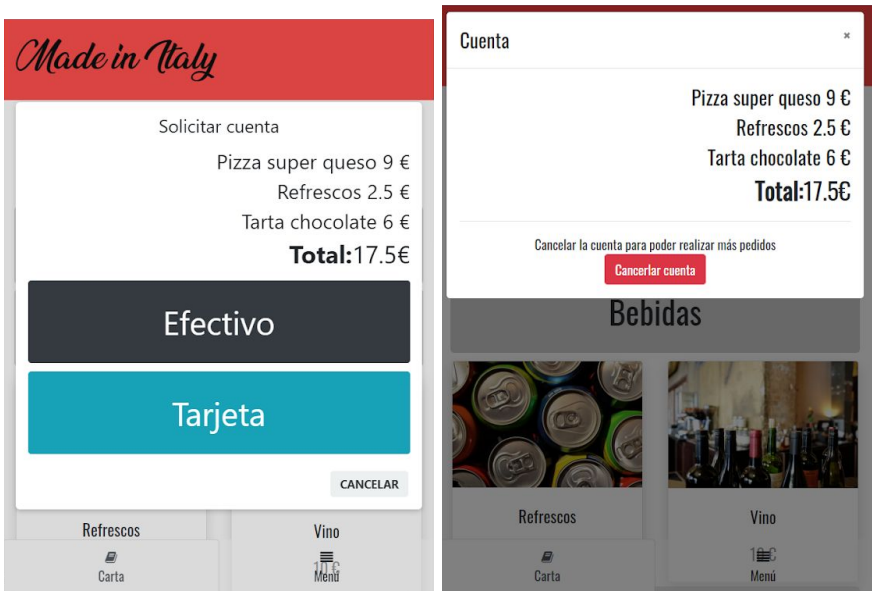

Figura 4.32 Solicitud de cuenta

En la primera imagen se muestra el formulario para indicar si se desea pagar en efectivo o mediante tarjeta. Una vez solicitada la cuenta, la plataforma se bloqueará para que no se puedan realizar más pedidos. Existe un botón para cancelar la cuenta y así poder realizar más pedidos en el caso de que se desee.

Ya hemos visto la plataforma de clientes, a continuación veremos las diferentes funcionalidades implementadas en la plataforma de empleados.

4.3.2 Plataforma de empleados

En este apartado vamos a ver como ha quedado la plataforma de empleados después de desarrollar todas las funcionalidades vistas anteriormente.

La primera página que se visualiza es el formulario de inicio, en el que el empleado introducirá su nombre y contraseña. Una vez se introducen las credenciales se muestra un menú con todas las opciones disponibles.

Figura 4.33 Inicio de sesión de la plataforma de empleados y menú inicial

Las opciones disponibles están separadas en tres secciones, por un lado las que configuran elementos de la plataforma, segundo lugar las que sirven para consultar y interactuar en tiempo real con los pedidos y mesas y por último encontramos el apartado de estadísticas.

Configuración

En primer lugar veremos los apartados de la plataforma que tienen relación la configuración de los diferentes elementos de la plataforma. Estos son: Productos, Categorías, Mesas y Usuarios.

La primera sección que vamos analizar es la de productos.

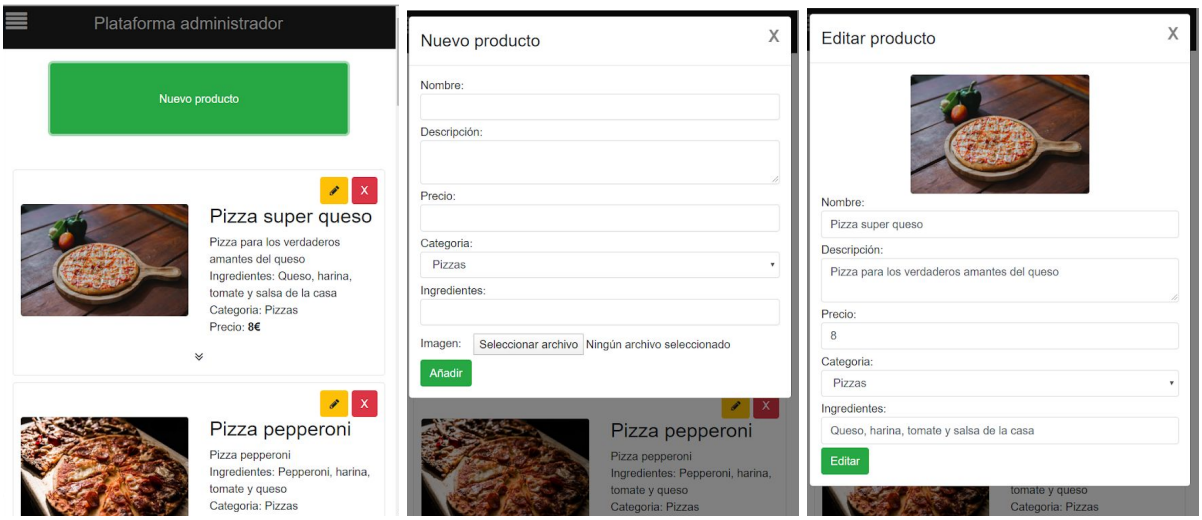

Figura 4.34 Sección de productos

Al seleccionar esta opción veremos la lista de todos los productos, y habrá un botón para abrir un modal que nos permitirá añadir nuevos productos a la plataforma. En cada producto hay un botón para poder editar los campos del producto y por otro lado otro botón para eliminarlo de la lista.

En segundo lugar está las categorías, como hemos visto en apartados anteriores estas nos permitirán agrupar los diferentes productos de la carta.

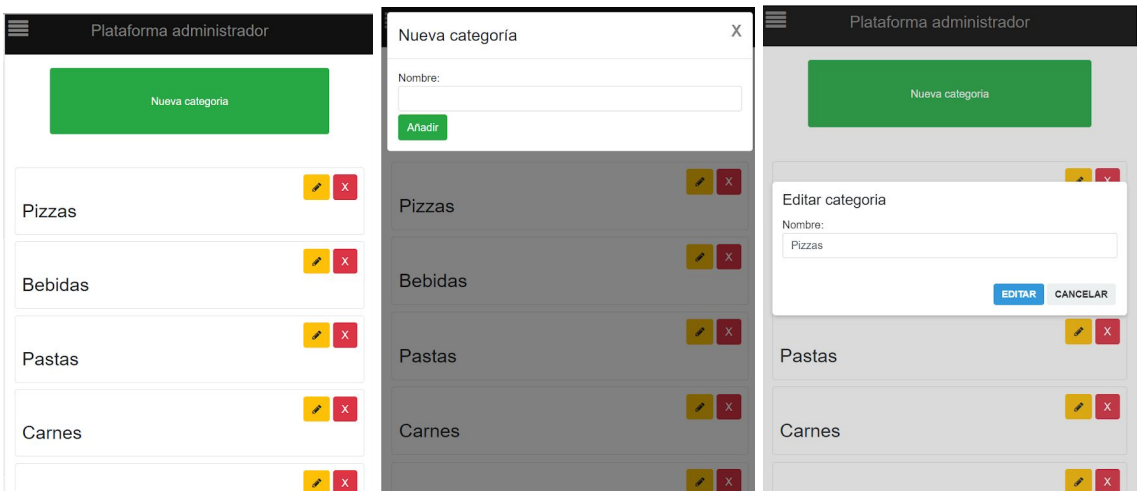

Figura 4.35 Sección de categorías

Al igual que en los productos podremos visualizar la lista con todas las categorías, añadir una nueva y editar o eliminar las existentes.

En cuanto a las mesas encontramos lo mismo, pero esta vez los campos a completar son el nombre y el código de la mesa.

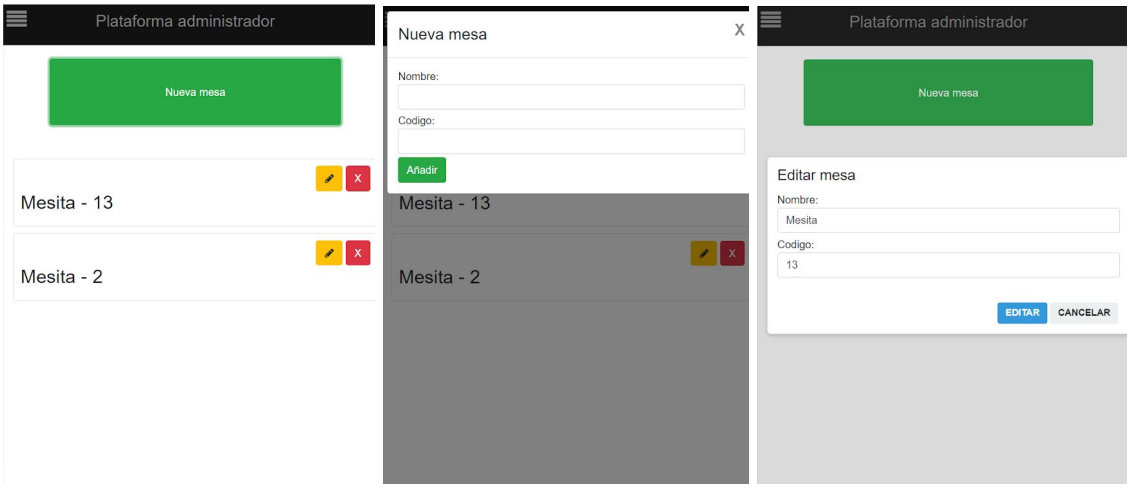

Figura 4.36 Sección de mesas

Y por último está la sección para administrar los usuarios. En esta se mostrará la lista de usuarios con la posibilidad de editar el usuario con el que se ha iniciado la sesión, y, en caso de ser administrador eliminar otros usuarios.

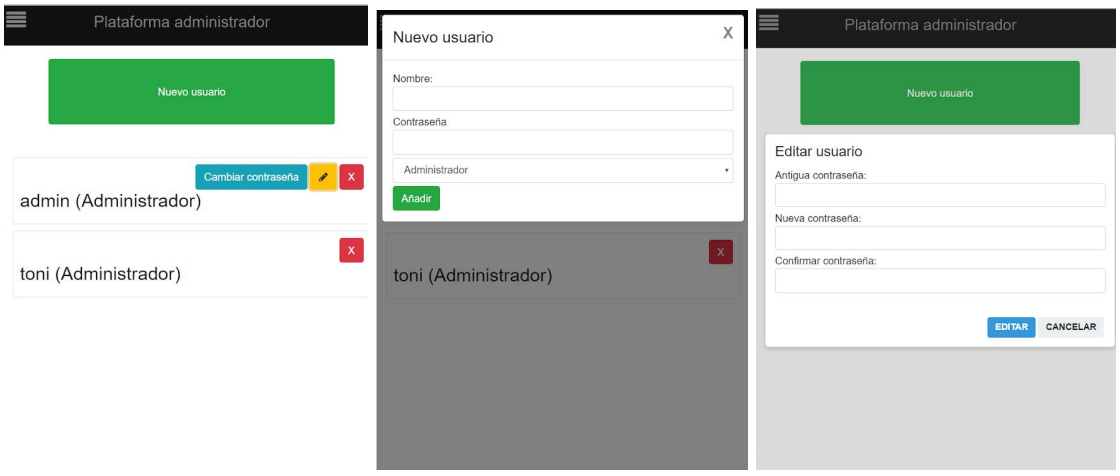

Figura 4.37 Sección de usuarios

Al igual que en las secciones anteriores se podrán crear nuevos usuarios en la plataforma, esta vez indicado los permisos que tendrá este nuevo usuario.

Plataforma en tiempo real

Las siguientes secciones son en las que los empleados interactúan con la plataforma en tiempo real, y así conocer los pedidos que tienen que preparar o entregar.

La primera sección de la que vamos a hablar es la que administra el estado de las mesas. En esta sección se podrá ver que estado tiene cada mesa así como modificar estos en función de cuando llegue un cliente al restaurante o se vaya.

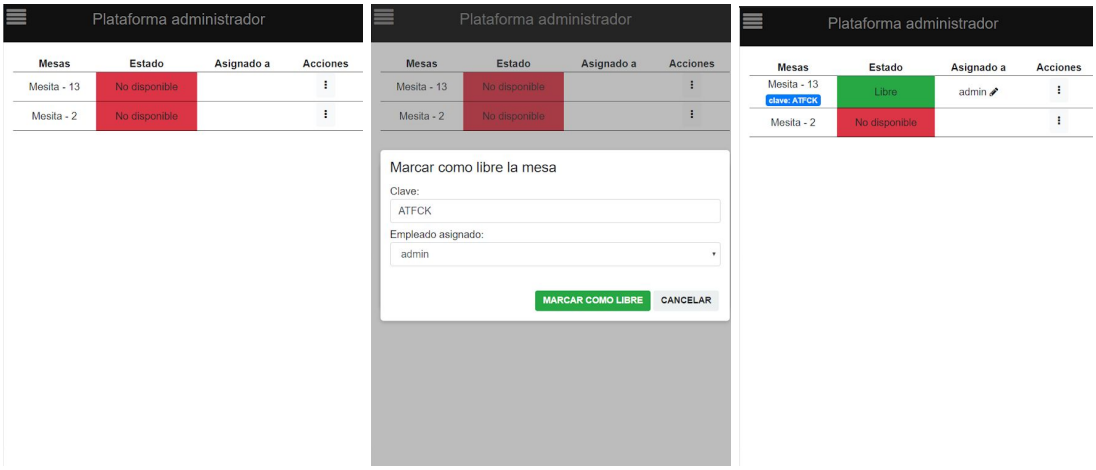

Figura 4.38 Sección de visualización de mesas

En las imágenes anteriores podemos ver cómo se visualizan todas las mesas, y, al pulsar sobre las opciones de una de ellas podemos marcarla como libre para que así los clientes puedan realizar pedidos con el código y la clave de la mesa. En el formulario para marcar como libre podemos ver como se pide la clave (la cual se genera automáticamente pero permite que el empleado la modifique) y el empleado asignado a esa mesa.

Por otro lado encontramos la sección para administrar los pedidos, como hemos visto antes esta se actualiza automáticamente mediante peticiones ajax para conocer en tiempo real los pedidos realizados por los clientes y el estado en el que están estos.

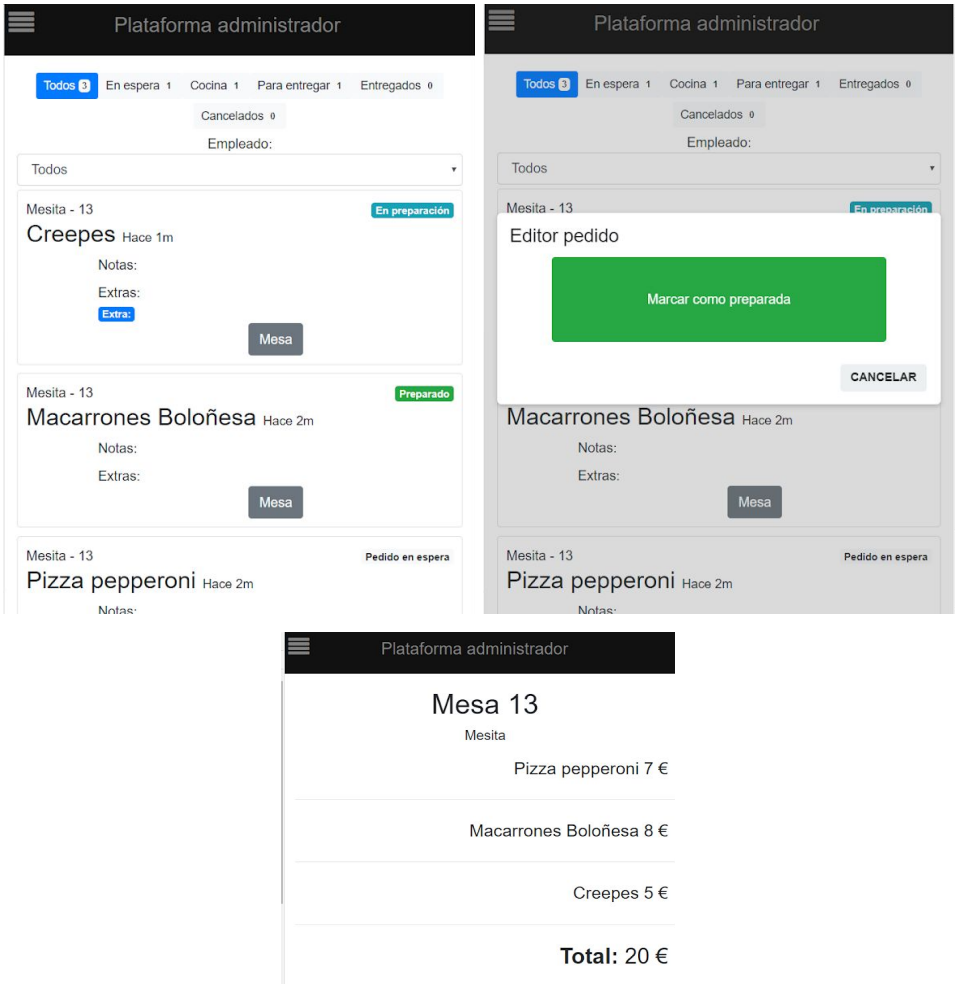

Figura 4.39 Sección de visualización de pedidos

En la primera imagen podemos ver cómo se pueden visualizar todos los pedidos realizados, con la posibilidad de filtrar en función del estado en el que esté en ese momento y el empleado al que esté asignado esa mesa. Al pulsar sobre uno de los pedidos podemos editar el estado de este. También se ha implementado una opción para visualizar todos los pedidos asociados a una mesa.

Estadísticas

Por último tenemos la sección de estadísticas. Como se ha descrito anteriormente existen tres estadísticas que vamos a tener en cuenta. En primer lugar está el historial de sesiones.

Figura 4.40 Sesiones históricas del restaurante

Aquí se mostrarán todas las sesiones de compra de las mesas con la fecha a la que se ha iniciado y a la que se ha finalizado junto al precio total de la sesión.

Por otro lado está la gráfica de facturación, en la cual se podrá ver la evolución de la facturación del restaurante, mediante los filtros podremos establecer el intervalo que queremos analizar y la separación entre cada dato, este puede ser diario, semanal, mensual o anual.

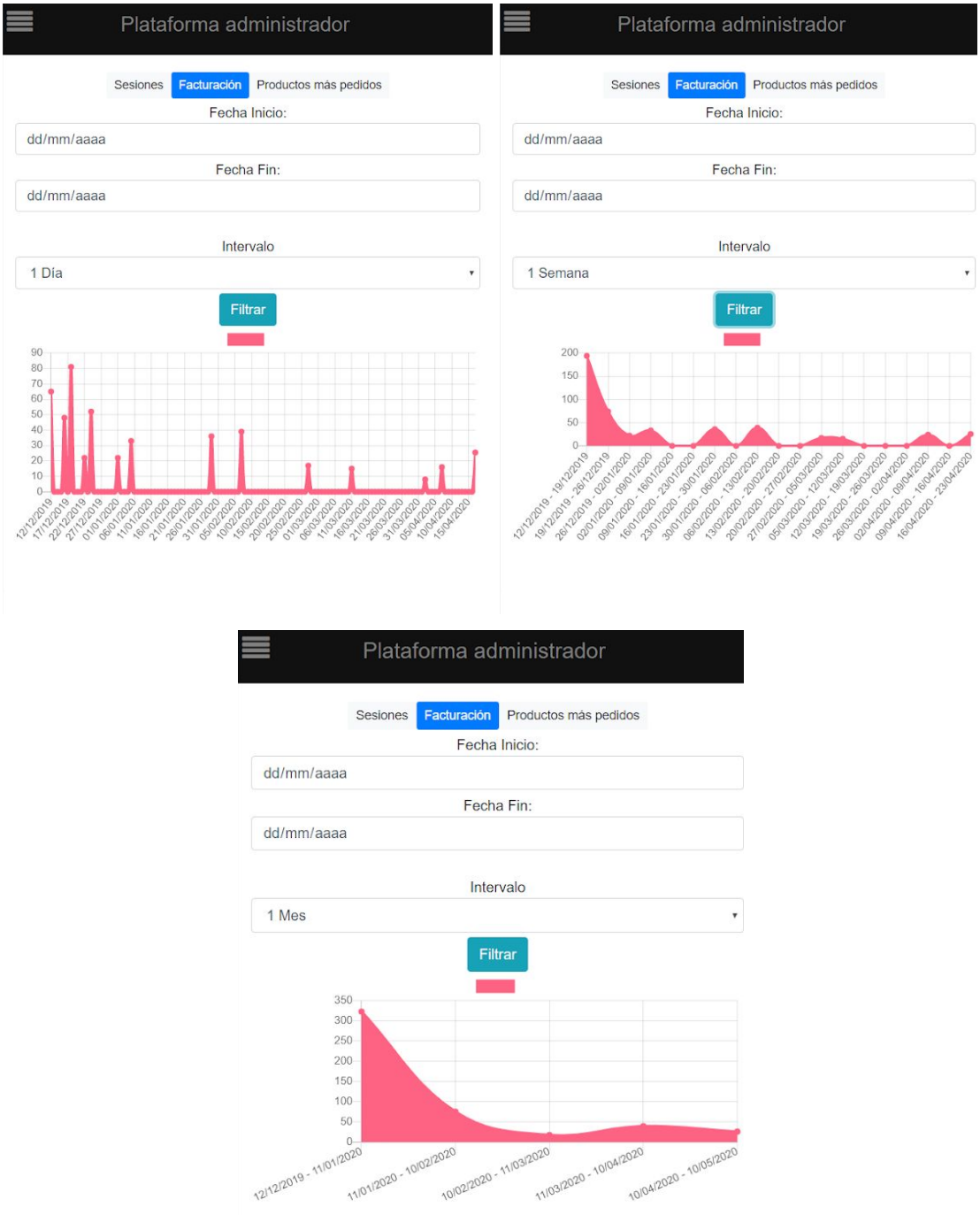

Figura 4.41 Gráficas de facturación

En la imágenes anteriores podemos ver las gráficas que se generan en función de los parámetros aportados. Como no se han especificados las fechas de inicio y de fin la

gráfica abarcará desde el primer pedido hasta el último. En las gráficas de la figura 4.41 podemos ver los pedidos de prueba realizadas durante las pruebas de desarrollo.

Por último tenemos la gráfica que muestra los productos más vendidos del restaurante.

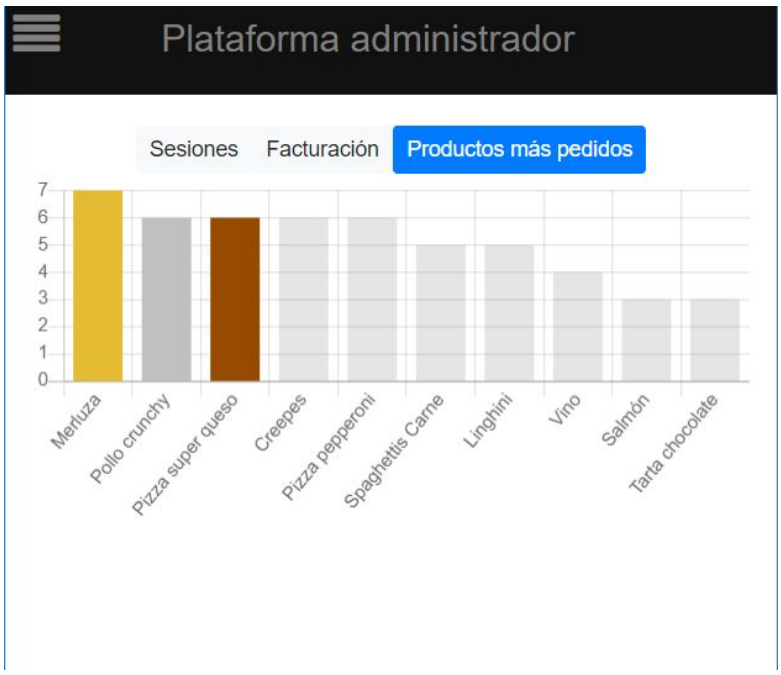

Figura 4.42 Productos más vendidos

Como hemos visto en el apartado de desarrollo, en esta sección se mostrarán los productos que tienen más pedidos relacionados.

5 Conclusiones

En el proyecto presentado se ha mostrado como se puede diseñar y desarrollar una plataforma para realizar pedidos en un restaurante mediante una arquitectura distribuida de tipo cliente-servidor. En este caso,para la implementación se ha optado por utilizar el conjunto de tecnologías LAMP pero también se podrían utilizar otras como Ruby on Rail, Django o MEAN(Mongodb, Express Js, Angular y Node Js). La metodología para realizar este proyecto se puede aplicar para cualquier otro proyecto de naturaleza similar.

A nivel personal, ya estaba familiarizado con las tecnologías mencionadas pero nunca había realizado un proyecto de estas dimensiones. Esto me ha hecho entender más a profundidad cómo utilizarlas correctamente y a resolver problemas que surgen durante el desarrollo.

5.1 Posibles nuevas funcionalidades

A pesar de que la plataforma desarrollada presenta todas las funcionalidades descritas sería posible añadir más para mejorar esta. Una de las partes menos pulidas de la aplicación es el diseño. Esto se ha dejado un poco de lado en la implementación. Para mejorar esto lo ideal sería la colaboración de un diseñador de interfaces para hacer la plataforma más profesional y más atractiva para el cliente. Otra de las funcionalidades que se podrían añadir sería la de permitir varios idiomas y así internacionalizar la plataforma para llegar a más clientes potenciales. Y por último también se podría añadir una sección para realizar pedidos a domicilio, y, en el caso de que el restaurante estuviese interesado en esto, tenga unificado todo en la misma plataforma.

6 Bibliografía

Estado del arte

[1] Estudio sector de hostelería en España

https://www.hosteltur.com/133119_el-sector-de-la-hosteleria-modera-su-crecimiento-hasta-el-24.html

[2] Estudio comida rápida en España

https://www.foodretail.es/retailers/delivery-comida-rapida-ventas-2019-observatorio-dbk_0_1428457147.html

[3] Estudio uso de plataformas de delivery
https://www.ocu.org/organizacion/prensa/notas-de-prensa/2019/appscomida051119

[4] Plataforma Maitre Serie 4 Menu

https://astarteinformatica.com/programa-restaurante-maitre-serie-4-menu.php?menu=2

[5] Plataforma EtreCartas https://www.entrecartas.com/

Implementación

[6] Libro PHP y MYSQL (3º ED.) Oliver Heurtel ISBN: 9782409008047

[7] Documentación PHP https://www.php.net/docs.php

[8] Documentación Apache https://httpd.apache.org/docs/2.0/

[9] Documentación React https://es.reactjs.org/docs/getting-started.html

[10] Documentación Bootstrap https://getbootstrap.com/docs/4.1/

[11] Documetación MariaDB https://mariadb.com/kb/en/documentation/

[12} Ejemplos Chart JS https://www.chartjs.org/samples/latest/

[13] Jquery Confirm https://craftpip.github.io/jquery-confirm/

[14] CertBot https://certbot.eff.org/

7 Anexos

[1] Código fuente de la plataforma en la carpeta “codigo_fuente”

[2] Configuración de la base de datos en el archivo “base_datos.sql