TRABAJO FIN DE CICLO

DESARROLLO DE APLICACIÓN PARA GESTIÓN DE RESTAURANTES Y NEGOCIOS DE HOSTELERÍA

DESARROLLO DE APLICACIONES WEB

CURSO 2021/2022

AUTOR: BORJA RODRÍGUEZ RUIZ

TUTOR/A: MARÍA TERESA MARTÍNEZ SORIA

10 de enero 2022

AGRADECIMIENTOS

Me gustaría agradecer este trabajo en primer lugar a mis compañeros de ciclo, ya que es muy complicado poder tener un apoyo cuando se estudia a distancia, y ellos lo han sido en todo momento que la necesidad ha apremiado.

En segundo lugar, a la gente que tengo más cercana, mis amigos, mi familia y sobre todo mi pareja, por aguantar y soportar dos años y medio de enclaustramiento para que todo saliera adelante y siempre con una sonrisa en la cara.

Y, por último, y no menos importante, a mi saboteador interno, esa voz que me lleva diciendo todo este tiempo que no soy capaz y que no merece la pena, porque gracias a él he podido levantarme y demostrar que se equivocaba.

RESUMEN

Este trabajo de fin de ciclo de Formación Profesional aborda el desarrollo de una aplicación web que sirva de herramienta para la gestión de negocios de hostelería, y que pueda favorecerles a través de la integración de un sistema de gestión sencillo, interactivo, atractivo y de fácil comprensión.

En la actualidad las aplicaciones existentes o bien no cumplen con los requisitos mínimos para la tarea, dando como resultado la necesidad de tener numerosas aplicaciones diferentes para una misma meta, o bien no disponen de un diseño llamativo que facilite la visualización de esta. Por ello, para la consecución de la misma, se ha empleado un lenguaje de programación orientado a objetos como base, reuniendo en esta todos los elementos que intervienen como objetos, y herramientas de diseño de interfaces que mejoran la apariencia y aspecto visual.

El principal objetivo de la aplicación, como se nombra anteriormente, es dar solución a la gestión de restaurantes y aunar en una sola todas las herramientas disponibles actualmente para simplificar el trabajo, desde la entrada del cliente al local hasta su marcha.

ABSTRACT

This final grade work is about the development of a web application that work as a tool for hostelry business management, and it could make it easy to this work by the integration of a simple, interactive, attractive and easily understandable system.

Nowadays, the existing applications or doesn't accomplish the minimum requirements for the task, giving as a result the need to have different applications for just one goal, or it doesn't have an attractive design which make it easy the visualization of itself. For this reason, for getting this, it has been used an object-oriented programming language as a base, recognizing every element of the process as an object, and interface design tools which improves the appearance and the visual aspect.

The mean objective of this application, as it is said before, is to give a solution for restaurants management and get together all the tools possible in one to simplify the work, since the customer get inside the local until this one leave it.

ÍNDICE

1. INTRODUCCIÓN

Con este proyecto se ha planteado la idea de la realización de una aplicación web disponible para todo el espectro de locales de hostelería que quieran acceder, mediante el cual podrán tener recogidas en una sola aplicación muchas de las funciones que actualmente se encuentran disponibles en diferentes herramientas, aunándolas en una sola, como son:

- Gestión de clientes y mesas, desde la llegada de los comensales hasta su cierre de cuenta y marcha.
- Gestión de facturas, incluyendo creación, visualización, impresión en PDF, envío por mail y eliminación.
- Gestión de platos de carta, mediante el cual los camareros y/o responsables del local tendrán acceso al listado de los platos disponibles y sus precios.
- Gestión de reservas, creación y eliminación de las mismas.
- Cierre del día, por el cual se podrán ver un total de los balances realizados por el local durante el día.

De igual forma y para facilitar el trabajo a los clientes de la aplicación, se ha trabajado en una aplicación visualmente atractiva que facilita tanto su uso como su aprendizaje, y un diseño responsivo que puede ayudar al trabajo en equipo de los equipos de servicio. Para todo esto se han utilizado las tecnologías más actuales y un lenguaje de programación orientado a objetos que facilita la comprensión de la actividad que realiza la aplicación a la vez que mejora el entendimiento del código generado siendo más sencilla la actualización del mismo y la mejora de posibles futuros errores.

2. JUSTIFICACIÓN.

Este proyecto surge del estudio detallado del funcionamiento interno y procesual de los locales dedicados a la hostelería, desde grandes empresas a pequeñas.

Teniendo como base el conocimiento de cómo funciona, poniendo de ejemplo, un restaurante, se ha procedido a analizar el funcionamiento del día a día de este trabajo, y a dar solución a los problemas e inconvenientes que suelen darse, siempre desde el punto de vista de los trabajadores que emplean esas herramientas a diario como puede ser el visionado de las reservas del día de manera sencilla para poder preparar mesas, o la visualización simple de las mesas existentes y de los comensales que hay en cada una para llevar un conteo exacto del aforo del local.

Todo este análisis lleva a la necesidad de facilitar a estas empresas un trabajo que es indispensable para el buen funcionamiento de sus negocios, ya que gracias a la suma de todas las capacidades de la aplicación no es necesario el conocimiento profundo de diversas aplicaciones, ahorrando tiempo y dinero a las empresas, dando como resultado la creación de esta aplicación web de gestión de restaurantes.

3. OBJETIVOS

A continuación, se exponen el listado de objetivos que se han planteado para el desarrollo y finalización del proyecto.

- Diseñar una aplicación sencilla, de fácil comprensión, y basada en los principios de UX/UI.
- Desarrollar un diseño que apoye al fácil aprendizaje de funcionamiento de la herramienta.
- Obtener cobertura total con el cómputo de capacidades que puede tener la aplicación para el trabajo.
- Usar un lenguaje de programación eficaz para el óptimo funcionamiento de la aplicación, y hacer de esta una herramienta eficaz y veloz.
- Evolucionar la metodología de trabajo actual de los locales de hostelería a una forma de trabajo online actualizándolos.

4. MÉTODO

4.1 METODOLOGÍA DE DESARROLLO

Para la adecuada visualización del desarrollo de este proyecto se separa en dos apartados diferentes todo lo relativo al desarrollo visual del desarrollo estructural, tratando en el primero todo lo referente a los lenguajes y estilos que se han empleado para el desarrollo visual y en el segundo lo relativo al lenguaje de programación usado y a la metodología de desarrollo del mismo.

4.1.1 Desarrollo visual

El apartado visual de la aplicación se encuentra diseñado, distribuido y maquetado con HTML5 para la composición del esqueleto de la web, CSS3 para la parte de estilos de la página, ayudado por la librería Sass (librería que ayuda a la mejora y legibilidad de las páginas CSS, permitiendo el anidamiento de etiquetas, creación de métodos y creación de variables, la cual una vez se ejecuta transforma todo su contenido en hojas de estilo legibles por un procesador CSS) y de la biblioteca multiplataforma Bootstrap para dar más versatilidad a la web.

Para la validación de formularios y las escucha de eventos se ha utilizado JQuery, una potente librería de JavaScript que facilita el trabajo que realiza JavaScript en los navegadores. Se han validado todos los formularios de la web no solo a través de las etiquetas correspondientes de HTML como pueden ser "*required"* o los tipos de inputs (*text, password, email*), sino que se ha validado la entrada de información con los eventos de teclado y a su vez se ha validado al enviar los formularios. Debido a que JavaScript es un complemento del que se puede prescindir, o bien se puede modificar en el inspeccionador de código que ofrecen los navegadores, se ha añadido un tercer nivel

de cotejamiento de datos y filtrado por medio de lenguaje PHP, desarrollado en el siguiente apartado.

4.1.2. Desarrollo estructural

Se ha optado por el uso del lenguaje PHP por su gran potencia, alcance y versatilidad, y la modularización del código con una estructura de modelo vista-controlador a nivel básico, abstrayendo la idea principal de esta metodología y a través de la cual hemos dividido el proyecto en tres capas bien diferenciadas que llamaremos a partir de ahora vistas, modelos y controladores.

En la capa de vistas dispondremos de todo aquello que muestre contenido visual y lo formatee para su lectura posterior. En este proyecto concreto algunas de las páginas que harían la función de vista son "*historialFactura.php"*, que en este caso muestra un histórico de las últimas 10 facturas con la posibilidad de aumentar de diez en diez el número total de facturas visualizadas, o la misma página principal, "*main.php",* la cual muestra las mesas ocupadas en el restaurante e incluye un formulario de adición de mesas, páginas que se encuentra en la carpeta raíz del proyecto.

En la segunda capa, la de controladores, se ejecutan todas las páginas que hacen de conexión o puente entre la vista y los modelos, de tal forma que las vistas realizan la consulta a estas páginas y son estas las encargadas de realizar una conexión a los modelos. Todas ellas se encuentran agrupadas en una carpeta a la que se ha llamado "*utils"*.

Por último, una vez efectuada la conexión con los modelos, son estos los encargados de comunicarse con la base de datos, ejecutar las consultas pertinentes y devolver un resultado o ejecutar una modificación sustancial en esta. Todos los modelos de esta aplicación se han trabajado desde la programación orientada a objetos (POO), teniendo en cuenta cada elemento que participa dentro del flujo de información de la aplicación como objetos de la vida real contabilizables,

y es por ello que todos se encuentran agrupados en la carpeta "*clases"* del proyecto.

Aparte de estos objetos se ha trabajado con el patrón de trabajo DAO (Data Access Object), que es considerado como un objeto más que se encargara de las consultas a base de datos que generan vistas nuevas (consultas *select*) mientras que el resto de objetos trabajan sus propios métodos de inserción, modificación y eliminación de datos cuando es necesario.

Como se ha comentado anteriormente, además de todo esto se ha procedido a realizar un filtrado de los parámetros pasados por los envíos de formularios con la función de PHP "*filter_var()"* que ofrece la posibilidad de analizar lo recogido por url en las variables y eliminar todas las infiltraciones y código malicioso que pueda entrar a la web a través de estas, eliminado la posibilidad de daños a la base de datos o robo de información.

4.2 PLAN DE PROYECTO

Para el desarrollo programado de la aplicación se ha seguido un plan de proyecto diferenciando los siguientes pasos:

- Definición y diseño de diagrama de casos de uso.
- Definición de diagrama UML o diagrama de clases.
- Diseño y creación de bases de datos.
- Diseño del apartado visual por medio de un mock-up simple de la estructura, así como un proto boceto del logotipo.
- Desarrollo de código dividido en subfases:
    - Conexión a base de datos
    - Registro de restaurante
    - Inicio de sesión y recuperación de contraseña
    - Listado de mesas.
    - Listado de cuentas.

    - Listado de facturas.
    - Listado de platos.
    - Listado de reservas.
    - Información del restaurante.
    - Cierre del día.
- Realización de pruebas.
- Consecución de resultados y análisis.
- Despliegue.

4.3. TECNOLOGÍAS EMPLEADAS

Se precede a continuación a enumerar y describir las tecnologías empleadas para la consecución del proyecto:

- **PHP (Hypertext Preprocessor)** como lenguaje de programación principal. Se ha decidido utilizar este debido a su gran versatilidad y funcionalidades, con la capacidad de trabajar con orientación a objetos para ofrecer modularidad al código, y siendo uno de los lenguajes de programación más extendidos y usados por lo que tiene una comunidad muy amplia.
- **XAMPP** como entorno de desarrollo que nos ofrece la posibilidad de trabajar en loca con un servidor apache remoto, PHP y bases de datos.
- **JQuery,** como librería externa de JavaScript, que ha sido empleada para facilitar el trabajo con lenguaje JavaScript simplificando la manera de interactuar con los elementos del DOM y ofreciendo la posibilidad de trabajar con AJAX (Asynchronous JavaScript and XML).
- **Sass**, extensión del lenguaje CSS que sirve de apoyo para el desarrollo de las hojas de estilo de la aplicación, permitiendo un trabajo más ágil y con las mismas posibilidades que el propio de CSS, ya que permite la creación de funciones, variables y el *nesting* o anidamiento de código.

- **Bootstrap,** la librería HTML, CSS Y JavaScript por excelencia que sirve para amoldar el trabajo y la maquetación con un sistema de rejilla, todo ello por medio de clases CSS preestablecidas en la misma, pudiendo modificar tamaños, tipografías, o colores añadiendo clases al elemento deseado.
- **Librería JSPDF** para generación de documentos PDF, empleada en la emisión de facturas e impresión de las mismas.
- **Librería SweetAlert**, complemento de JavaScript que embellece las alertas JavaScript dándole un toque distinguido.
- **Librería TableExport** para la generación de tablas en formato *.xlsx* como en nuestro caso la salida de los cierres de día.
- **SonarLint** extensión del IDE empleado que sirve para corrección de errores, pruebas, evasión de redundancia de código y mejor de calidad del código desarrollado.

4.4. DESARROLLO

4.4.1 Casos de uso

El primer paso que se lleva a cabo es la realización del diagrama de casos de uso. Este diagrama nos informará del camino a seguir que se realizará al comienzo de la utilización de la aplicación, cuando finaliza, y los procesos que se pueden desarrollar durante este proceso.

Según el análisis del diagrama (Imagen 1), se puede observar que el recorrido principal que se sigue es en primer lugar el actor principal es el restaurante que desea hacer uso de la aplicación, el cual necesita registrarse para el empleo de la misma. Una vez registrado, este actor da comienzo al inicio de sesión, el cual ofrece la posibilidad de añadir mesa, o generar una reserva. En caso de añadir una mesa, existe la posibilidad de eliminarla, o bien crear una cuenta, esta última incluye la acción evidente del pago de cuenta y cierre de la misma, debido a que, si se inicia una cuenta, siempre se acabará

pagando. El siguiente paso sería, indistintamente de si ha habido mesas o reservas, el cierre de día, lo que llevaría directos a una finalización de la aplicación.

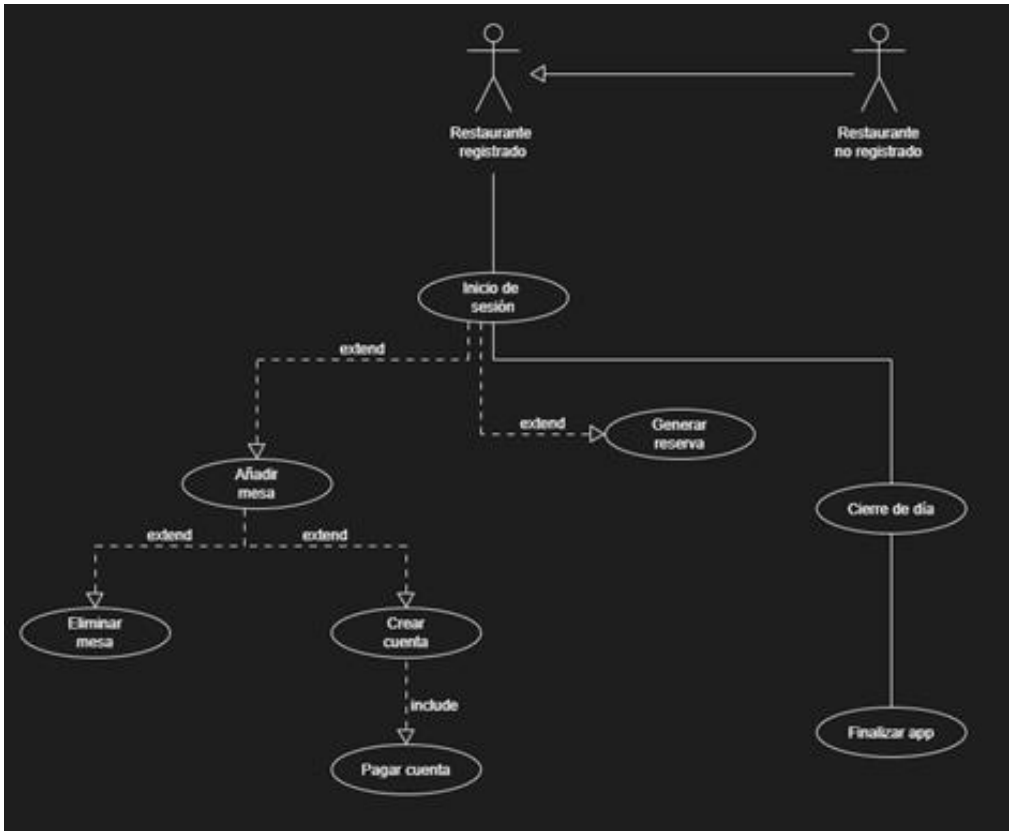


Imagen 1. Diagrama de casos de uso

4.4.2. Diseño orientado a objetos

Tras el diseño del diagrama de casos de uso, se procede a la realización del diagrama UML o de clases, mediante el cual se establecen las clases principales de la aplicación, sus atributos, la accesibilidad de los mismos, y los métodos que tendrán para el intercambio y modificación de información de los atributos.

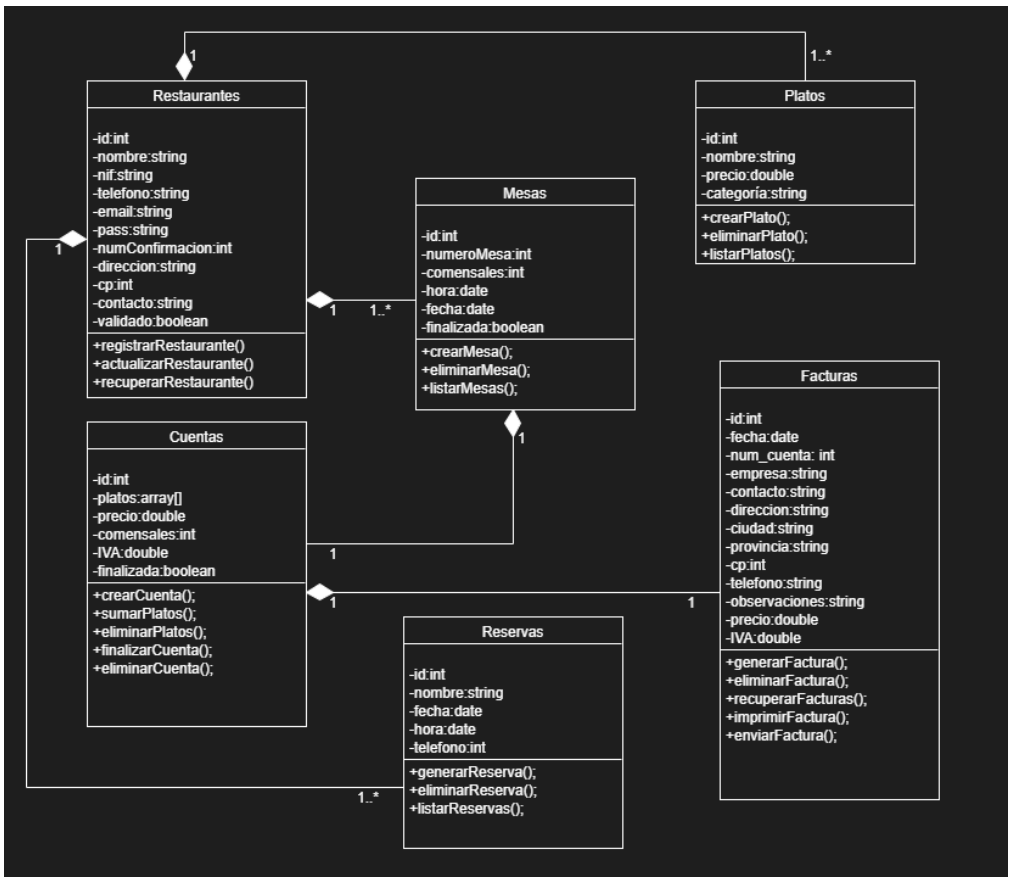


Imagen 2. Diagrama UML de clases

En la anterior imagen podemos observar tanto las clases resultantes como la relación que existe entre ellas. Se pasa a detallar cada una de ellas:

- Restaurantes. Clase principal de la aplicación. Esta clase representa a los restaurantes registrados, y tiene todos sus atributos privados, siendo estos ID, nombre, NIF, teléfono, email, contraseña, número de confirmación, dirección, código postal, contacto y por último el atributo “validado”. Los dos atributos que merece la pena explicar son el número de confirmación, que es un número de 4 cifras que hace las veces de contraseña, y la cual es necesaria para los procesos de eliminación de mesas, facturas, platos y reservas, y el atributo de validado, que se ha representado como un valor booleano que indica si el restaurante ha iniciado sesión por primera vez o no en la aplicación. Los métodos que la caracterizan son *registrarRestaurante(),* que se

encarga de generar un nuevo restaurante, y *actualizarRestaurante()*, que modifica los datos del restaurante seleccionado.

- Mesas. Esta clase representa todas las mesas que existen en el restaurante relacionado en el momento activas. Sus atributos, todos ellos privados, serian un ID, el número de la mesa, los comensales, la hora de entrada, la fecha actual y su finalización. Esta última valida si la mesa sigue activa, o si ha finalizado ya. Los métodos que tiene son *crearMesa(),* que inicializa una mesa nueva, *eliminarMesa(),* que elimina una mesa inicializada, y *finalizarMesa(),* que finaliza la mesa que ha terminado.
- Cuentas. Esta clase es considerada como correlativa a la clase mesas, ya que al crearse un elemento de la clase mesas, se crea automáticamente un objeto de la clase cuentas (no existe una mesa sin cuenta, aunque este vacía), y dispone de los atributos privados ID, platos, precio, número de comensales y finalizada. Al igual que el atributo de “mesas”, este dato “finalizada” es un dato booleano que se modifica al modificarse el del objeto “mesas” correspondiente. El atributo platos ha sido considerado como un array de objetos, el cual se rellenará con objetos de la clase platos. Los métodos que tiene son *crearCuenta(), eliminarCuenta()* y *finalizarCuenta()*, que se ejecuta al inicializar una mesa, eliminarla o finalizarla, y *sumarPlatos()* y *eliminarPlatos()*, que añade o elimina platos del array de platos.
- Facturas. Esta clase es de vital importancia para una de las funciones principales de la aplicación, que es la de generar las facturas correspondientes a las cuentas finalizadas. De entre sus atributos, todos privados, encontramos ID, fecha de generación de factura, número de cuenta correspondiente, nombre de la empresa, contacto, dirección, ciudad, provincia, código postal, teléfono, observaciones en caso de que fuera necesario, precio final de la factura, e IVA. Sus métodos son *generarFactura()* y *eliminarFactura()*, que crea o destruye facturas.

- Platos. Al igual que facturas, esta clase se relaciona directamente con la clase restaurantes y con la clase cuentas, concretamente con uno de sus atributos, platos. Esta clase contiene los siguientes atributos privados, ID, nombre del plato, precio y categoría. Sus métodos principales son *crearPlato()* y *eliminarPlato()*.
- Reservas. Esta clase, dependiente de la clase restaurantes, viene a ser las reservas que tiene el restaurante durante el día. Entre sus atributos privados encontramos ID, nombre de la reserva, fecha de la reserva, hora, y teléfono de contacto. Sus métodos principales son *crearReserva()* y *eliminarReserva()*.

En lo relativo a las relaciones entre ellas, encontramos las siguientes relaciones:

- Restaurantes – Platos (*1 – 1..**). Esta relación indica que, por cada restaurante, habrá 1 o muchos platos.
- Restaurantes – Mesas (*1 – 1..**). Esta relación indica que, por cada restaurante, habrá 1 o muchas mesas.
- Mesas – Cuentas (*1 – 1*). Esta relación indica que, por cada mesa solo podrá existir una cuenta.
- Cuentas – Facturas (*1 - 1*). Esta relación indica que, por cada cuenta únicamente existirá una factura.
- Restaurantes – Reservas (*1 – 0..**). Con esta relación se entiende que por cada restaurante existirán de 0 a muchas reservas.

Aparte de todo lo anteriormente nombrado, cabe recalcar que debido a la metodología de trabajo que se sigue existe una clase extra, a la que se ha llamado DAO (*Data Access Object*), la cual es un cómputo de métodos que sirven para el acceso a base de datos, extrayendo estas funciones directamente de las clases principales para abstenerlas de tener conexión con la información almacenada. Esta clase, llamada intuitivamente como “Métodos” contiene las siguientes funciones:

- validarInicioSesion(): recibe como parámetros un email y una

contraseña, las cuales coteja con la base de datos y comprueba que tanto mail como restaurante sean correctos, validando el inicio de sesión.

- recuperarPass(): recibe como parámetro el email al cual llegará un correo electrónico con un enlace que redirige, en función al email, a un formulario de cambio de contraseña.
- nuevaPass(): junto a la función anterior, recibe como parámetro el id del restaurante y la nueva contraseña, y valida el cambio de contraseña nuevo.
- arreglarMesasNoFinalizadas(): este método es considerado como auxiliar. Recibe como parámetros la fecha del día y el id del restaurante en cuestión, y, en caso de haberse finalizado la aplicación de manera errónea, y hubiera quedado alguna mesa abierta sin finalizar, este método finaliza las mesas no finalizadas con fecha anterior a la del día actual. Como se indica, no debería de ejecutarse, es meramente de control.
- validarCierreDia(): recibe como parámetro el id del restaurante, y en caso de quedar alguna mesa no finalizada, devuelve *false* no permitiendo el cierre del día.
- recuperarRestaurante(): recibe como parámetros un email y una contraseña, y coteja con la base de datos el restaurante en cuestión, devolviendo un objeto de la clase restaurante y almacenándolo como objeto en una variable.
- listarMesas(): recibe como parámetro el id del restaurante y lista las mesas no finalizadas del restaurante para mostrarlas.
- localizarMesa(): recibe como parámetros el número de mesa, el número de comensales, la fecha, la hora y el ID del restaurante, para localizar y cotejar en la base de datos la mesa en concreto y poder atribuir al objeto que se ha creado de la misma un ID.
- listarCuentasNoFinalizadas(): recibe como parámetro el ID del restaurante, y lista el total de cuentas no finalizadas que existen en el

momento según el ID de restaurante.

- listarCuentasNoFinalizadas(): al igual que la anterior, recibe como parámetro el ID del restaurante, y lista el total de cuentas finalizadas que existen en el momento según el ID de restaurante. Estas cuentas podrán optar a generar una factura.
- localizarCuenta(): recibe como parámetros el número de mesa y el ID del restaurante, y coteja con la base de datos localizando la cuenta correspondiente a la mesa que se le pasa, estando esta no finalizada.
- localizarCuentaFinalizada(): igual que la anterior función, a diferencia de que localiza cuentas finalizadas.
- listarReservas(): recibe como parámetros la fecha solicitada y el ID del restaurante, y lista en total de reservas existentes para ese restaurante en la fecha solicitada, mostrándolas.
- contarReservas(): recibe como parámetro el ID del restaurante, y realiza un conteo de las reservas realizadas en la aplicación para el mismo día.
- listarFacturas(): recibe como parámetros el ID del restaurante un número entero al que se llamara límite. Esta función coteja en la base de datos y devuelve un listado de las facturas de ese restaurante de más nueva a más antigua, con un límite igual al número que se le pasa por parámetro, que en este caso es diez. Al hacer clic sobre el botón “mostrar más” de la aplicación el límite se aumentará este límite, mostrando más.
- localizarFacturas(): recibe como parámetro el número de factura y el ID de restaurante, y localiza la factura concreta mostrando sus atributos.
- enviarFactura(): esta función recibe como parámetros el número de factura, el cuerpo del mensaje y un email válido. Con esta, se envía la factura correspondiente al número recibido, formateada en una variable de texto como tabla al correo electrónico indicado.
- listarPlato(): recibe como parámetro el ID del restaurante, listando el total de platos que tiene este en base de datos y devolviendo un array

con todos ellos.

- localizarPlato(): recibe como parámetros el ID del plato seleccionado y el id del restaurante, y localiza dicho plato devolviendo un elemento de la tabla “platos”.
- datosCierre(): recibe como parámetro el ID de restaurante, y realiza un conteo en la base de datos de mesas atendidas, clientes totales, número total de facturas generadas y la suma total de dinero obtenido.
- platosCierre(): recibe como parámetro el ID del restaurante, y localiza de todas las mesas del día actual los platos, devolviendo un array. Cabe decir que lo que se almacena en la base de datos es un array de objetos serializados, codificados en base64, por tanto, lo que devuelve es un array de *strings* que deben ser decodificados de base64 y deserializados. Esto será explicado más adelante.
- generarListadoDiscriminativo(): Como se ha explicado, una vez decodificados y deserializados se almacenan todos los platos vendidos durante el día en un array. Este array es pasado como parámetro de esta función junto al ID del restaurante, y añade uno a uno todos los platos a una tabla, para después contabilizar la suma total de platos de un mismo tipo vendidos y generar un array con este conteo. Acto seguido se elimina de esta tabla toda esta información, ya que no se volverá a utilizar y en caso de generar un segundo cierre del día podría causar error. El nombre de esta función es determinado por el array discriminativo que se genera de los sumatorios de platos.

4.4.3. Diseño de base de datos

El diseño de las tablas de bases de datos se realiza en su correspondiente diagrama entidad-relación (imagen 3).

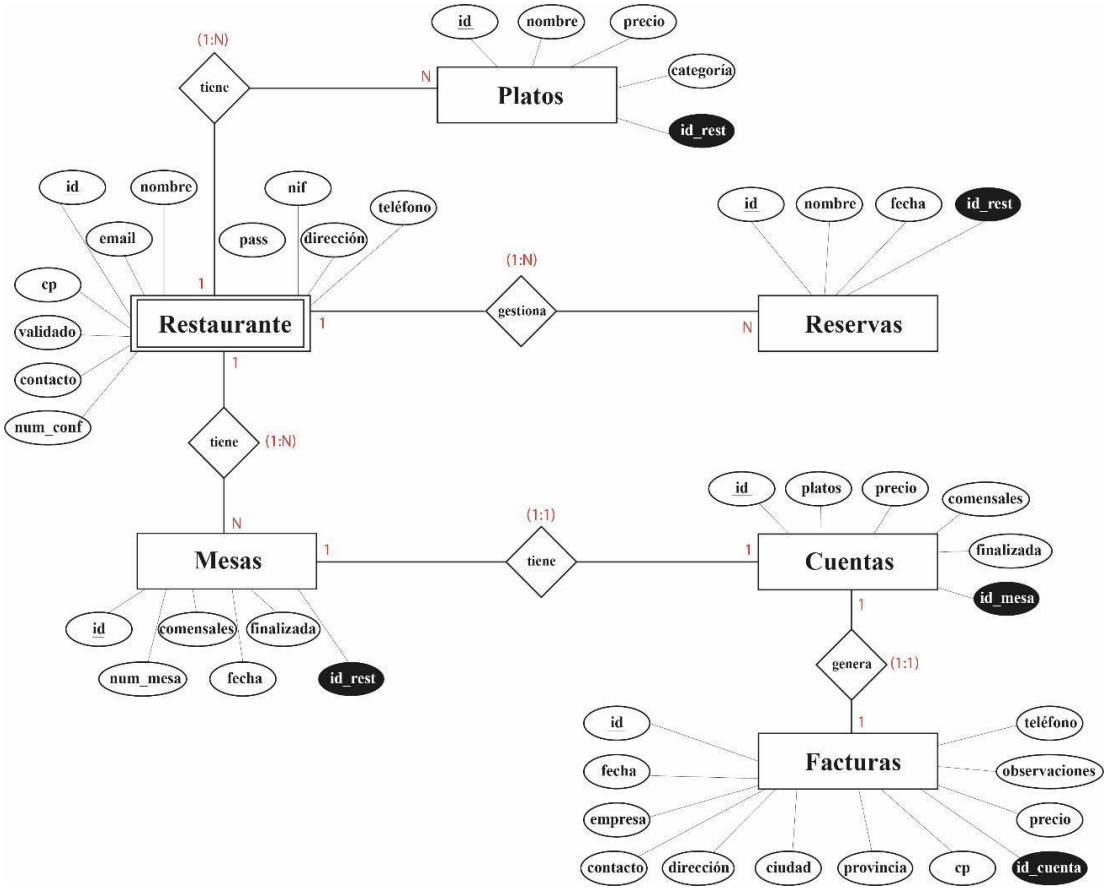


Imagen 3. Diagrama entidad-relación

Al igual que se vio en el diagrama de clases, con el análisis de este se puede ver que la entidad “Restaurante” es la entidad fuerte, ya que todas las demás dependen de la existencia de esta y si dejara de existir, el resto de ellas lo harían también. Por otro lado, se les ha dado el atributo “id” único en cada entidad como identificador y clave primaria, mientras que todas las entidades débiles tienen como clave foránea el ID de la tabla de la que dependen, como es el caso del ID de restaurante en las entidades “Mesas”, “Platos” y “Reservas”, el ID de la mesa a la que corresponde la cuenta en la entidad “Cuentas” y el ID de la cuenta correspondiente en la entidad “Facturas”.

Una vez identificadas las entidades, se procede a la descripción detallada de cada entidad como una tabla en base de datos, y de sus atributos como columnas en cada tabla (imagen 4).

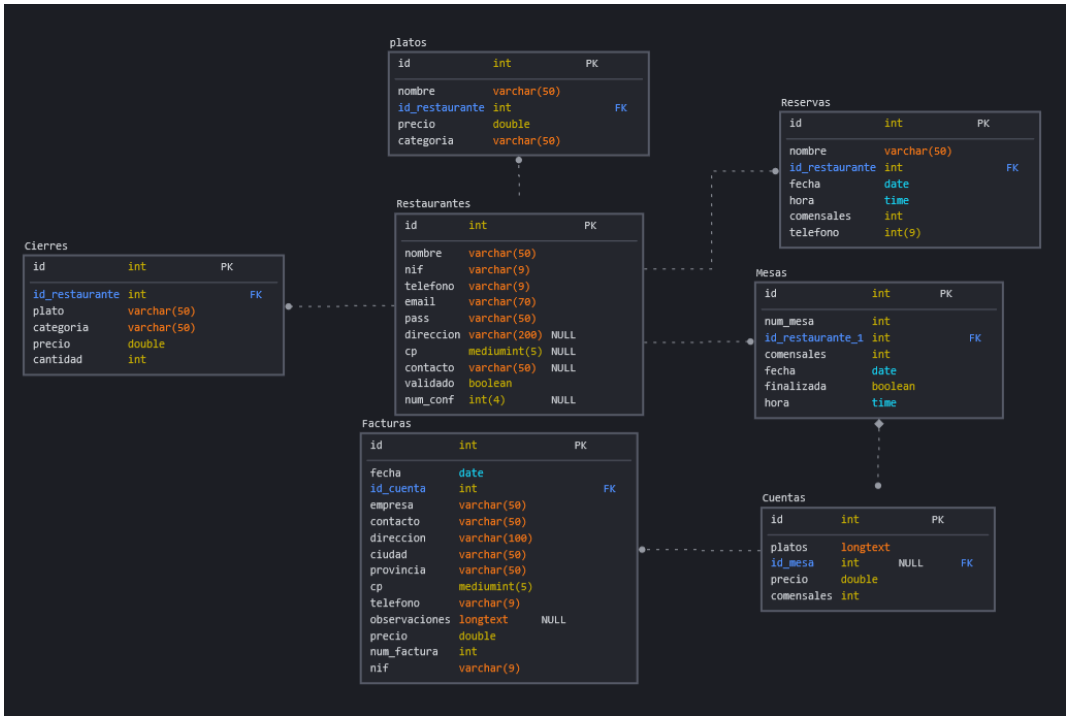


Imagen 4. Despliegue y desarrollo de tablas de base de datos

Las tablas “Cuentas” y “Facturas” tienen un eliminado y un modificado recursivo dependiente de la tabla la cual les da la clave foránea, esto viene a significar que, si un elemento de la tabla “Mesas” es eliminado, se eliminan automáticamente la cuenta asociada, y las posibles facturas que se hayan generado de esa mesa.

Como se explica más adelante, una mesa solo puede borrarse cuando aún no ha sido finalizada, por lo que no existirán facturas relativas a esa mesa. Esto se consigue añadiendo en la sentencia de creación de la tabla “ON UPDATE CASCADE, ON DELETE CASCADE”.

Ya configurado, estas han sido las líneas de código empleado para la creación de las tablas mediante lenguaje DDL (Data Definition Lenguage).

RESTAURANTES

```
CREATE TABLE restaurantes(
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(50) NOT NULL,
    NIF VARCHAR(9) UNIQUE NOT NULL,
    telefono VARCHAR(9) NOT NULL,
    email VARCHAR(70) UNIQUE NOT NULL,
    pass VARCHAR (50) NOT NULL,
    direccion VARCHAR(200),
    cp MEDIUMINT(5),
    contacto VARCHAR(50),
    validado BOOLEAN NOT NULL,
    num_conf INT(4) NOT NULL
);
```

Imagen 5. Creación de tabla "Restaurantes" en base de datos

MESAS

```
CREATE TABLE mesas(
    id INT PRIMARY KEY AUTO_INCREMENT,
    num_mesa INT NOT NULL,
    comensales INT NOT NULL,
    fecha DATE NOT NULL,
    hora TIME NOT NULL,
    finalizado BOOLEAN NOT NULL DEFAULT 0,
    id_restaurante INT NOT NULL,
    FOREIGN KEY (id_restaurante) REFERENCES restaurantes(id)
);
```

Imagen 6. Creación de tabla "Mesas" en base de datos

CUENTAS

```
CREATE TABLE cuentas(
    id INT PRIMARY KEY AUTO_INCREMENT,
    platos LONGTEXT,
    precio DOUBLE,
    comensales INT NOT NULL,
    id_mesa INT NOT NULL,
    FOREIGN KEY (id_mesa) REFERENCES mesas(id)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

Imagen 7. Creación de tabla "Cuentas" en base de datos

FACTURAS

```
CREATE TABLE facturas(
    id INT PRIMARY KEY AUTO_INCREMENT,
    num_factura INT NOT NULL,
    fecha DATE NOT NULL,
    empresa VARCHAR(50) NOT NULL,
    nif VARCHAR(9) NOT NULL,
    contacto VARCHAR(20) NOT NULL,
    direccion VARCHAR(100) NOT NULL,
    ciudad VARCHAR(40) NOT NULL,
    provincia VARCHAR(20) NOT NULL,
    cp INT(5) NOT NULL,
    telefono VARCHAR(9) NOT NULL,
    observaciones VARCHAR(400),
    precio DOUBLE NOT NULL,
    id_cuenta INT NOT NULL,
    FOREIGN KEY (id_cuenta) REFERENCES cuentas(id)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

Imagen 8. Creación de tabla "Facturas" en base de datos

PLATOS

```
CREATE TABLE platos(
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(50) NOT NULL,
    categoria VARCHAR(50) NOT NULL,
    precio DOUBLE NOT NULL,
    id_restaurante INT NOT NULL,
    FOREIGN KEY (id_restaurante) REFERENCES restaurantes(id)
);
```

Imagen 10. Creación de tabla "Platos" en base de datos

RESERVAS

```
CREATE TABLE reservas(
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(50) NOT NULL,
    comensales INT NOT NULL,
    fecha DATE NOT NULL,
    hora TIME NOT NULL,
    telefono INT(9) NOT NULL,
    id_restaurante INT NOT NULL,
    FOREIGN KEY (id_restaurante) REFERENCES restaurantes(id)
);
```

Imagen 9. Creación de tabla "Reservas" en base de datos

CIERRES

```
CREATE TABLE cierres(
    id INT PRIMARY KEY AUTO_INCREMENT,
    plato VARCHAR(50) NOT NULL,
    categoria VARCHAR(50) NOT NULL,
    precio DOUBLE NOT NULL,
    cantidad INT NOT NULL,
    id_restaurante INT NOT NULL,
    FOREIGN KEY (id_restaurante) REFERENCES restaurantes(id)
);
```

Imagen 11. Creación de tabla "Cierres" en base de datos

4.4.4. Guías de estilo

Los estilos básicos que se han dado a la aplicación vienen precedidos por un diseño *mockup* previo a la realización de la aplicación. Este diseño (Anexo 2. Mockup de la aplicación) se ha dividido en una parte de inicio de sesión y registro, y una parte interna con un menú lateral que incluye los apartados principales de la aplicación. Si bien el mockup es orientativo, la aplicación final ha resultado muy ligeramente diferente en diversos apartados, como son la visualización de las facturas, la impresión, o el cierre de día.

La gama de colores empleada está basada en amarillos, naranjas y rojos para los elementos más significativos, y colores oscuros para los fondos menos relevantes, detallando los lugares importantes.

Se ha empleado para las alertas la librería JavaScript SweetAlert, la cual embellece las alertas para hacerlas más amigables, y se ha aplicado un efecto desenfoque a las páginas principales al surgir una alerta para mantener el foco de atención en estas cuando es necesario.

Para la maquetación y diseño estructural se ha empleado Bootstrap, un sistema de rejillas para la colocación responsiva de los elementos de la web, dividiendo el total del espacio del navegador en columnas y rellenando con elementos estas a voluntad.

Para la definición de los estilos de color, texto y demás características visuales se ha empleado Sass como lenguaje complementario a CSS, estableciendo variables de color, tamaño o espaciado.

4.4.5. Modularización del código

Tal y como se ha nombrado anteriormente, al separar todo el proyecto entre modelos, vistas y controladores, el sistema de archivos, carpetas y subcarpetas tiene una separación similar (Imagen 12).

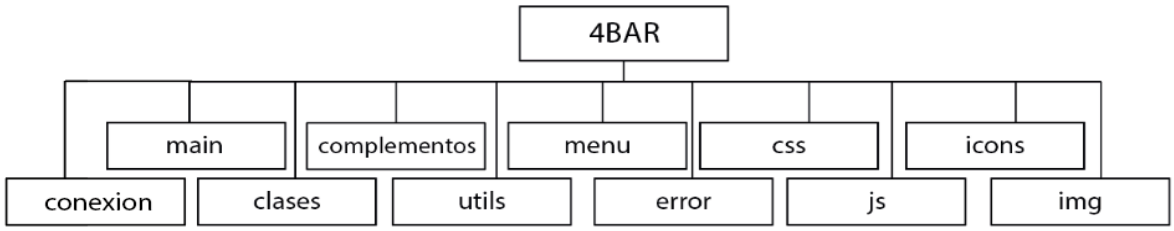


Imagen 12. Esquema de la división de carpetas y modularización de código

Partiendo de este esquema, se procede a explicar punto por punto cada apartado.

Conexión. Esta carpeta contiene el archivo conexión a la base de datos. Al igual que con el resto de ficheros se ha modularizado de esta manera para separarla del resto y ofrecer más seguridad, sobre todo con la conexión a la base de datos. Este fichero viene a ser una clase, de la cual se creará e instanciará cada vez que se requiera realizar una conexión la base de datos para visualización o modificación de datos.

Main. El apartado main contiene todos los ficheros que representa el apartado visual de la aplicación. Separado por código de validación se encuentra dividido en dos secciones, la sección de inicio de sesión, que es el apartado principal al que se accede a la aplicación, la cual estaría formada por el índex, registro de restaurantes y cambio de contraseña, y, por otro lado, la sección principal, que abarca el resto de archivos que constituyen los diferentes apartados de la aplicación, que son el listado de mesas, el listado de cuentas activas y finalizadas

(incluido la creación de una factura nueva), el listado de facturas generadas, el listado de platos, listado de reservas, visualización de ventas por cierre del día y la actualización de los datos del restaurante.

En esta última sección se han validado al inicio de cada documento, registrando únicamente si se inicia sesión de manera correcta el restaurante en una variable de sesión, de forma que, si se intentara acceder a cualquiera de las páginas por url de manera explícita, al no existir dicha variable de sesión se descarta la posibilidad de acceder, redirigiendo al usuario a la página de inicio de sesión.

Por último, contiene el documento polPrivacidad.php, documento que muestra la política de privacidad de la aplicación, ya que tiene datos personales no solo de los restaurantes sino de las reservas que se han generado por restaurante, y debe ser aceptada por el restaurante a la hora de registrarse.

Menú. En todas las páginas que componen la sección principal se ha incluido el menú por medio de un include, esto se ha hecho con la finalidad de, teniendo en cuenta las líneas futuras de trabajo, si se agregara alguna funcionalidad extra o si se cambiara alguna de las existentes, la cantidad de trabajo a realizar sería mucho menor, pasando de trabajar todos los documentos a un único documento que actualiza automáticamente el resto. Esta carpeta solamente contiene un fichero que es menú.php.

Complementos. Esta carpeta contiene las cabeceras de los ficheros del apartado main y todos aquellos complementos que, al igual que el menú, puedan sufrir cambios posteriores de contenido o funcionalidad. Dentro de ella se encuentran la cabecera de la parte del índex, la cabecera de la parte principal, y una hoja llamada seleccionHoras.php, que se encarga de generar diferentes opciones del input de selección de horas que se incluye en la realización de reservas, y por lo que, como solo se debe cargar en esta hoja (generarReserva.php) se le ha considerado un complemento.

Se han separado las cabeceras en dos debido a que tanto la sección de inicio como la principal tienen diferente composición estructural y de estilos, ambas cabeceras cargan diferentes hojas de estilos, títulos, etc. Por ello separarlo ha sido considerado la mejor opción para no cargar ficheros innecesarios.

Clases. Contiene todas las clases necesarias creadas para la aplicación. Cada clase se compone de unos atributos privados propios, identificados previamente en el diagrama de clases, para evitar el acceso a esos atributos de manera externa, un método constructor que inicializa los objetos que se generan a partir de estas clases, y los propios getters y setters de las clases, para el acceso y modificación de los atributos de ese objeto.

Aparte de los métodos genéricos que constituyen las clases, cada clase tiene un número de métodos propios diferentes que realizan las funciones de modificación, inserción y/o eliminación de tuplas en base de datos.

Se encuentra también en esta carpeta la clase DAO, explicada anteriormente, la cual es considerada en este proyecto como clase de apoyo, y cuando se cree un objeto de esta clase lo único a lo que tiene acceso son métodos estáticos a los cuales podrá acceder para el acceso a base de datos y creación de vistas de la información extraída. De esta forma se separa la vista de la información de la base de datos de los cambios efectivos que se puedan realizar en esta.

Utils. Esta carpeta contiene todos los archivos que sirven de enlace entre las vistas y los modelos, o como se les ha llamado antes, controladores. Todos ellos contienen, por normal general, las validaciones de los datos que reciben por url de formularios, y en casi de ser correcta esta validación redirigirán a la ejecución de un método concreto de un modelo o bien redirigirán a la página anterior con un mensaje de error a través de url.

Por ejemplo, el primer uso de esto en la aplicación, se corresponde con la realización de añadir una mesa al restaurante. A la

hora de realizar el formulario de añadir mesa, este redirige a una página llamada addMesa.php, la cual valida todos los datos enviados por url como son el número de mesa y el número de comensales entre otros necesarios, y si la validación (posterior al filtrado) es correcta, se crea un objeto de la clase mesa, que ejecutara el método crearMesa().

Estos archivos contienen al comienzo de su desarrollo una validación de variables recogidas por url por medio del método filter_var() de PHP, que dependiendo del tipo de variable realiza unas funciones u otras con el fin de eliminar todo código malicioso que pueda ser enviado por formularios por atacantes, cubriendo, junto al código JavaScript, el apartado de ciberseguridad que se ha realizado en este proyecto.

Error. A pesar de utilizar sentencias try{..} catch(){…} en los métodos de las clases para manejo de errores, se ha generado esta carpeta que contiene sentencias que emiten mensajes de error dependiendo del parámetro de error que se haya enviado por url. En resumidas cuentas, al ejecutar un método, pueden suceder dos escenarios, o bien todo sucede como está previsto, por lo que el mensaje de error no es tanto un mensaje de error si no un mensaje de aviso de éxito en la operación, o bien existe algún error o inconveniente en el proceso que lleva a cabo el método, por lo que dependiendo del error, se redirigirá hacia la página principal que carga estas hojas con un error en la url, y al recibirla, si está declarada esa variable por url emitirá un mensaje con el error correspondiente.

Los encargados de emitir estos errores y redirigir a las páginas principales son los controladores, una vez que reciben lo necesario de los métodos ejecutados por los modelos. Cabe señalar que se ha separado en cinco archivos los mensajes de error, según su uso en la aplicación, de tal forma que cuando carga el apartado de facturas, se cargan los mensajes de error de facturas, sin tener que cargar el resto de archivos evitando el aumento de volumen y disminuyendo la velocidad de ejecución, lo cual sucede igual con el apartado principal, los platos, las reservas o el inicio de sesión.

Css. Esta carpeta contiene todos los ficheros Css y Scss que se han empleado en la composición visual y de estilos de la aplicación. Al emplear Scss, los archivos sobre los que se trabajan tienen extensión .scss, y al ejecutarlos generan dos archivos nuevos, un archivo con extensión .css, que es el archivo que enlazaremos en las cabeceras correspondientes y que será el válido (HTML no es capaz de ejecutar e interpretar código scss debido a su nesting o anidamiento, por ello se transforma en css), y un archivo con extensión .map, que es el que otorga el mapeo de estilos y hace de puente en la transformación de los archivos anteriores.

Además, se encuentra en la carpeta el fichero _colores.css, el cual tiene un guion bajo delante. Este guion es el que evita que el motor de Scss genere un archivo válido de este fichero. Esto se realiza debido a que el fichero solamente contiene variables y funciones del propio lenguaje Scss, las cuales se van a importar a las hojas de trabajo incluyendo el código. Al igual que antes, se ha modularizado este código para facilitar el flujo de trabajo en líneas de trabajo posteriores y agilizar los cambios.

Js. Esta carpeta contiene, por un lado, los dos archivos JavaScript que sirven para validación de formularios en la aplicación, separados en introducción o inicio de sesión, y apartado principal (aunque la extensión es de JavaScript, el lenguaje empleado es el de JQuery, por facilidad, comodidad, y la rapidez que ofrece a la hora de programar), y por otro las dos librerías de JavaScript que se han empleado en este proyecto, una de ellas jsPDF, cuya utilidad es la de exportar un elemento del DOM a PDF, y 4BAR ejecuta esa librería al exportar las facturas a PDF, y la otra es TableExport, cuya utilidad es la de exportar una tabla del documento donde se encuentre a formato .xlsx u hoja de cálculos, que en este caso es ejecutada en el cierre de día para exportar la tabla resultante a un archivo Excel.

Img. Contiene todas las imágenes en formato png que han sido empleadas para la realización visual de la aplicación, como pueden ser el logo de la app, el favicon, el símbolo que representa a las mesas,

o las imágenes de fondo de los apartados (textura de pizarra e imagen de restaurante estándar).

Icons. Para el uso de símbolos e iconos como el de sumar, restar, finalizar mesa, o imprimir entre otros, se ha empleado una librería de iconos generada con la web www.icofont.com, que genera automáticamente un font-face con los elementos visuales seleccionados. Este font-face se realiza en un fichero .css que será añadido a las cabeceras y se usarán incluyéndolos en una etiqueta “i” de HTML con la clase correspondiente al icono. Por ejemplo, el icono de sumar plato del apartado menú, un círculo con un símbolo “+” en medio, se ha empleado el siguiente código:

```
<i class=”icofont-plus-circle”></i>
```

4.4.6. Detalles de desarrollo y proceso

Tal y como se plantea en el apartado 4.2 de este documento, “Plan de proyecto”, el desarrollo de este proyecto se ha programado en diferentes apartados siguiendo una estructura lógica de funcionamiento, que se desarrollarán a continuación con las partes más significativas en cuanto a programación y su proceso propio.

Conexión a base de datos.

El primer punto que se ha tratado ha sido la conexión con la base de datos. Se ha trabajado desde PDO, y al estar trabajando con XAMPP las bases de datos que se han utilizado son localhost, por lo que la conexión es relativamente sencilla. Se ha considerado la primera parte del trabajo y la principal para trabajar, ya que, sin esta, no podría realizarse ninguna tarde de inicio de sesión siquiera.

Registro de restaurante.

Tras la conexión a base de datos, el siguiente paso es el registro de restaurante. Contamos con que, para poder iniciar sesión, es necesario un registro previo. Para ello se crean tres ficheros en el

sistema, por un lado, la vista, por otro lado, la clase Restaurantes, y por otro el controlador que recibe la información de la vista y registra el restaurante en el sistema.

Recuperación de contraseña.

Se contempla la opción de solicitar un cambio de contraseña por parte del restaurante. El proceso a seguir para ello sería desde la página index.php hacer clic sobre el enlace “he olvidado mi contraseña”, este redirigirá al archivo recuperaciónPass.php, que solicita un email válido, y envía un email personalizado con el ID del restaurante correspondiente al email, y un enlace que redirige en función a ese ID a una página donde podremos indicar la nueva contraseña a cambiar.

Algo positivo sobre este apartado es que, hasta que no se indica la nueva contraseña, la anterior sigue vigente, por lo que si hemos indicado por error el cambio de contraseña no se hará efectivo hasta realizar este último paso ni se borrará la actual.

Una vez cambiada, se nos enviará a la página de inicio para poder iniciar sesión de manera normal.

Inicio de sesión.

Tras el registro (y la posible solicitud de cambio de contraseña), se debe seguir con el inicio de sesión. Para el inicio de sesión directamente desde la página index.php indicaremos el mail de registro y la contraseña. Al enviar el formulario este comunica con el controlador inicioSesion.php, que valida las credenciales indicadas y en caso de ser correctos y existir en la base de datos ambos dos, nos enviará a la página main.php, página principal de la aplicación y la interfaz principal que se verá al introducirse en la app.

Tanto el mail como la contraseña se envían por url, por lo que para darle un punto más de seguridad se envían codificadas en base64, y una vez en se decodifican, y se guarda en una variable de sesión toda la información relativa al restaurante para su uso posterior.

Un dato muy importante que es necesario almacenar es el número de confirmación que se solicita al registrar el restaurante. Este número se considera una clave de seguridad a la hora de eliminar elementos importantes de la aplicación, como son mesas, facturas, platos o reservas. De esta forma contamos con una protección extra ante posibles errores humanos al introducir datos en la aplicación.

Otro dato importante de esta página principal es que, en caso de cerrar la aplicación mal y quedar mesas sin finalizar o eliminar, al comienzo de la aplicación contamos con un método que recibe la fecha del día actual y en caso de existir mesas sin finalizar anteriores al día las eliminaría, ya que realmente no deberían de contar para el recuento de ningún día real.

Todas las páginas de ahora en adelante a excepción del cierre del día y la información del restaurante cuentan con dos datos en la parte superior. A la izquierda el título de la página donde se encuentra en cliente, y a la derecha un resumen general que muestra el día actual y el número de reservas con el total de clientes reservados (pax).

Mesas.

Este proceso contaría con la creación, eliminación, finalización y listado de mesas. Todos estos procesos se realizan directamente desde la página main.php, una vez se ha iniciado la aplicación.

El listado de mesas trabaja directamente con la base de datos, generando un listado de las mesas no finalizadas y almacenándolo en una variable de sesión, junto a las cuentas no finalizadas y las finalizadas. Al arrancar la aplicación, estas tres deberían estar vacías, debido a que no hay ninguna existente.

Para la creación de mesas disponemos de un formulario que recibe tanto el número de la mesa en la que se han sentado los clientes como el número de comensales, y al enviar el formulario se establecerá una comunicación entre el controlador addMesa.php y la clase mesas, que creará en base de datos una tupla en la tabla de

mesas y a su vez almacenará en la variable de sesión de mesas activas los datos de la mesa creada, y a su vez se creará una cuenta vacía asociada a esa mesa tanto en base de datos como en la variable de sesión de cuentas no finalizadas. Hasta no disponer de mesas, no se podrá disponer tampoco de la función de eliminarlas o finalizarlas, ya que estas dos opciones serán visibles pinchando sobre el icono de mesa que aparece en la vista.

Para eliminar o finalizar la mesa, haciendo clic sobre el icono de la mesa que queremos trabajar aparecerá un display inferior con tres iconos. El icono circular con la “x” eliminaría la mesa, previa contraseña o código de seguridad, de la base de datos y de las variables de sesión junto a la cuenta asociada a la mesa. Con el icono con un tic finalizaría la mesa, lo que no solo eliminaría la mesa de la variable de sesión de mesas, sino que movería la cuenta asociada de cuentas no finalizadas a cuentas finalizadas en las variables de sesión, y cambiaría el estado de la mesa en la base de datos de no finalizada a finalizada.

Cabe decir que, al finalizar la mesa, la variable de sesión habrá ido almacenando entre sus valores dentro de un array una serie de objetos de la clase Platos. Estos objetos se almacenarán en este momento de finalizar en base de datos, pero como no se pueden almacenar objetos el programa realiza dos pasos muy sencillos para poder guardarlos como un string. En primer lugar, los transforma a formato JSON, y luego los codifica en base64. De esta forma lo que se almacena es un string en base de datos que posteriormente será decodificado y decodificado de JSON a array de objetos.

Cuentas.

Este apartado cuenta con la visualización de la cuenta de la mesa deseada, y el listado de las cuentas finalizadas y no finalizadas.

En primer lugar, para la visualización de las cuentas se trabajará directamente en la página principal, dentro del display de cada mesa nombrado anteriormente. Pinchando sobre el icono del ojo, este nos redirigirá a la página visualizarCuenta.php, con un ID de mesa

concreto. Esto hará que se localice la cuenta exacta dentro de la variable de sesión que está almacenando los platos y se muestre un listado de los platos introducidos por el camarero en la aplicación, y realizando un sumatorio del total incluyendo un 10% de IVA estipulado por ley. Estos platos darán la oportunidad de ser borrados, eliminándolos del lugar donde se almacenan.

Saliendo de la página principal y entrando en historialCuentas.php, se dispondrá de partes bien diferenciadas, por un lado, cuentas no finalizadas, que lista todas las cuentas de mesas que no han sido finalizadas, y por otro las que si han sido finalizadas, estas últimas con opción de generar factura. Los dos tipos mostrarán el número de mesa que tienen, los comensales, la fecha y la hora de llegada de los clientes. De esta forma, será más fácil localizar una cuenta si se requiere de hacer factura.

Platos.

Este apartado cuenta con el listado de platos existentes en el menú, la creación de platos nuevos, su eliminación y la posibilidad de añadir un plato a una cuenta existente.

Desde el apartado cuentas en adelante, a excepción de la página main, para corroborar que se ha iniciado sesión y no se intenta acceder de manera ilícita, ejecuta un condicional que solicita la recuperación del restaurante con las variables de sesión. De haber iniciado sesión de manera normal, no habrá problema y el código continuará, pero si no se ha iniciado sesión, la variable de sesión de restaurante no existirá por lo que redirigirá a la página index para poder iniciar sesión.

Para el listado de platos, se instanciará un objeto de la clase DAO (con métodos de acceso y visualización de datos) para listar los platos del menú. Como es evidente en un primer momento no habrá ninguno creado por lo que es trabajo del restaurante crearlos y categorizarlos.

Arriba dispondremos de un menú selectivo trabajado con JQuery, el cual muestra solamente los platos cuya categoría corresponda con el nombre del enlace.

Para la creación de platos, hay un formulario de creación de estos en la parte superior junto al menú de categorías. Este formulario recoge tres datos, el nombre, la categoría a la que pertenece y el precio. Los tres campos están validados con JavaScript y cliente, para evitar la inserción de código malicioso, y con PHP para la validación de los datos y el filtrado de las variables recibidas para aportar seguridad. Una vez recibidos los datos por el controlador, este crea un objeto de la clase plato que ejecuta su propio método para la creación de plato, devolviendo en caso de éxito el flujo hacia la página de listado de platos.

En el caso de la eliminación el proceso es similar, a diferencia de que, como es un proceso importante que requiere confirmación, solicitará antes el código de confirmación necesario para realizar este tipo de operaciones.

Por otro lado, el proceso de añadir plato a cuenta es más complejo. Este solicita tres parámetros, el número de mesa, que deberá estar activa, la cantidad, y el ID del plato añadido. Con estos parámetros el controlador origina en primer lugar un objeto de la clase plato, le establece una cantidad, serializa el objeto para almacenarlo en la variable de sesión de cuenta no finalizada correspondiente a la mesa seleccionada y acto seguido localiza la cuenta en cuestión en base de datos y le suma el precio de lo pedido por la cantidad seleccionada al precio total de la cuenta. En este momento, la cuenta tiene un precio en su columna precio, pero tiene la de platos a cero. Esto se debe a que, hasta que no se finaliza la mesa, no se almacena el string final de los platos consumidos por el cliente a la base de datos, lo cual es de gran utilidad, ya que solo accederíamos a la base de datos una sola vez para actualizar esa columna, y no cada vez que añadiéramos platos, trabajando directamente sobre la cuenta en concreto de la variable de sesión. En resumen, el precio si cambia según se añaden o eliminan platos, pero los platos solo se actualizan al finalizar la mesa.

Reservas.

Este apartado lista las reservas del día actual, genera nuevas reservas para el día seleccionado o permite cambiar el día para comprobar las reservas o realizar nuevas reservas. Al comenzar en este punto se valida, al igual que antes, si se ha iniciado sesión o no. Una vez comprobado cargará las reservas del día actual. Esto se hace teniendo en cuenta la visión de trabajo de los empleados de un restaurante, ya que lo importante es visualizar en primer momento en número total de reservas de ese día.

Para crear una reserva simplemente se rellenará un formulario al que redirige el botón "+" de la página, generarReserva.php, que incluye la fecha del día, y que requiere de rellenar cada uno de los apartados del formulario, validados tanto en cliente como en servidor. Este formulario no impide que dos mesas reserven a la misma hora, esto es debido a que, en un mismo restaurante, puede haber muchas mesas que llegan a la vez, por lo que dependerá de número total de mesas que disponga el restaurante.

Si se desea generar una reserva para un día que no es el actual, se dispondrá de un selector de fecha en la parte derecha, que cambiará la fecha de visualización de este apartado a la requerida, pudiendo hacer reservas para ese día y visualizando las existentes.

Facturas.

Este apartado muestra las últimas diez facturas generadas por la aplicación en orden inverso, es decir, de más actual a más antigua. Este número de facturas se puede aumentar de diez en diez con el botón "Mostrar 10 más" de la parte superior, que aumenta en 10 la variable que sirve de bandera para la función que crea la vista de las facturas. Estas facturas mostrarán en primer lugar el ID que tienen dentro de la aplicación, la fecha de creación, la empresa que ha solicitado la factura y el contacto que la solicitó. Si se necesitaran más datos del cliente se podrá visualizar la factura haciendo clic sobre el icono del lateral derecho, que redirige a la página visualizarFactura.php con el ID

correspondiente, y muestra todos los datos de la factura, permitiendo su descarga.

Cada factura dispone de sus propios botones de visualización, descarga, envío y eliminación. Evidentemente con la eliminación sucede como con todos los datos importantes en la app, requieren de una contraseña o número secreto que valide esta eliminación. En cuanto a la descarga, nos evita tener que abrir y visualizar la factura si tenemos claro que es la correcta, al igual que enviarla por correo, función que requiere de un correo válido que no será almacenado en la base de datos para enviar dicha factura.

Información del restaurante.

La información del restaurante se muestra en este apartado. Este solamente mostrará los datos existentes, por lo que, en caso de no haber rellenado el apartado dirección, o la persona de contacto, este dato no se mostrará. Sin embargo, ofrece la posibilidad de cambiar los datos del restaurante haciendo clic sobre el enlace inferior, lo cual nos redirige a un formulario que se autocompleta con los datos actuales del restaurante. De todos los datos, el único que no se puede cambiar es el NIF, debido a que por ley una vez autorizado el NIF a un local de hostelería este es un identificador único e inmutable.

Este formulario, al igual que el resto en toda la aplicación, se valida en cliente y servidor, tanto al indicar los valores de los campos, como al enviar el formulario.

Una vez enviado y validado el cambio, el flujo redirige a la visualización de los datos del restaurante para corroborar que todos los datos han sido cambiados con éxito.

Cierre del día.

Para finalizar, se ha realizado este apartado como finalización del flujo de trabajo de la aplicación. Este paso sería el último del día dentro de esta y nos muestra un cómputo de resultados obtenidos de las bases de datos y sus sumatorios. De esta forma se obtiene, en un primer lugar, el número de mesas atendidas durante el

día, ejecutando una consulta de suma de mesas finalizadas para el día actual. En segundo lugar, los comensales atendidos, que de nuevo será la suma de los comensales totales por mesa atendida y finalizada para ese día. A continuación, se muestra un resumen de todos los platos vendidos durante el día, el precio unitario la cantidad de platos de ese tipo vendidos y el total obtenido monetariamente.

Este último dato deriva en otro dato, un sumatorio del total obtenido por el restaurante a lo largo del día en productos.

Como extra, el restaurante podrá exportar esta tabla en formato Excel. Si bien no es necesario, ya que no todos los restaurantes lo realizan de esta forma, este documento para el cierre del día puede ser de utilidad si el hostelero llevara la contabilidad de esta manera, aportándole a la aplicación un valor añadido.

4.5 PRUEBAS

4.5.1. Plan de pruebas

Para la realización de pruebas se ha elaborado un plan de pruebas consistente en la realización de las siguientes pruebas:

-Pruebas unitarias, en las que se probarán uno a uno todos los métodos existentes en la aplicación para corroborar errores por medio de pruebas de caja blanca. Una vez finalizadas, en caso de haber algún error se realizarán pruebas de regresión, realizando las pruebas para comprobar si los cambios realizados han modificado el correcto funcionamiento de los métodos ya probados con anterioridad.

-Pruebas de integración, en las que se comprobará que en las pruebas anteriores funcionan con el resto del código en conjunto en sus respectivos apartados.

-Pruebas de validación, mediante las cuales se efectúa la comprobación de si lo que realiza el programa cumple con lo establecido en los objetivos de la aplicación.

-Pruebas de sistema, las pruebas más completas y

específicas de la aplicación, que se realizarán mediante casos prácticos reales y mediante las cuales se validarán los objetivos de la aplicación y el cumplimiento de todas las funciones previstas de manera correcta. Si bien este tipo de pruebas debe realizarlas el cliente final para tener pruebas reales, al ser una app en desarrollo se someterá a test tipo.

4.5.2 Pruebas unitarias

Dentro de cada método que se ha programado en la aplicación, se han realizado dos tipos de pruebas unitarias, por un lado, las pruebas de caja blanca, *white box* o estructurales, que han corroborado que el diseño de la aplicación a nivel de estructura de código ha sido el correcto, y, por otro lado, pruebas de caja negra, *black box* o funcionales, que se realizan en la parte interactiva de la aplicación y validan que el resultado que devuelve sea el correcto a nivel funcional. En lo referente a las primeras, se han realizado principalmente sobre todas las iteraciones, bucles y condicionales de la aplicación, que son habituales de encontrar en los ficheros de la carpeta utils (o lo que entendemos por controladores), y en los métodos de los modelos o clases existentes, como, por ejemplo, las siguientes porciones de código:

```php
<!-- PRUEBA CAJA BLANCA 1: CONDICIONAL DE CREACION DE MESA -->
<?php

    $mesa = 1;
    $comensales = 2;

    if($mesa!=""&&$comensales!=""){
        $registro = new Mesa(null, $mesa, $comensales, $fecha, $hora, 0);
        $respuesta = $registro->crearMesa($mesa, $comensales, $fecha, $hora, $_SESSION['restaurante']['id']);
        if($respuesta){
            $id = $dao::localizarMesa($mesa, $comensales, $fecha, $hora, $_SESSION['restaurante']['id']);
            $cuenta = new Cuenta(null, $comensales);
            $respuestaCuentas = $cuenta->crearCuenta($comensales, $id, $idRest);
            if($respuestaCuentas){
                array_push($_SESSION['cuentasNF'], $respuestaCuentas);
                echo "Mesa creada con éxito, cuenta creada con éxito";
            }else{
                echo "Mesa creada con éxito, pero no se ha creado la cuenta.";
            }
        }else{
            echo "No se ha podido crear la mesa.";
        }
    }else{
        echo "No existen número de mesa o comensales.";
    }
?>
```

Imagen 13. Prueba de caja blanca n.º1, creación de una mesa

Como se observa en el código (Imagen 13), se inicializan dos variables necesarias para entrar en la condición, de no existir entraríamos en el else del primer nivel, una vez dentro se crea una mesa, se registra en base de datos, y si la respuesta es positiva, se localiza la mesa, se crea una cuenta y se añade a la base de datos, de igual manera si este proceso ha funcionado se añade a la variable de sesión correspondiente y finalizaría la prueba. Como es evidente, este código es de prueba, por lo que el código real no lanza mensajes de prueba si no redirecciones a las páginas que corresponden con mensajes de éxito o error en url.

```php
<!-- PRUEBA CAJA BLANCA 2: CAMBIO DE CONTRASEÑA EN APLICACIÓN -->
<?php

    $id=1;
    $_REQUEST['pass']=1234;
    $_REQUEST['pass2']=4321;

    if(isset($_REQUEST['pass'])&&$_REQUEST['pass']==$_REQUEST['pass2']){
            $dao = new Metodos();
            $dao::nuevaPass($id, $_REQUEST['pass']);
    }else{
        $id=base64_encode($id);
        header("Location:../nuevaPassword.php?id=$id&error_log=1");
    }

?>
```

Imagen 14. Prueba de caja blanca n.º2, cambio de contraseña

En este código (Imagen 14) observamos el código de cambio de contraseña que se realiza al solicitar una nueva contraseña. Se generan variables de ID, y contraseña 1 y 2, y se puede ver como en caso de existir la variable de contraseña, y ser iguales, entra en el condicional, solicitando el cambio de contraseña, y en caso de fallar alguna de las condiciones, codifica el ID y reenvía al login de la aplicación con un ID codificado en base64 y un mensaje de error.

Estas pruebas han sido realizadas en cada parte clave del código susceptible de cometer errores, de esta forma se ha corroborado que, efectivamente, a nivel estructural no se ha cometido error en las

sentencias y el código ha quedado pulido de fallos. No obstante, gracias a la extensión SonarLint, que se ha empleado para esta parte del trabajo entre otras, todos los posibles errores en variables o estructurales han sido solventados.

Tras estas pruebas, en caso de error en cualquiera de ellas, se realizarían las pruebas de regresión, que son todas aquellas pruebas que se efectúan sobre el código probado con anterioridad a la salida del error para comprobar que el cambio realizado no haya afectado al código validado. Debido a que las pruebas de caja blanca salieron positivas en su totalidad, no ha sido necesario la realización de pruebas de regresión.

En cuanto a las pruebas de caja negra, que son aquellas que se realizan para comprobar que el resultado es el esperado sin tener en cuenta la estructura de código, las cuales al final han demostrado que, al indicar unos valores correctos el resultado que da la aplicación es el esperado, y en caso de introducir valores erróneos o inválidos, la aplicación sigue devolviendo el mensaje esperado, pudendo utilizar como ejemplo las líneas de código anteriormente vistas. Siguiendo con la línea anterior, observamos que:

-Como se observa (imagen 13), se introduce un número de mesa y un número de comensales factible, y al ser dos variables posibles, se espera que se cree la mesa mostrando un mensaje de éxito.

-Por el contrario, si se introduce un número de mesa ya ocupada, la aplicación debe mostrar un mensaje de error diciendo que la mesa está ocupada y acto seguido no realizar nada más, si el número de mesa no ha sido declarado o el número de comensales tampoco, la aplicación deberá mostrar un mensaje diciendo que no se han indicado las variables designadas, y si por motivos de acceso a base de datos no se hubiera podido crear o la mesa o la cuenta asignada, la aplicación debería mostrar un mensaje indicándolo.

-Como resultado obtenemos que, en ambas situaciones, el resultado es el esperado para ambas situaciones, por lo que esta prueba quedaría realizada con éxito.

-Siguiendo con el ejemplo, vemos que si se cumplen las condiciones exigidas en el cambio de contraseña (imagen 14), se realizará el cambio de contraseña, y si no deberá devolver a la página origen con mensaje de error.

-Como resultado obtenemos que sucede lo esperado, por lo que esta segunda prueba queda resuelta con éxito.

Todos los apartados de la aplicación susceptibles de fallo en la aplicación han sido testeados con este tipo de pruebas igual que con las anteriores, desde el inicio de sesión, el registro de restaurante y el cambio de contraseña hasta la creación de mesas, creación de platos, creación de facturas, generación de cierres del día, etc.

4.5.3. Pruebas de integración

Las pruebas de integración que se han realizado han servido para comprobar que el código realizado y testeado en las pruebas unitarias se integran como su propio nombre indica con el resto de código.

Entre las pruebas realizadas, se han comprobado que entre cada módulo de código testeado en las pruebas unitarias existe una relación y el funcionamiento es correcto y conforme a lo deseado. No obstante, se muestran a continuación varios ejemplos.

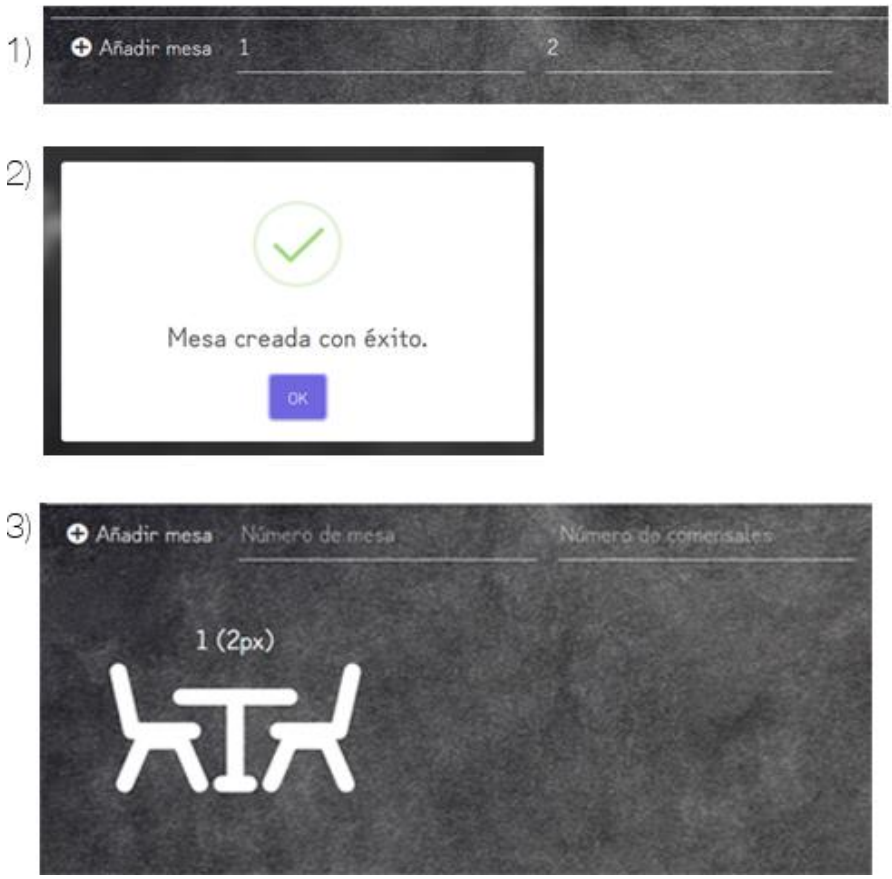


Imagen 15. Prueba de integración n.º1, creación de mesa correcta

En esta muestra (imagen 15) vemos la integración del código que crea las mesas con su apartado visual, funcionando correctamente y generando una mesa nueva, por lo que la prueba de integración se consideraría válida y realizada con éxito.

De igual forma, se comprueba que funcionan todas las posibilidades del código introduciendo un número de mesa ocupada por lo que la app devuelve una negativa ante el intento de creación de mesa (imagen 16).

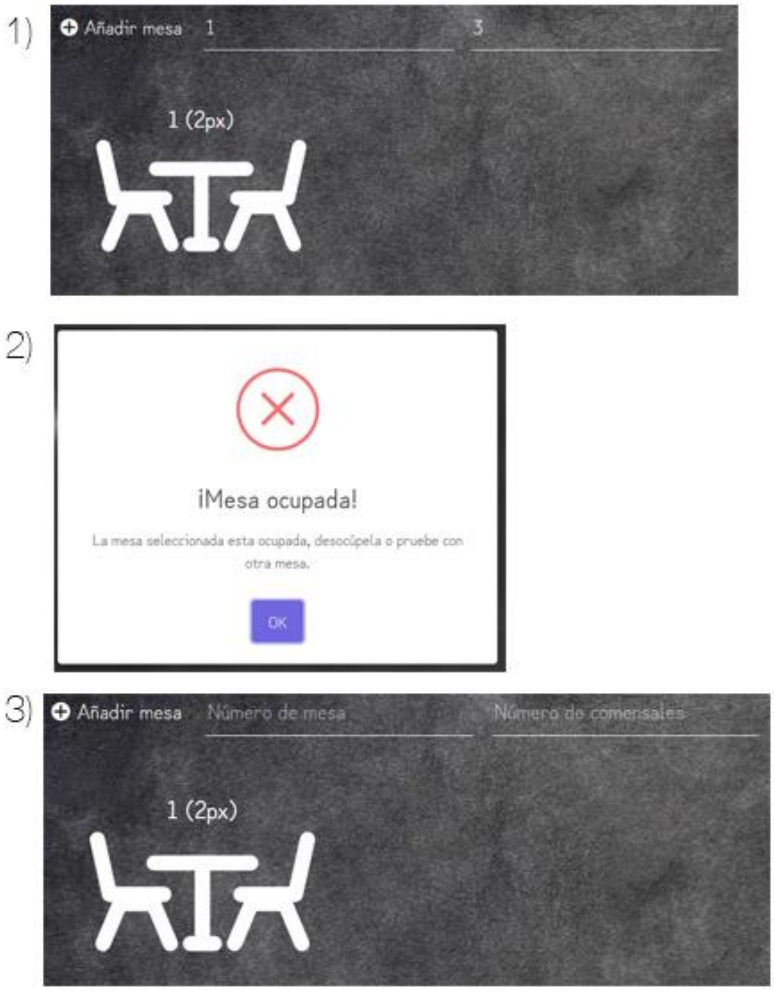


Imagen 16. Prueba de integración n.º2, creación de mesa incorrecta.

En la eliminación de platos (imagen 17), vemos como se coordina funcionando sin error el módulo de eliminación de platos de la carta. Así se observa como si se introduce el número de confirmación o código seguro secreto erróneo en el apartado de confirmación salta un

mensaje de error (escenario 1) y en caso contrario, se procede a la eliminación del plato (escenario 2).

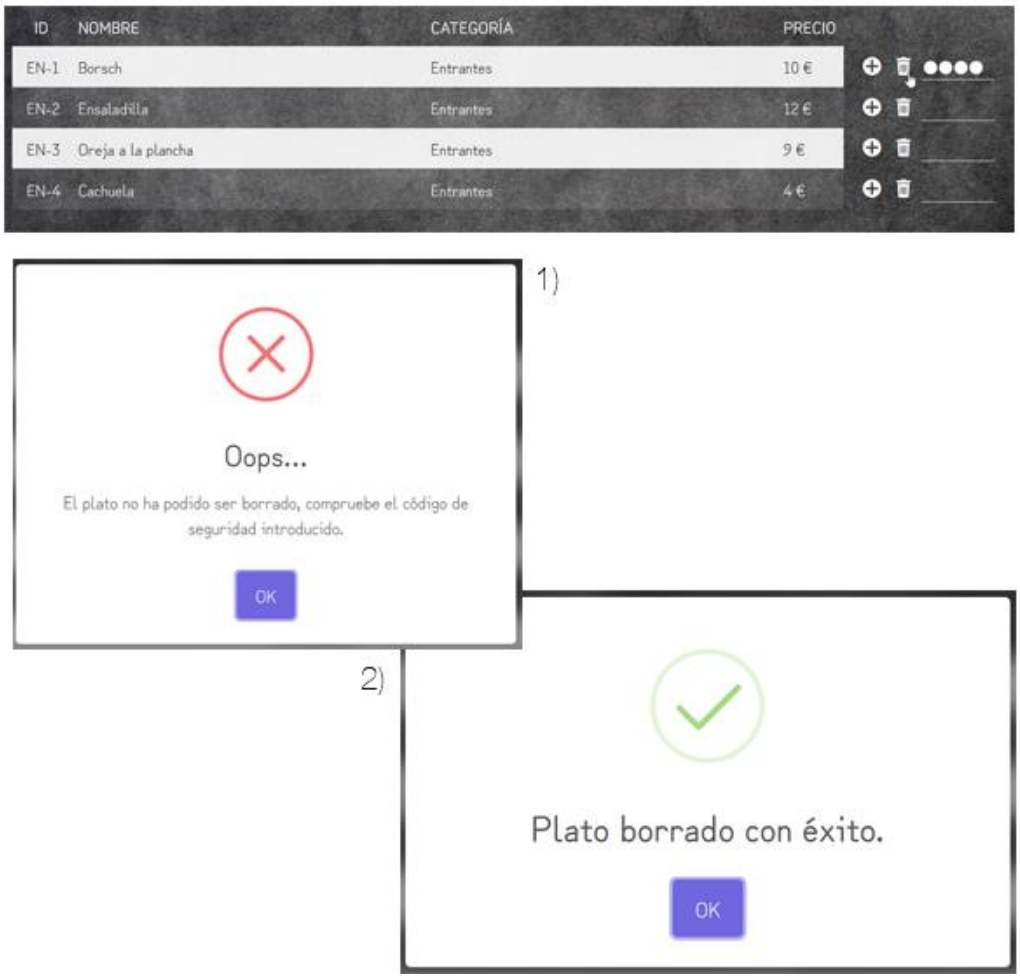


Imagen 17. Prueba de integración n.º3, eliminación de platos

Por último, se muestra a continuación como no solo se han realizado pruebas en el lenguaje principal de la aplicación (PHP), sino que se han realizado pruebas mediante las cuales se ha comprobado el buen funcionamiento conjunto de las validaciones JavaScript con PHP en el apartado de creación de reservas (imagen 18).

Según el escenario uno, siempre que se introduzca una letra en un campo numérico como es comensales o teléfono, dará un mensaje de error, o incluso si, evadiendo el código JavaScript por medio de navegador, intentamos mandar el formulario, este mostrara el mensaje de error dos. Mientras, en el escenario dos vemos que el formulario se

ha rellenado correctamente, la validación ha salido positiva tanto en JavaScript como PHP y se ha podido generar la reserva, lo que demuestra la correcta integración de ambos paquetes de código.

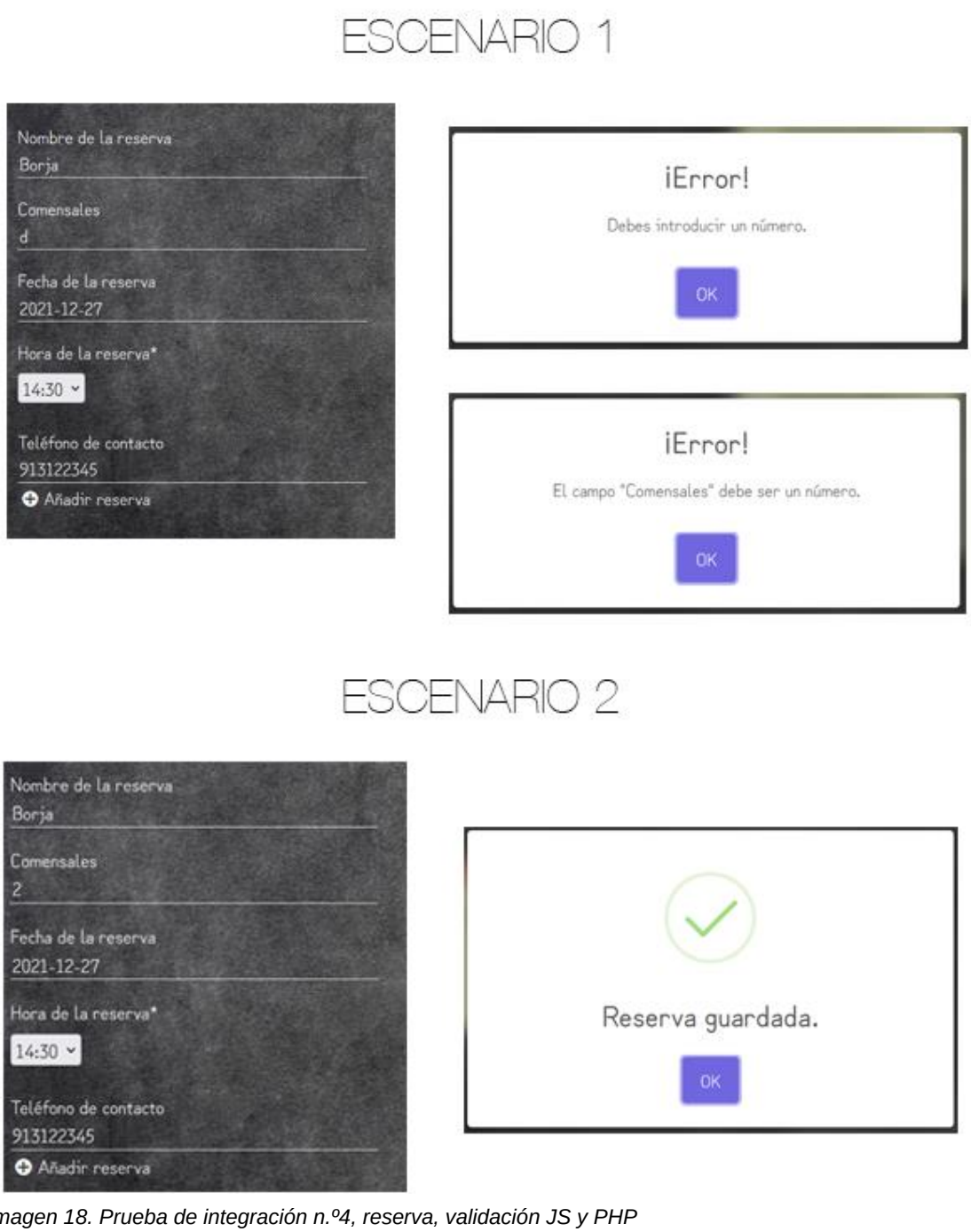


Imagen 18. Prueba de integración n.º4, reserva, validación JS y PHP

4.5.4. Pruebas de validación y resultados

Las pruebas de validación se rigen por el cumplimiento de los requisitos establecidos en el siguiente listado, que son las funciones principales que debe cumplir la aplicación y sin las cuales no sería la aplicación final al completo:

- Listado de mesas. La prueba que se ha realizado es la visualización en el entorno gráfico del listado de las mesas ocupadas en el restaurante a tiempo real. Tras entrar en la pantalla de inicio se puede ver como la lista de mesas está vacía y según se van añadiendo aumenta el número total de mesas. Además, en caso de cerrar mal la aplicación o de terminar el día sin realizar el cierre de día, al abrir la aplicación se mostrará vacía la “sala”, ya que no existen esas mesas en el día, y al no haberse finalizado no serán contabilizadas para el día anterior.

- Creación, eliminación y finalización de estado de mesas. La prueba realizada consta de la creación de dos mesas diferentes, la finalización de una y la eliminación de otra. El resultado esperado es el mismo que el obtenido, por lo que cumple con el requisito.

- Listado de cuentas no finalizadas y finalizadas. La prueba se realiza posterior a la anterior, de tal forma que, una vez finalizada la mesa, comprobaremos que se listan las mesas existentes no finalizadas, y en la lista de finalizadas aparecerá la anterior.

- Creación de facturas según número de cuenta. La prueba realizada será la creación de una factura ficticia a raíz de una cuenta finalizada. El resultado esperado es la correcta creación de la factura, devolviéndonos a la página de listado de facturas.

- Listado de facturas, visualización, descarga, envío y eliminación de facturas. Desde la página de facturas, visualizaremos el listado de facturas creadas, y realizar una descarga, envío vía mail, visualizarla correctamente y eliminarla. El resultado esperado es la correcta visualización de la factura, la descarga de la misma en el

sistema que nos encontremos, el envío de la misma por mail y que se elimine del sistema correctamente.

- Listado de platos del menú, y visualización discriminativa por categoría. La prueba resultante será la visualización del listado completo de platos, diferenciados con un identificador único compuesto por un número y las dos primeras letras de la categoría, el nombre, la categoría propia y el precio. Al hacer click sobre la categoría del menú superior nos hará una criba discriminativa mostrándonos solamente los platos correspondientes a esa categoría.

- Creación y eliminación de platos. Tras la anterior prueba, se probará la creación de un nuevo plato.

- Adición de platos a cuentas según número de mesa. Desde el apartado de menú, se prueba a añadir un plato a una mesa existente indicando cantidad, el mismo plato, pero si indicar cantidad, y la adición del plato a una mesa no existente. El resultado esperado es en primer lugar la suma de los platos a la cuenta de la mesa, en segundo lugar, un mensaje de error indicando que debe especificar cantidad y en tercer lugar un mensaje que indica que la mesa seleccionada no existe.

- Listado de reservas según fecha. Visualización de las reservas del día actual en el panel de reservas.

- Creación y eliminación de reservas según elección de fecha. Se procederá a la creación de una reserva en el día actual, y tras su creación, se probará a eliminarla.

- Búsqueda de reservas según fecha seleccionada. En el buscador de fecha de la parte derecha del panel de reservas se probará a localizar una fecha diferente a la actual, para comprobar que muestra las reservas de esa fecha. En una primera búsqueda no encontrará reservas, por lo que procederemos a crear la reserva en ese día para la visualización de la misma.

- Visualización del cierre del día con el resumen de mesas, comensales, y listado de platos vendidos y ganancias obtenidas.

- Descarga del cierre del día en formato Excel. Se probará la opción de descarga del cierre del día en Excel por medio de un botón.

4.5.5. Pruebas de sistema

Para las pruebas de sistema lo que se requiere es que, aparte de funcionar cada módulo individualmente, funcionen en conjunto con casos prácticos reales, o, en su defecto, lo más cercano a la realidad posible. Para ello se han creado 4 escenarios diferentes:

-Escenario 1: Se inicia la aplicación, se introducen en la aplicación las reservas del día (2 reservas, una de 2 personas, otra de 4 personas), llega una mesa sin reserva de 3 personas, más las otras dos mesas reservadas. La mesa que viene sin reserva decide no comer, e irse, por lo que debe eliminarse la mesa. Las otras dos piden diferentes platos que serán añadidos a la cuenta. Ambas mesas pagan y se marchan (imagen 19).

-Escenario 2: Partiendo del escenario uno, la mesa de dos personas viene más tarde a solicitar una factura. Solicita a versión impresa y que se le envíe a su mail personal (imagen 20).

-Escenario 3: Después de todo el día, el jefe de cocina ha añadido un plato a la carta, un plato principal por 15€ la ración, por lo que se añade el plato y después se realiza el cierre de día para finalizar. Este cierre será descargado y almacenado para futura contabilidad (imagen 21).

-Escenario 4: Tras un tiempo en marcha, el restaurante cambia de nombre a “Florituras S.L.”, y de teléfono al 913452345. El resto de datos permanecen como hasta entonces (imagen 22).

Tras la realización de estas pruebas se concluye que, a niveles realistas, la aplicación cumple con todas las funciones designadas desde el comienzo del proyecto, y las pruebas han resultado positivas. Si bien es necesario que la aplicación se prueba a nivel real, estos cuatro escenarios constituyen el grueso de las necesidades que pueden cubrirse con la aplicación.

Imagen 19. Prueba de sistema. Escenario 1, reservas, mesas y cuentas

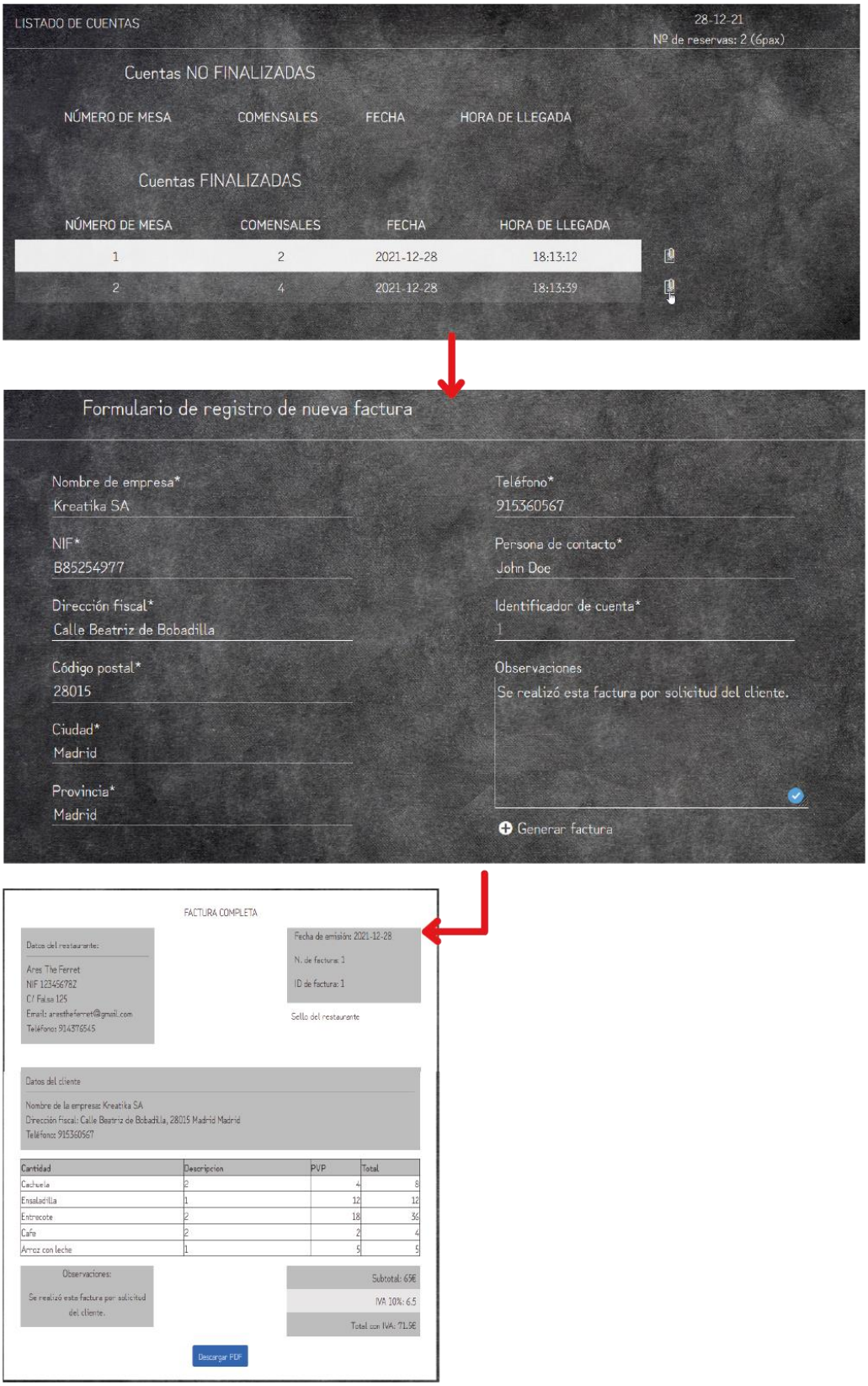


Imagen 20. Prueba de sistema. Escenario 2, generación de factura.

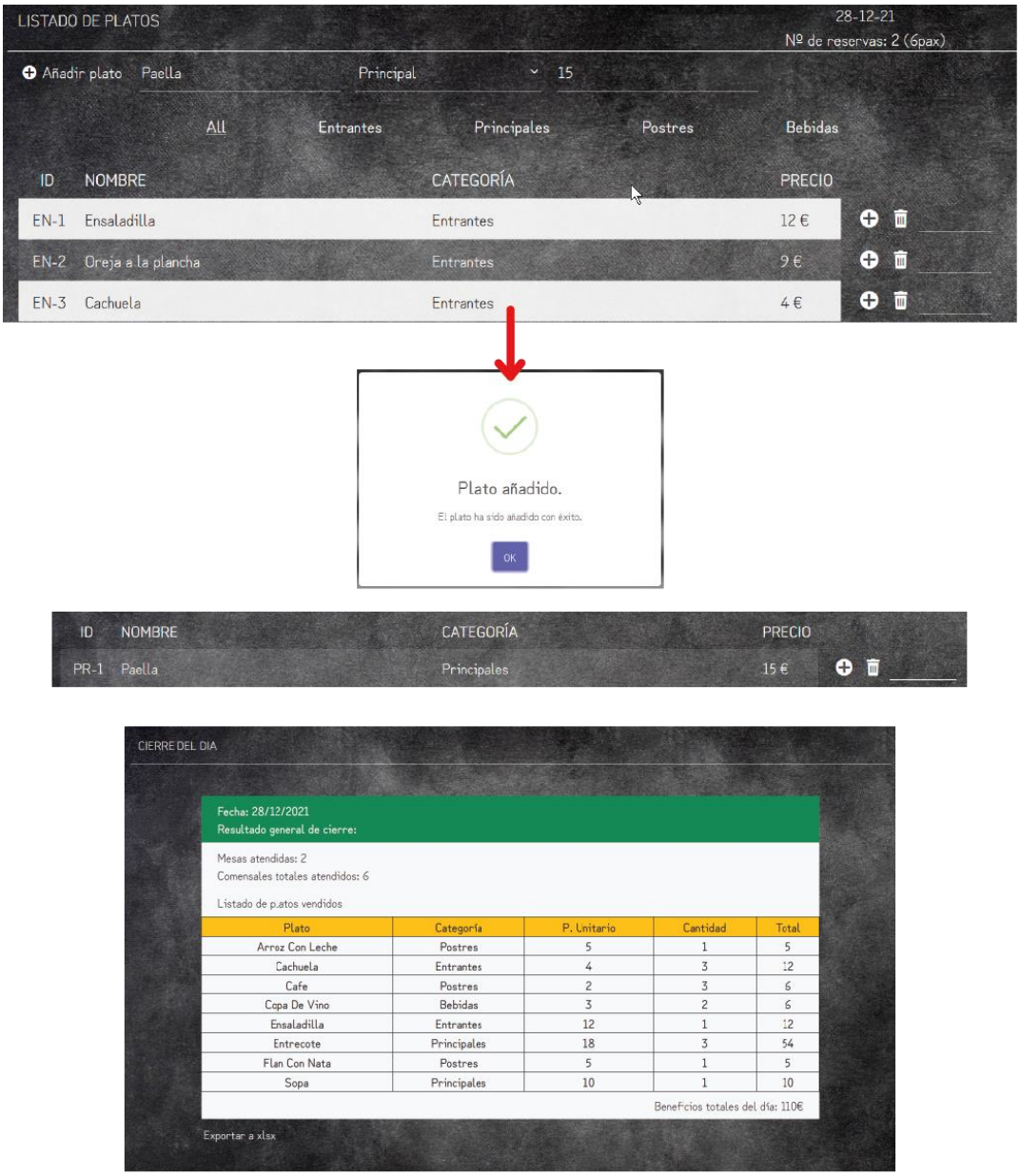


Imagen 21. Prueba de sistema. Escenario 3, generación de platos y cierre

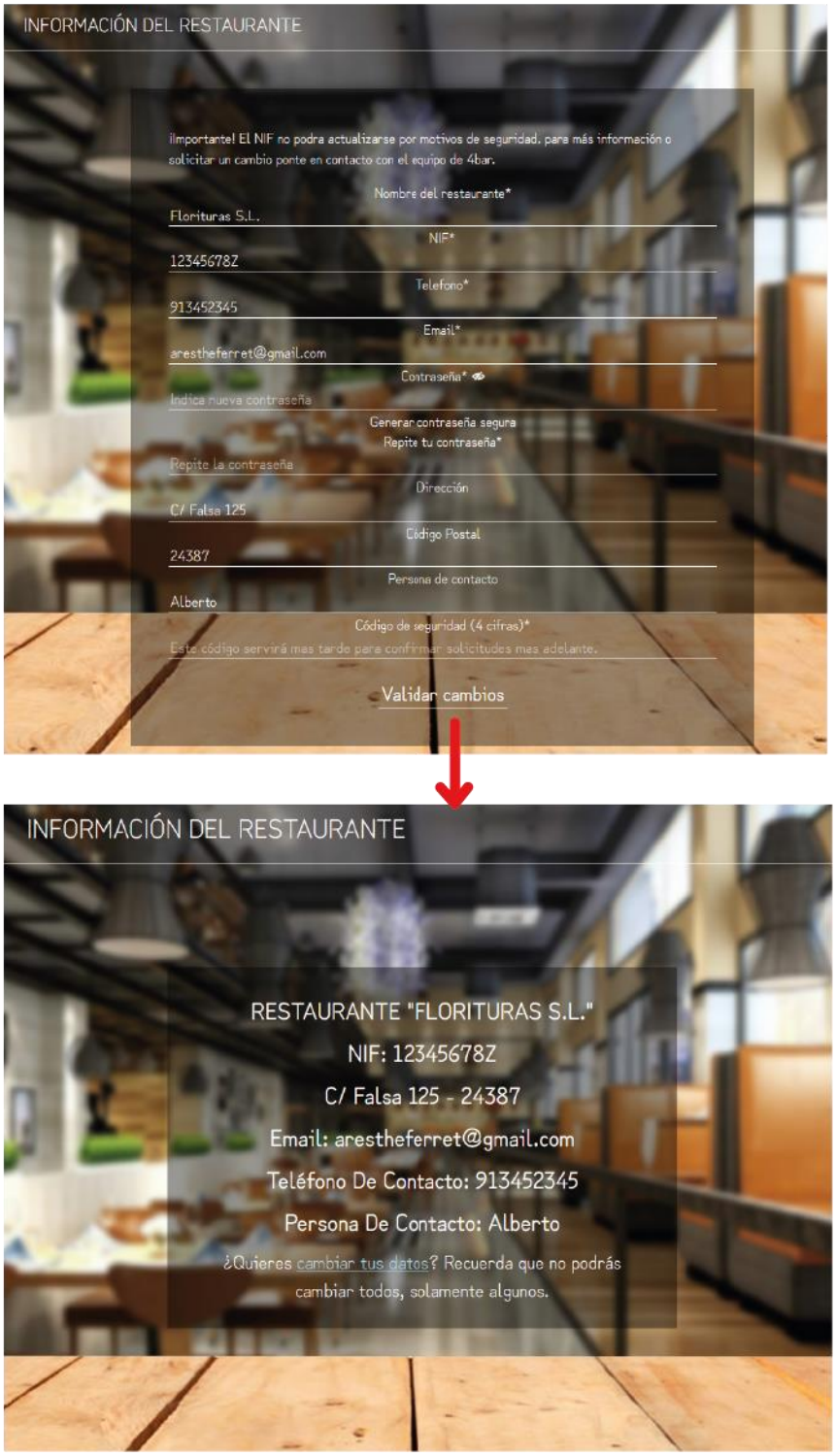


Imagen 22. Prueba de sistema. Escenario 4, cambio de datos del restaurante

5. ANÁLISIS DE RESULTADOS

Teniendo en cuenta tanto la consecución de objetivos de la aplicación, como lo obtenido de las pruebas, se pueden analizar los siguientes resultados tanto positivos como negativos.

De manera positiva, se ha logrado cumplir con todos los requisitos que necesita la aplicación para ser funcional, lo que significa que todas las funciones que realiza la aplicación no solo son ejecutables si no que, en mayor o menor medida, satisfacen las necesidades básicas que requiere un negocio de hostelería para la gestión del restaurante, ya que, como se ha comentado con anterioridad, la idea principal de esta aplicación es la de aunar en una sola todas las funciones necesarias.

Si bien es verdad que, en esta primera fase de la aplicación, esta tiene un gran número de funciones, es evidente que con la evolución natural de esta se irán implementando muchas más funciones y por supuesto mejora de las existentes, lo que equilibra bastante la balanza de cosas positivas y negativas.

Por otro lado, teniendo en cuenta los test realizados a sujetos ajenos a producción de la aplicación, se ha llegado a la conclusión de que, si bien la aplicación es muy completa y simple, requiere, por un lado, de un nivel de captación de conceptos elevado, ya que está enfocada a la jornada laboral de un empleado de hostelería el cual tiene una idea formada de como trabajar con estas aplicaciones, y por otro, la necesidad de crear un documento de manual de usuario. Este documento podría ser sustituido por una visita guiada por la aplicación al comienzo de su uso, siempre y cuando sea algo visual que el usuario pueda captar con facilidad.

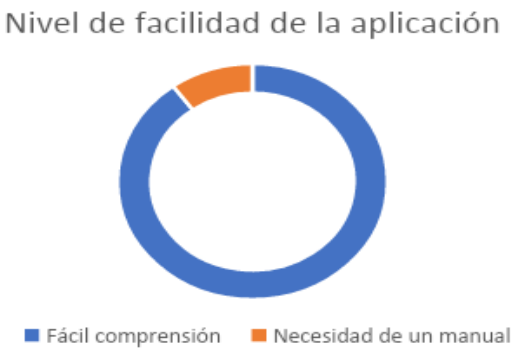


Gráfico 1. Gráfico circular sobre la facilidad de uso de la aplicación.

De todos los test realizados, un noventa por ciento de ellos considera que la aplicación se entiende correctamente, mientras que el diez por ciento restantes considera que sería de ayuda dicho manual o un video explicativo (grafico 1).

Además de todo esto, teniendo en cuenta el mock up realizado al comienzo del proyecto sobre la aplicación (anexo 2), se puede ver que de la idea inicial a la llevada a cabo ha habido grandes cambios en el planteamiento del desarrollo.

Entre estos cambios dividiremos en dos, funcionales y de diseño. Por un lado, el cambio más significativo a nivel funcional es la visualización de las reservas, ya que en un principio se planteó desde el punto de vista de un calendario, con posibilidad de ver los días reservados, pero de nuevo, desde el punto de vista de la persona que usará la aplicación, no es útil ver los días que tengo reservas, sino si efectivamente, tengo reservas ese mismo día, por ello se realizó este cambio.

Los cambios que podemos observar con respecto al diseño son la visualización de las mesas existentes, del display que muestran al hacer clic sobre ellas, o por ejemplo la visualización del cierre del día. Estos cambios son meramente visuales, ya que las funciones que realizan son exactas mostrando lo solicitado.

6. CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

Tras el trabajo realizado y los objetivos conseguidos, se ha llegado a la conclusión final de que el proyecto cumple con las expectativas iniciales formuladas, y por ello se considera un trabajo finalizado con éxito.

Como se ha comentado anteriormente, es evidente que este proyecto se encuentra en una primera fase de un proyecto mayor, con más funcionalidades y muchas mejoras posibles que se verán implementadas en las líneas de trabajo futuras. De entre todos los cambios posteriores que tendrá la aplicación, se destacan las funciones nuevas y las mejoras de las existentes.

FUNCIONES NUEVAS

Entre las funciones nuevas que se le añadirán se encuentran, para empezar, un recorrido interactivo para mostrar el funcionamiento de la aplicación y una sección con tutoriales, disponible siempre que sea preciso. Será añadida también la función de cobro, en la cual se podrá obtener un ticket de la mesa para la realización del pago y se podrá incluir la cantidad de dinero que da el cliente, de esta forma se podrá llevar una contabilidad mucho más exhaustiva incluso con el dinero en efectivo que hay en caja y contando los pagos con tarjeta efectuados por los clientes.

También se le añadirá al comienzo de su uso una pequeña encuesta para conocer más datos del restaurante. Hasta ahora se dispone de datos como nombre, NIF o similares, pero con esta encuesta podremos obtener datos como los turnos que tienen, que horario maneja el restaurante, de que número de mesas dispone, disposición de las mesas, cuantas categorías dispone su carta, o incluso una imagen del sello del local para poder incluirlo en las facturas, y hacer el uso de la aplicación mucho más amigable, cercana y personalizada.

Esta aplicación es principalmente para trabajo en sala, por lo que la parte de cocina se deja un poco más abandonada. Con las siguientes mejoras se creará un apartado de cocina para poder imprimir las comandas directamente en impresoras térmicas en cocina, fomentando la economía del tiempo y evitando fallos en el proceso de servicio por mala comunicación.

MEJORAS

En cuanto a las mejoras de funciones existentes, se pretende que el restaurante pueda disponer las mesas creadas a su antojo, mediante un sistema de Drag & drop clásico para poder moverlas. De igual forma se pretende corregir y mejorar el código para eliminar sentencias innecesarias consideradas como código basura y aumentar la velocidad de ejecución de la aplicación.

Como complemento veremos un pequeño calendario al lateral del apartado reservas, en el cual se visualizará los días que tienen reserva realizada resaltándolos de un color específico, en contraposición de los días sin reservas que no tendrán fondo. Se contempla que se pueda cambiar de día mediante este calendario y eliminar la opción de cambiar de día con el selector actual.

Para el apartado de cierre del día, se ofrecerá la posibilidad de almacenar estos cierres como información en la base de datos. Actualmente, como se ha dicho antes, no se suele hacer esto debido a que no todos los restaurantes llevan la contabilidad del negocio de esta manera, pero puede ser una mejora para la contabilidad muy sustancial.

Por último y más importante, la aplicación actual no dispone de un sistema responsive, lo que implica que de momento solo será visible en dispositivos laptop u ordenadores personales y tabletas en posición horizontal. Se pretende mejorar esto para poder, mediante dispositivos móviles, trabajar en las mesas cogiendo las comandas de los clientes. De esta forma se digitalizaría todo el proceso y el trabajo sería más eficaz.

7. REFERENCIAS BIBLIOGRÁFICAS

Cabezas Granado L.M. (2010) PHP 6. Anaya.

Bases de datos. (2019). En E. Sarabia (Comp.) Desarrollo de aplicaciones con tecnologías web. IES Virgen de la Paz.

Bootstrap. (s.f.) Bootstrap team. Disponible en: https://getbootstrap.com. Fecha de último acceso: diciembre 2021

PHP.net. (s.f.) The PHP Group. Disponible en: https://www.php.net. Fecha de último acceso: diciembre 2021.

SweetAlert2. (s.f.) Disponible en: https://sweetalert2.github.io. Fecha de último acceso: diciembre 2021.

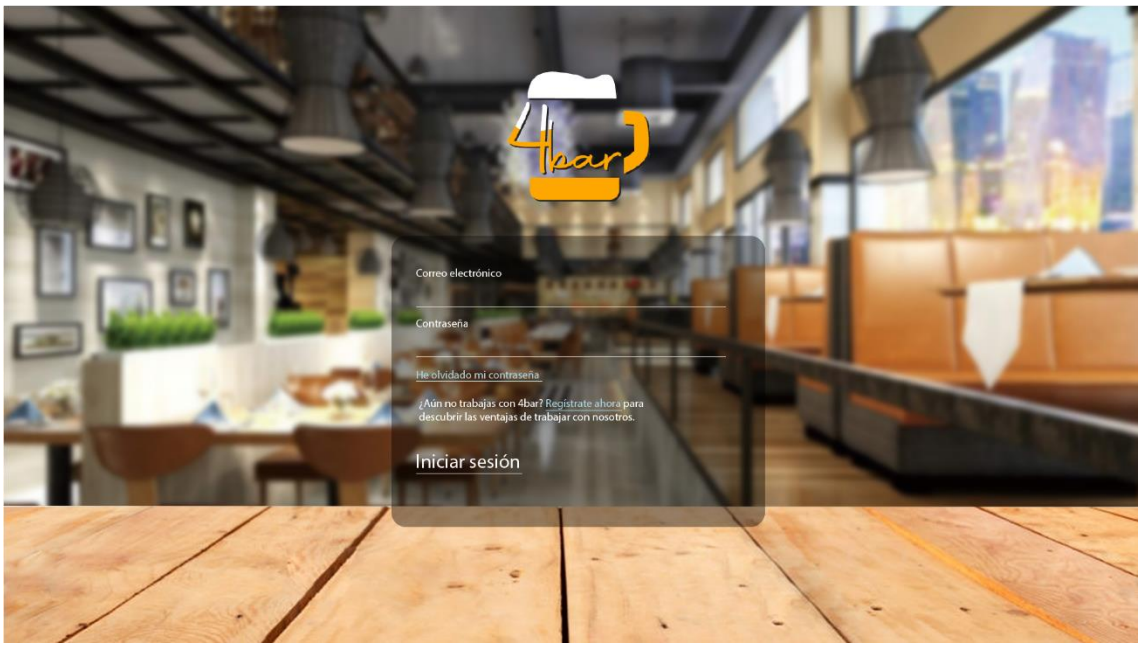
Gomez J., WEBPerfil (2018). Como recuperar e insertar arrays en base de datos de forma efectiva. Lugar de publicación: www.srcodigofuente.es. Recuperado de:https://www.srcodigofuente.es/tutoriales/ver-tutorial/como-almacenar-recuperar-array-base-de-datos

Syloper. Exportar tabla html a Excel (xsl) en PHP. Lugar de publicación: www.syloper.com. Recuperado de: https://www.syloper.com/blog/programacion/exportar-tabla-html-excel-xls-en-php/

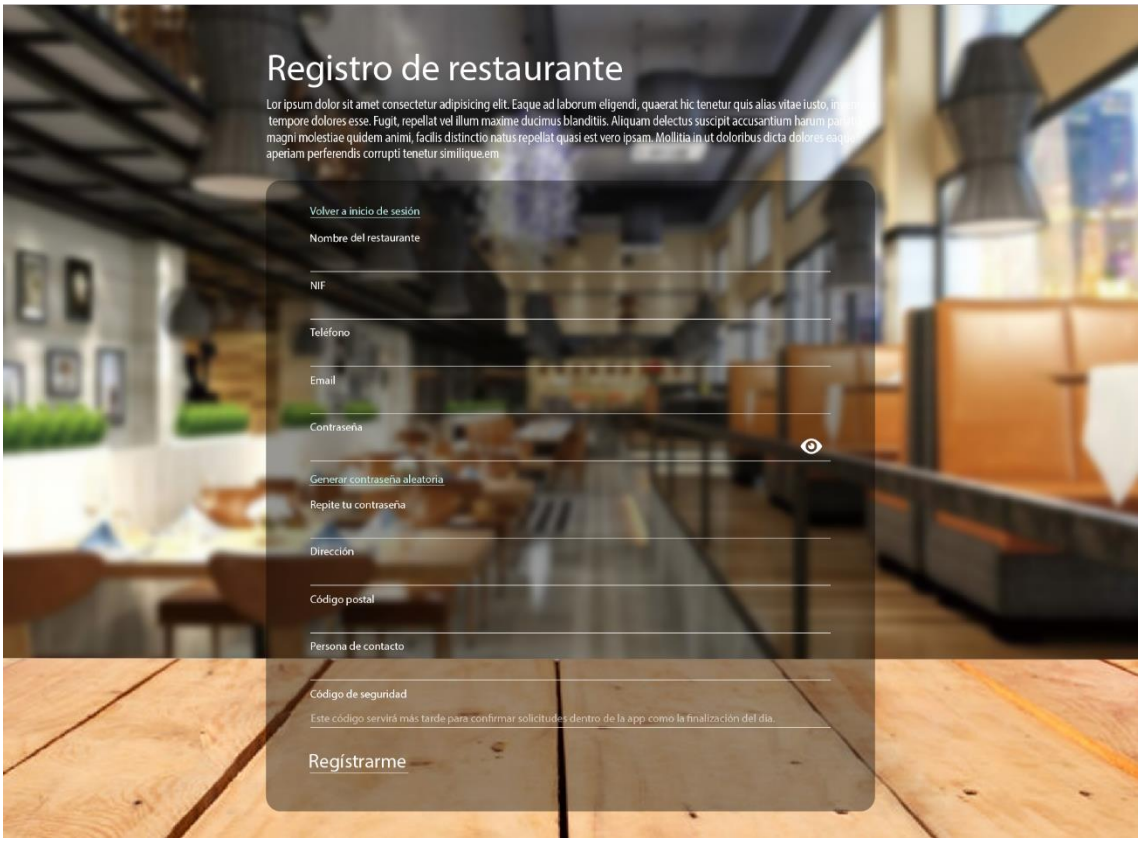
8. ANEXOS

8.1 ANEXO 1. TABLA DE IMÁGENES

8.2 ANEXO 2. MOCK-UP DE LA APLICACIÓN

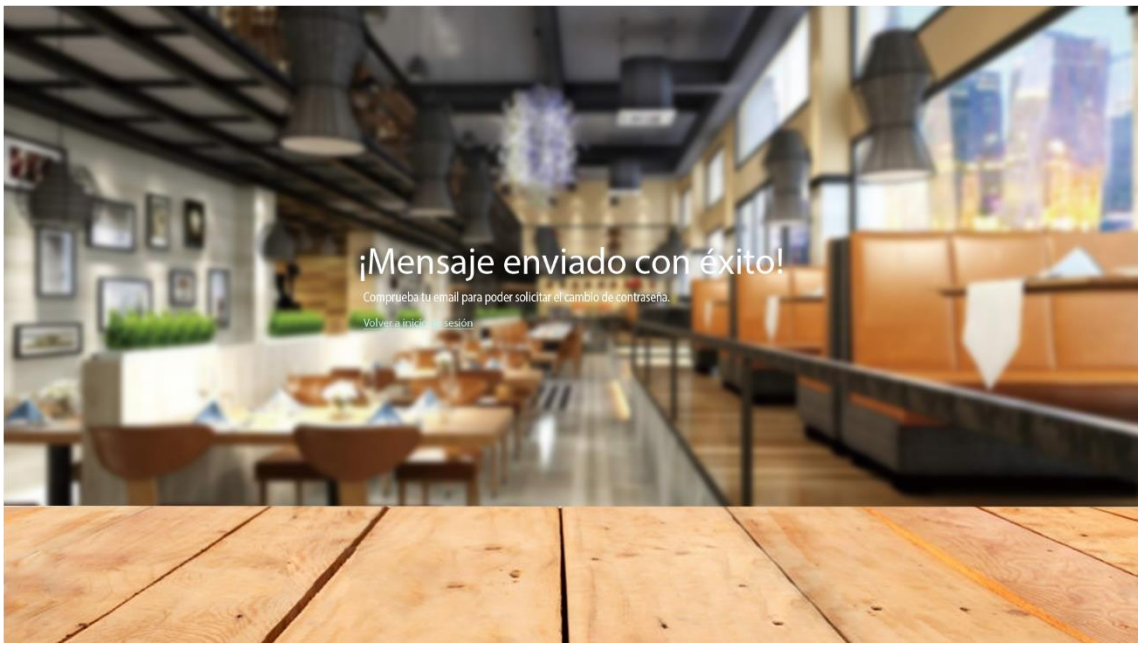


Captura 1 - Inicio de sesión

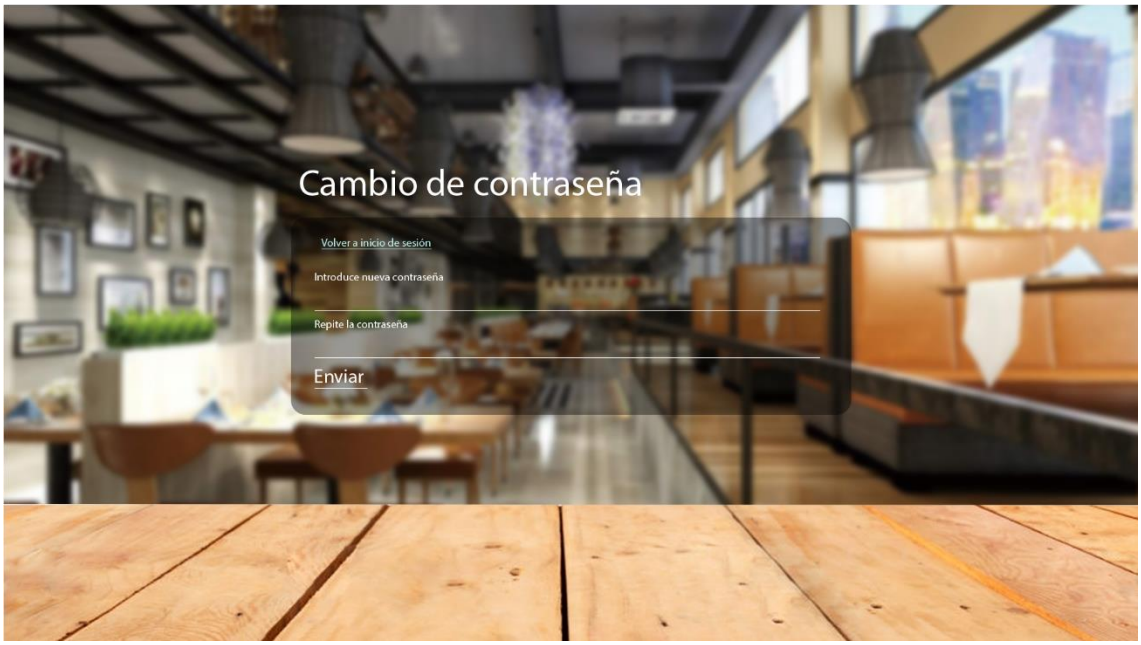


Captura 2 - Registro de restaurante.

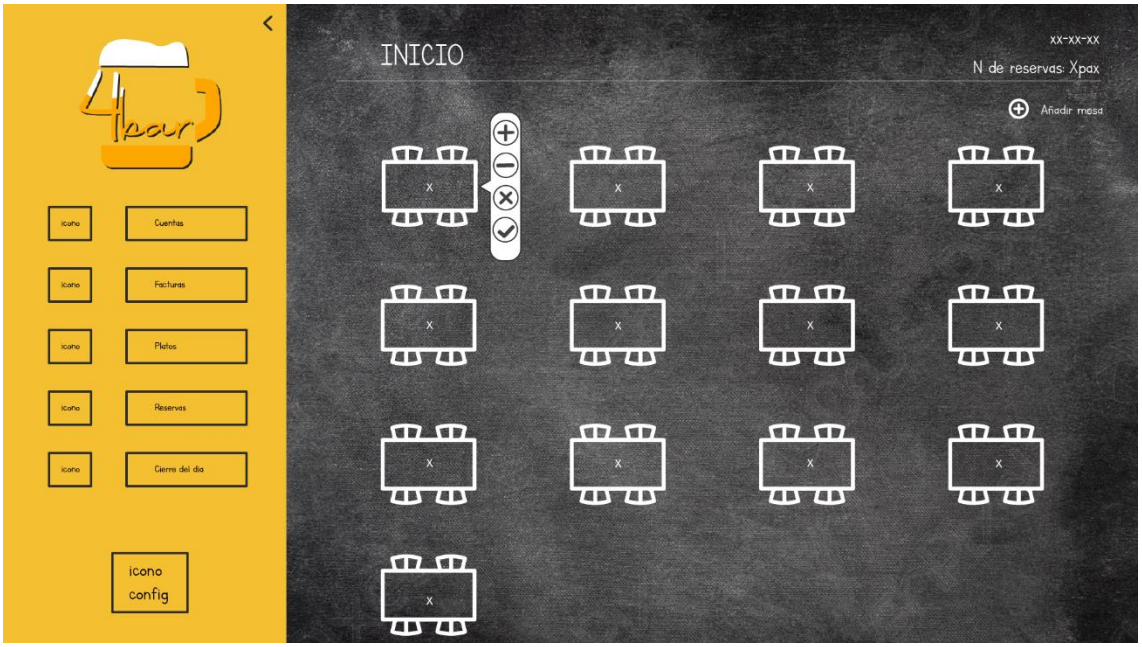
Captura 3 - Recuperación de contraseña.



Captura 4 - Confirmación de envio de mail para cambio de contraseña.

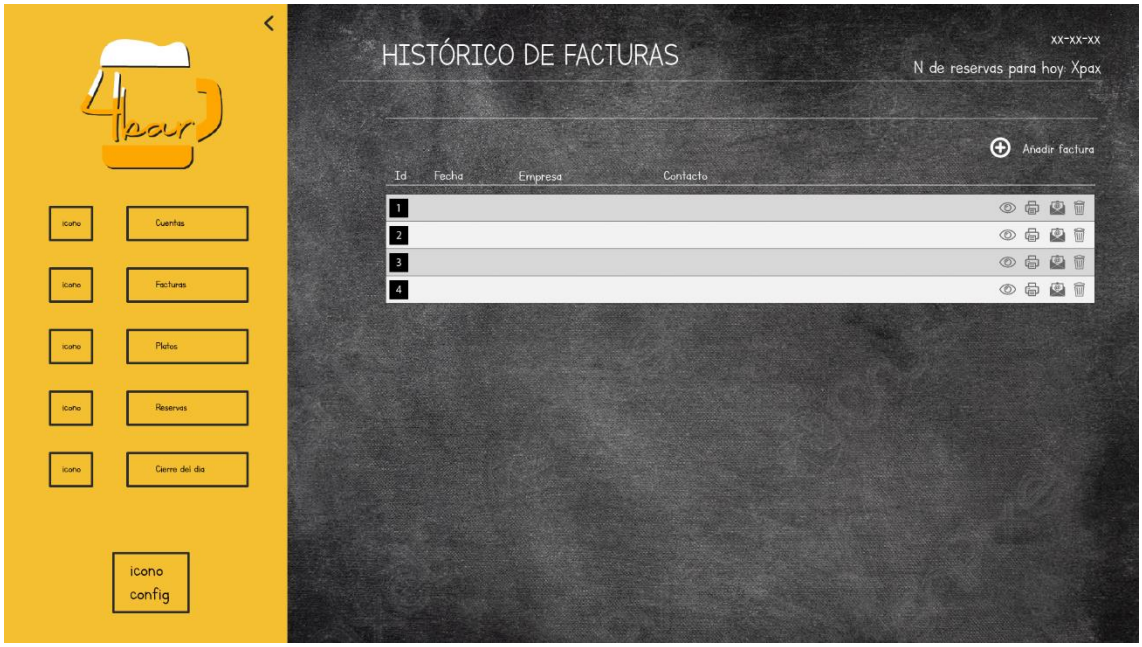


Captura 5 - Cambio de contraseña.

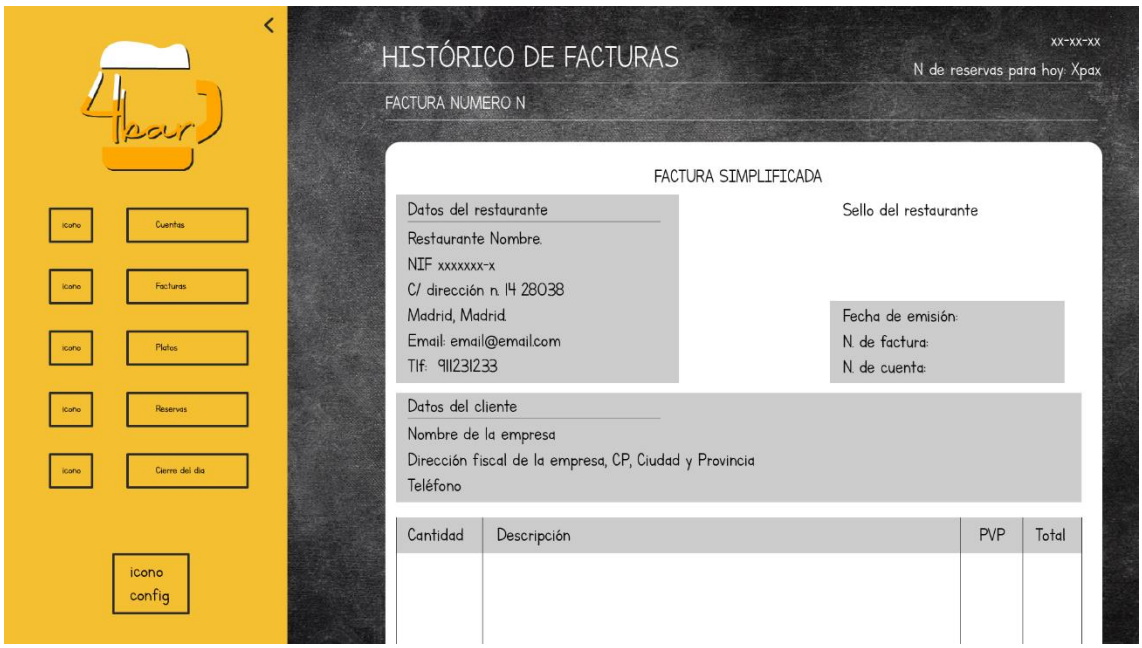


Captura 6 - Visualización de inicio de sesión con listado de mesas.

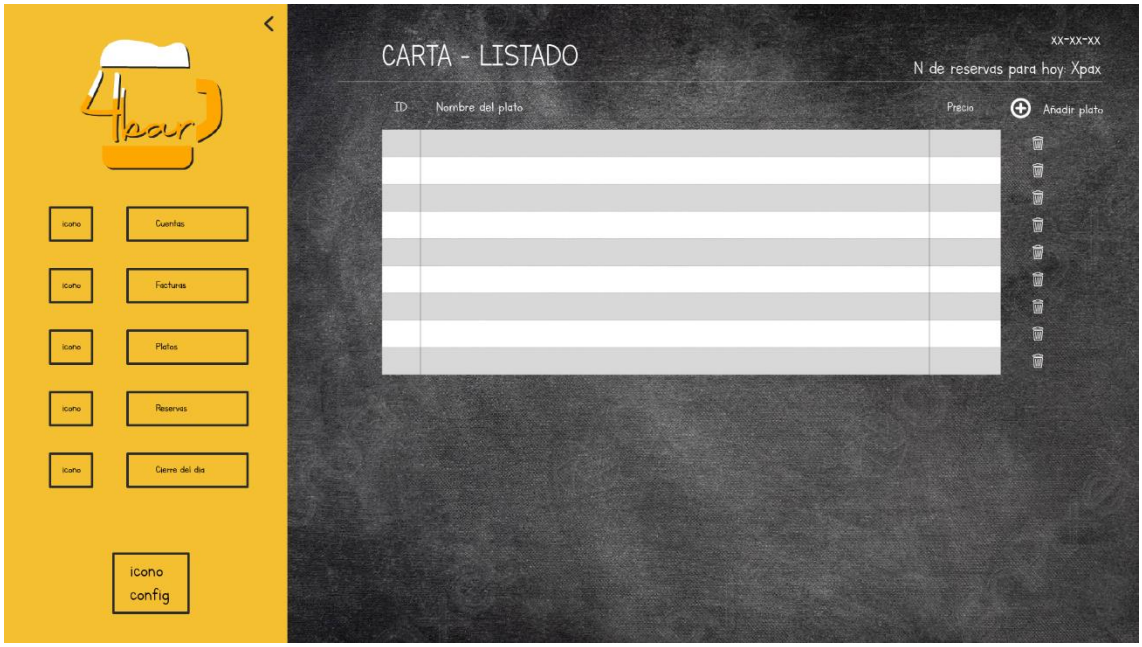
Captura 7 - Listado de cuentas finalizadas y no finalizadas.



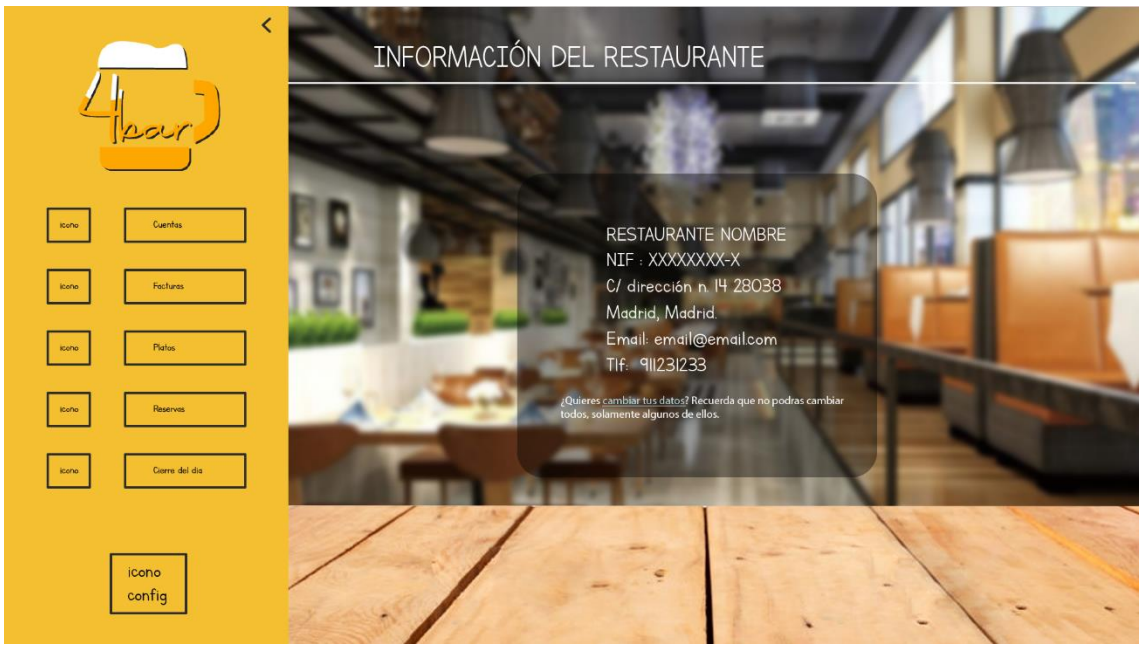
Captura 8 - Listado de facturas generadas.



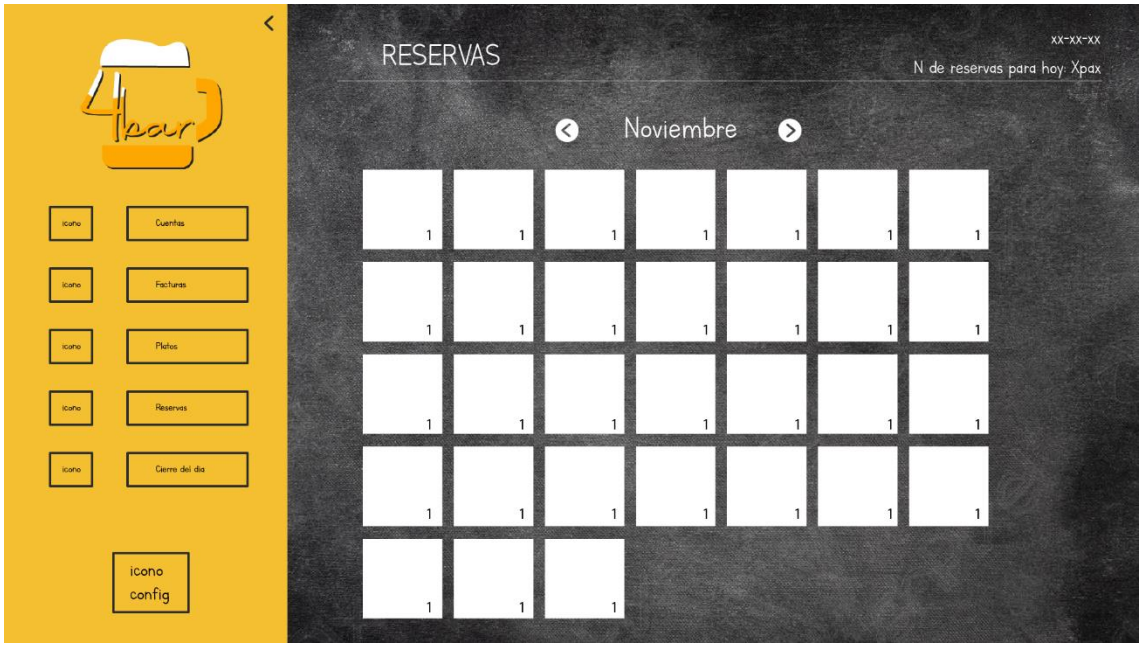
Captura 9 - Ejemplo de visualización de factura generada.



Captura 10 - Listado de platos en la carta.



Captura 11 - Información del restaurante.



Captura 12 - Visualización de las reservas.

Captura 13. Ejemplo de cierre de día.