

Práctica Final

2 DAW



Restaurante TPV

Pedro Sandoval Martínez

Contenido

[Spring 1 2](#_Toc183678907)

[Diseño y creación de la base de datos 2](#_Toc183678908)

[Esquema de Archivos 2](#_Toc183678909)

[Inicio de sesión y ruta de acceso 3](#_Toc183678910)

[Menús de opciones 4](#_Toc183678911)

[Spring 2 5](#_Toc183678912)

[Gestión y visualización de mesas 5](#_Toc183678913)

[Creación de pedidos 5](#_Toc183678914)

[Intento fallido con Javascript 5](#_Toc183678915)

[Funcionalidad final 6](#_Toc183678916)

[Crear Pedido 7](#_Toc183678917)

[Abrir mesas activas (listar pedido) 8](#_Toc183678918)

[Anotación Importante 8](#_Toc183678919)

[Mesa ocupada 9](#_Toc183678920)

[Cerrar mesa (pagar pedido) 10](#_Toc183678921)

[Spring 3 11](#_Toc183678922)

[Conexión con la impresora 11](#_Toc183678923)

[Tickets 12](#_Toc183678924)

[Auditoría de fechas 14](#_Toc183678925)

[**Spring 4: Funciones administrativas** 15](#_Toc183678926)

[Gestionar pedidos: 15](#_Toc183678927)

[**Gestionar productos:** 16](#_Toc183678928)

[**Gestionar camareros** 17](#_Toc183678929)

[Agregar camareros, suspender o eliminar. 17](#_Toc183678930)

[**Informes de ventas e historial de pedidos** 18](#_Toc183678931)

[Spring 5 19](#_Toc183678932)

[Cambios en el proyecto 19](#_Toc183678933)

[Referente a los pedidos: 20](#_Toc183678934)

[Añadir al pedido 21](#_Toc183678935)

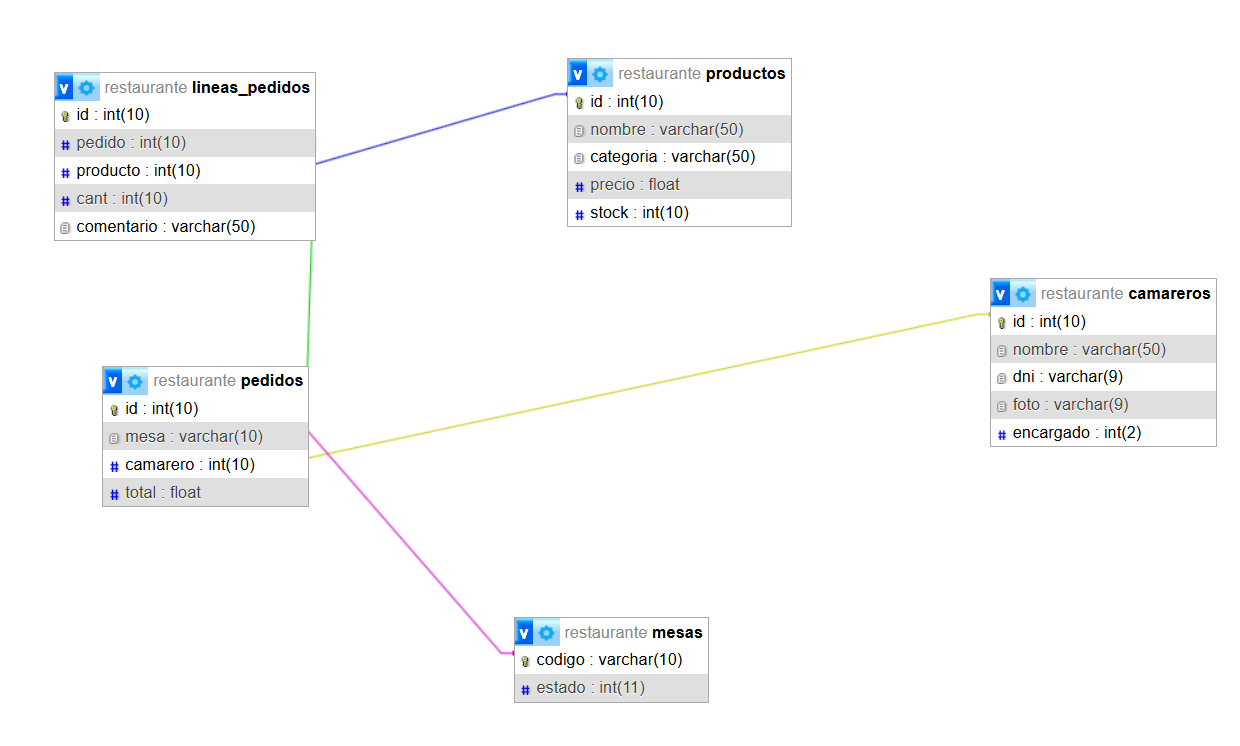
[Productos sin stock en el formulario 22](#_Toc183678936)

[Guía de uso 23](#_Toc183678937)

# Sprint 1

## Diseño y creación de la base de datos

Primer diseño de la base de datos. Centrada sobre todo en la tabla camareros, en la cual he comenzado a trabajar. En principio el id del camarero estará en el pedido para conocer quién registra el mismo.

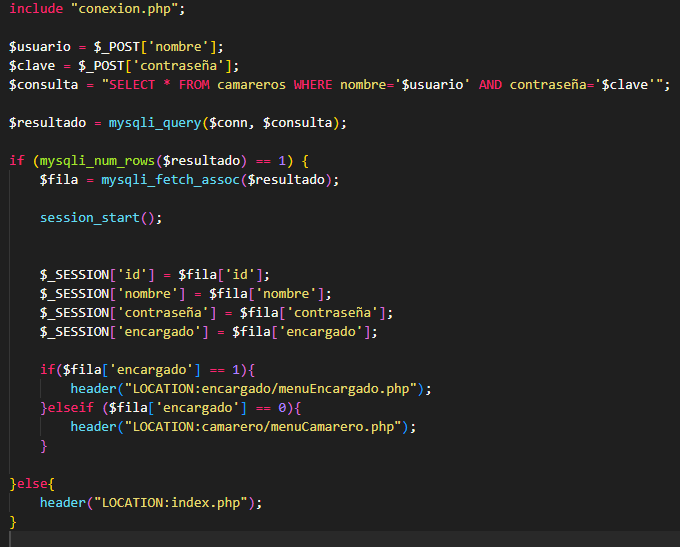


## Esquema de Archivos

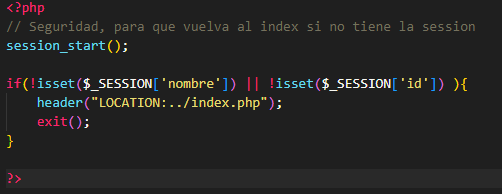
De momento he separado la visión de camarero y encargado.

Los archivos principales que conectan con la base de datos se encuentran en la carpeta raíz, al igual que los estilos del index (inicio de sesión) y el archivo que utilizo para validar la sesión.

## Inicio de sesión y ruta de acceso

Previamente insertado el primer camarero (admin), comprobamos que el usuario que intenta acceder se encuentre en la base de datos.

Si el usuario, es encargado (atributo encargado = 1), la página es redirigida al apartado de encargado. En cambio, si el atributo es 0, te lleva a la vista del camarero.

La seguridad del sitio web es comprobada por este archivo, que comprueba en cada una de las páginas la variable de sesión:

## Menús de opciones

En la vista de **camarero**, únicamente encontramos la opción de entrar al salón donde visualizamos las mesas, y dentro de ellas la toma de los pedidos.

El menú de **encargado** lo presentamos inicialmente así. Entre sus opciones encontramos la modalidad de añadir camareros, el listado de los mismos (sus datos), añadir nuevos productos y visualizarlos.

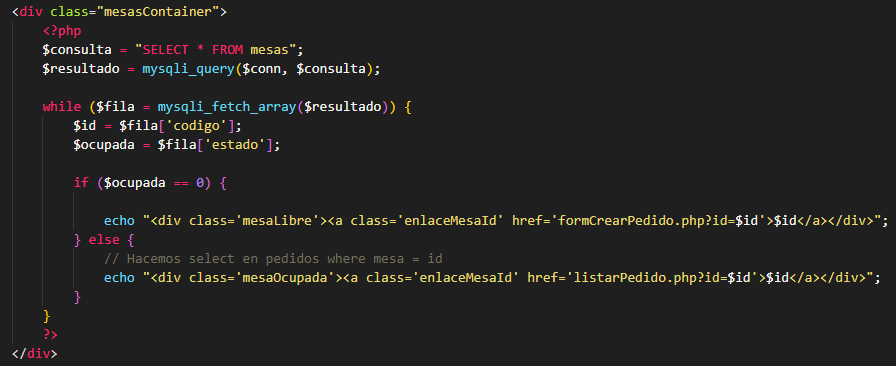


Además, introduzco un enlace para acceder a la vista de camareros por si el encargado lo necesitase.

# Sprint 2

## Gestión y visualización de mesas

Breve explicación: las mesas las utilizamos como contenedor del pedido, para localizarlo en el salón y poder acceder a las opciones (crear pedido y listarlo).

Archivo “salón.php” :

Mostramos las mesas que se encuentren en la base de datos, las que tengan estado 1 (ocupada) se mostrarán con una clase distinta, para que cambie el diseño y sea más fácil localizarlas en el salón.

## Creación de pedidos

### Intento fallido con Javascript

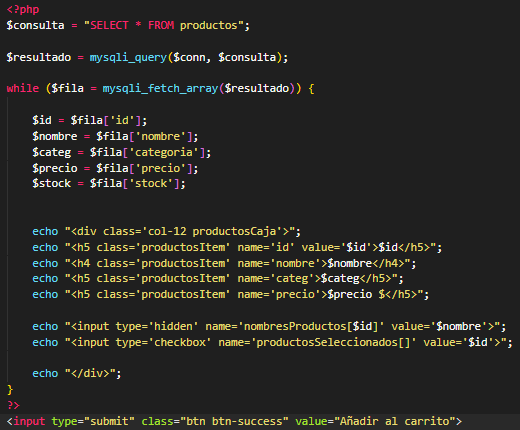
Aquí surge mi primer problema. La idea inicial era implementar parte de código javascript para hacer una “cesta” dinámica, sin recargar la página, puesto que sería un borrador antes de hacer la inserción del pedido.

* Ventajas: no es necesario guardar la información en la base de datos, mejoraría el rendimiento en un programa pesado.
* Problemática: la consulta de los productos almacenados debe ser transformada en un json, mostrarla entre etiquetas php, y transformarla una vez más para mostrarla en el carrito (y posteriormente debería ser tratada en formato php para la inserción).

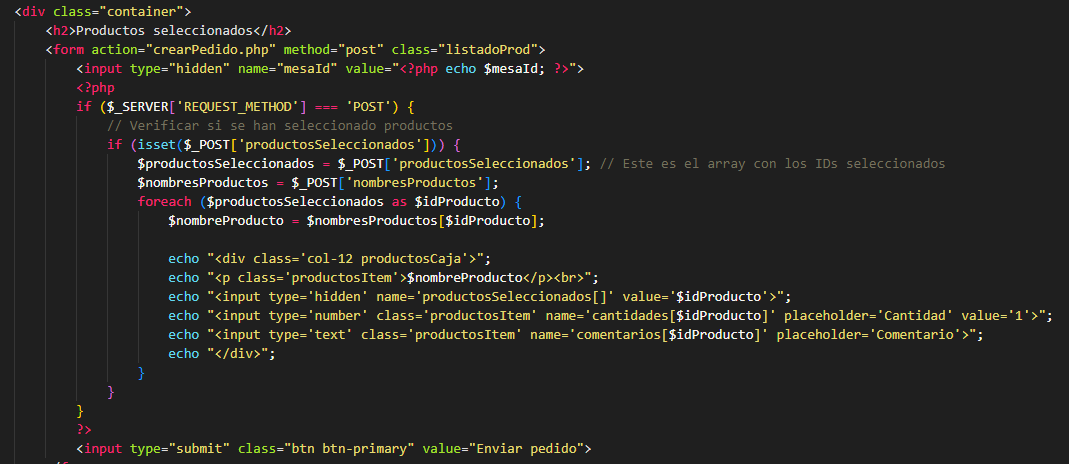
Por lo tanto, descartamos la idea de usar Javascript.

### Funcionalidad final

Una vez que entramos a la mesa, mostramos en un formulario la consulta de los productos. Los artículos que seleccionemos pasarán a otro formulario (líneas\_carrito).



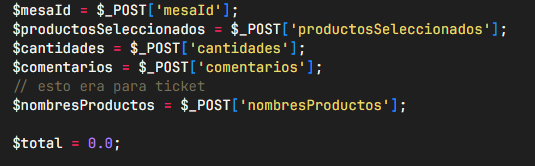
En el mismo archivo (formCrearPedido), hay otro formulario que usamos a modo de carrito, donde nos llevamos el array de productos seleccionados (con el checkbox).



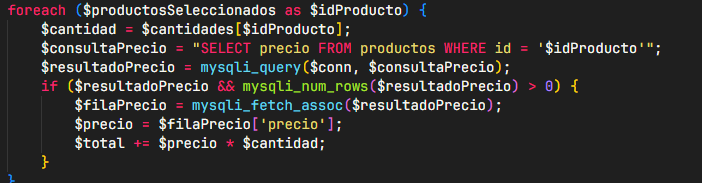
Este formulario es el que lleva a crearPedido, donde finalmente se hace la inserción en líneas\_pedidos. Para ello he tenido que llevarme el id del pedido, de la mesa y de los artículos seleccionados. Todo ello en forma de array con cada uno de los elementos ordenados por su id.

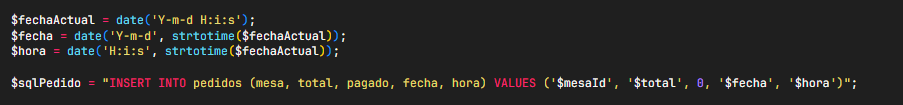
### Crear Pedido

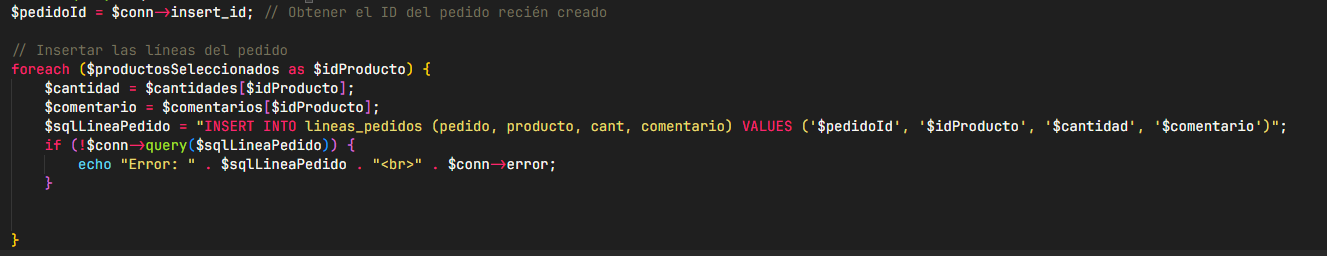
La cabecera del archivo contiene las variables más importantes con las que trabajamos:



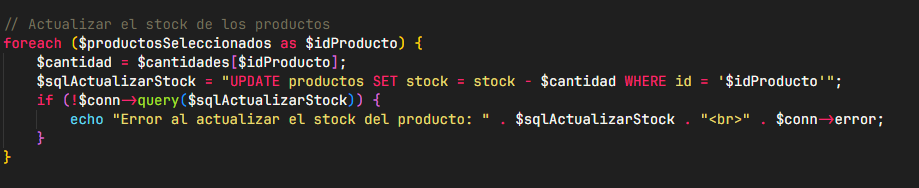
Calculamos total del pedido para su posterior inserción:



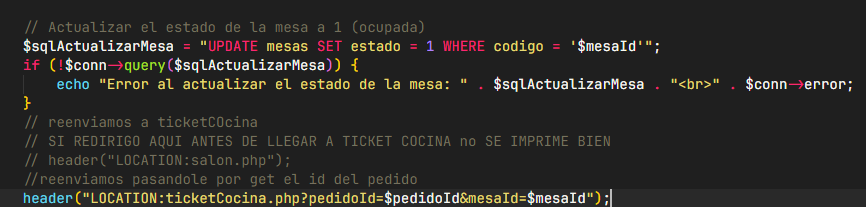
Guardamos fecha de creación y realizamos la inserción del pedido:

Y ahora sí, procedemos a la **inserción de las líneas del pedido**:

En este punto actualizo el stock de los productos, que nos será útil en el futuro para la gestión de productos, y que el camarero sepa a tiempo real si dispone de dicho artículo.

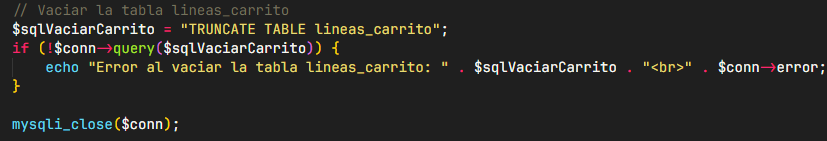


Y actualizamos el estado de la mesa:



Aquí ya vemos anotaciones de lo que se avecinaba en la siguiente sección, que explicaremos a continuación.

Lo último que modifica este archivo es la tabla del carrito, que debe ser vaciada una vez realizado el pedido:



## Abrir mesas activas (listar pedido)

### Anotación Importante

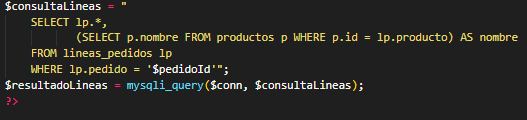
En todos los archivos relacionados con los pedidos, debemos comprobar que realmente nos llevamos la información adecuada, y trabajamos sobre la mesa y pedido que deseamos. Para ello en listar pedido encontramos estas líneas:



Así fui descartando errores y optimizando el código.

### Mesa ocupada

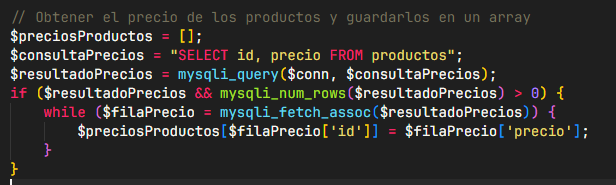
Una vez que la mesa aparece ocupada, si accedemos a ella desde el salón nos llevará a la página donde se listan las líneas del pedido que hay alojado en el mismo. Para ello he creado una consulta con producto cartesiano utilizando el *pedidoId* de arriba:



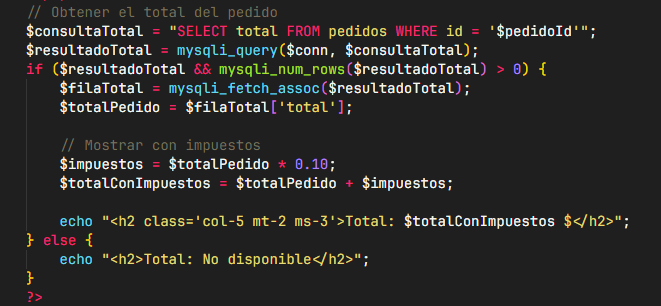
El producto cartesiano se utiliza para mostrar el listado de los productos en forma de tabla, a la que después según su categoría se le otorga un estilo distinto:



Para mostrar el precio del producto e insertar el total de la cuenta en el pedido, genero la siguiente consulta:



Consulta para obtener el total del pedido:

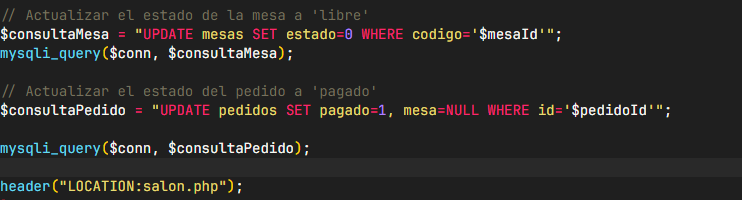


Lo utilizaremos para mostrar al final del listado el total de la cuenta, junto a los botones para imprimir el ticket y pagar la cuenta (y liberar la mesa).

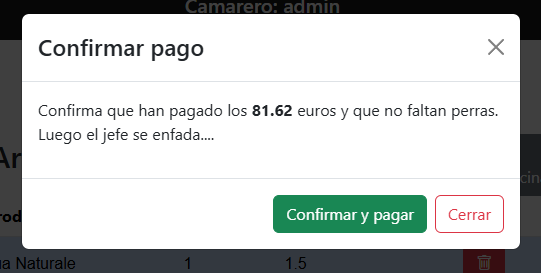
Ambos tickets los pospuse para el siguiente spring, puesto que no disponía de la impresora, lo veremos en el siguiente apartado.

## Cerrar mesa (pagar pedido)

Añado en este apartado la funcionalidad de liberar la mesa. La lógica es simple. Mi columna “mesa” admite nulos, así que cuando un pedido está pagado, este pedido asociado a una mesa deja de estarlo, y el atributo pasa a ser NULL.



Es una manera fácil de localizar los pedidos más tarde, puesto que no necesitamos la información de la mesa asociada al pedido para futuras auditorías.

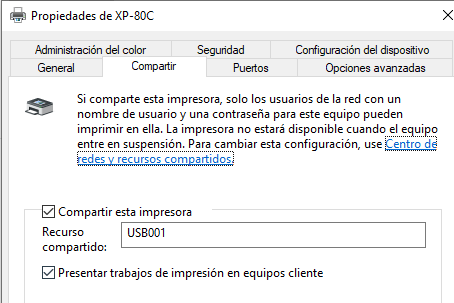
Cuando pulsamos el botón se abre un modal para confirmar la operación, y se ejecuta el código de arriba.

# Sprint 3

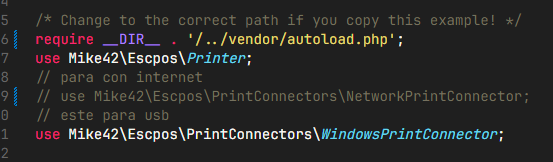
## Conexión con la impresora

Inicialmente hice las pruebas conectando la impresora (modelo XP-80C) por usb y posteriormente instalando los drivers pertinentes, en modo local con los propios archivos que nos facilitó el profesor D. Fernando Ureña.

El nombre que le otorgues es irrelevante, sin embargo, un paso indispensable es la configuración del puerto al que está conectada. En Windows (preferencias de impresora) tuve que activar el uso compartido de la misma, asignándole el puerto en el que será reconocida por nuestro servidor y por tanto en nuestros archivos PHP.



Por supuesto otro requisito indispensable es asegurarnos que el composer y el vendor previamente instalados en el repositorio están enlazados con el archivo donde creamos el ticket, que no haya errores en la ruta.

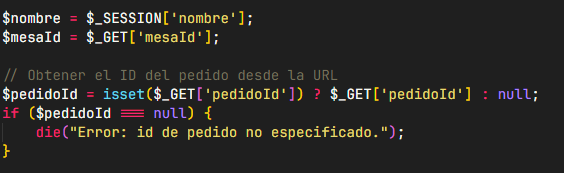


Esto incluye en nuestro archivo las librerías que necesitamos.

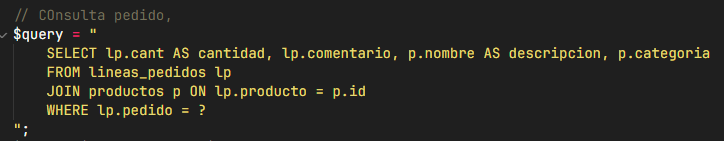
## Tickets

**Para cocina**

En el apartado donde creamos el pedido, ya vemos cómo se enlaza con este archivo, que tras la configuración inicial, recibe por get el id de la mesa donde se encuentra el pedido:

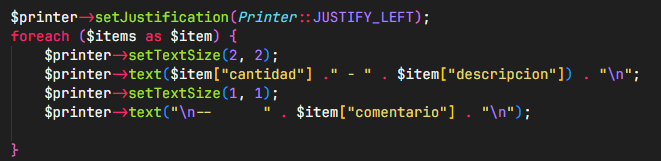


Hacemos la consulta utilizando join:

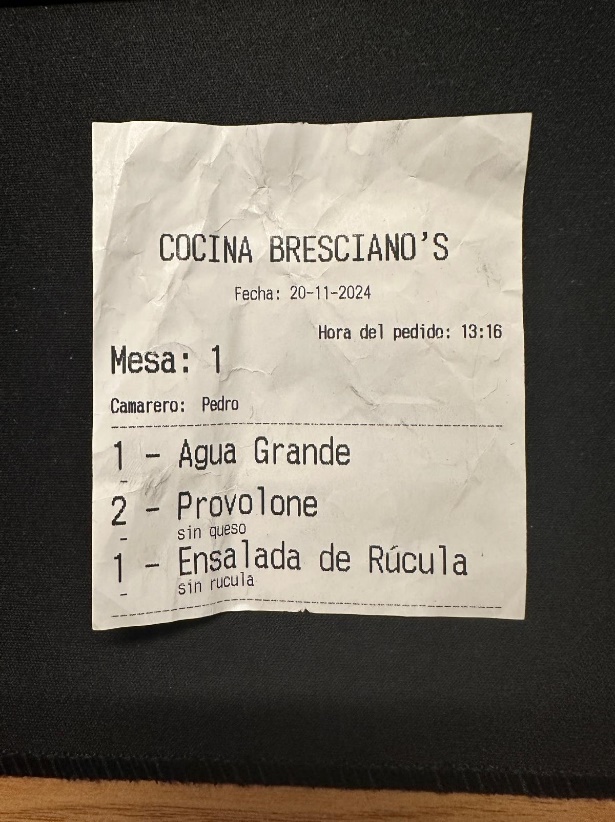


Ejecutamos la consulta y creamos array de productos (ítems) que deben imprimirse, teniendo en cuenta su descripción, cantidad y el comentario (si hay)

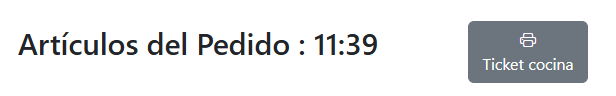
Tras la cabecera del ticket donde mostramos fecha, hora mesa y nombre del camarero, imprimimos array con saltos de línea para cada elemento:



Obteniendo como resultado este ticket:



Tras imprimirlo, nos devolverá al salón donde podremos comprobar y volver a reimprimir el ticket de cocina si lo necesitamos.

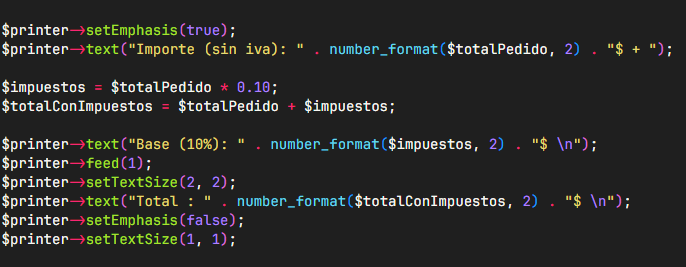


**Para cliente**

Utilizamos la misma configuración, las mismas variables, pero trabajamos con distintos datos. En el ticket del cliente no necesitamos los comentarios de las líneas del pedido.

En cambio, vamos a necesitar llevarnos del formulario los precios del producto y el total del precio para trabajar sobre él y conseguir el importe total con IVA.

Imprimimos de la misma forma, pero lo mostramos con otro formato. Los cálculos finales son los siguientes:



También creamos un CIF ficticio de la empresa y datos como la fecha de impresión, el camarero y la mesa, además de mostrar el total sin y con impuestos. Esto si o si debe aparecer en una factura simplificada como es un ticket.

Estos datos deberían ir almacenados en la base de datos del propio restaurante, por si sufriese cambios en el futuro. De momento lo dejamos como posible mejora.

Y el resultado es el siguiente:



En la interfaz, encontramos el botón de imprimir junto al modal donde se confirma el pedido, para poder hacer uso de la función cuantas veces necesitemos.



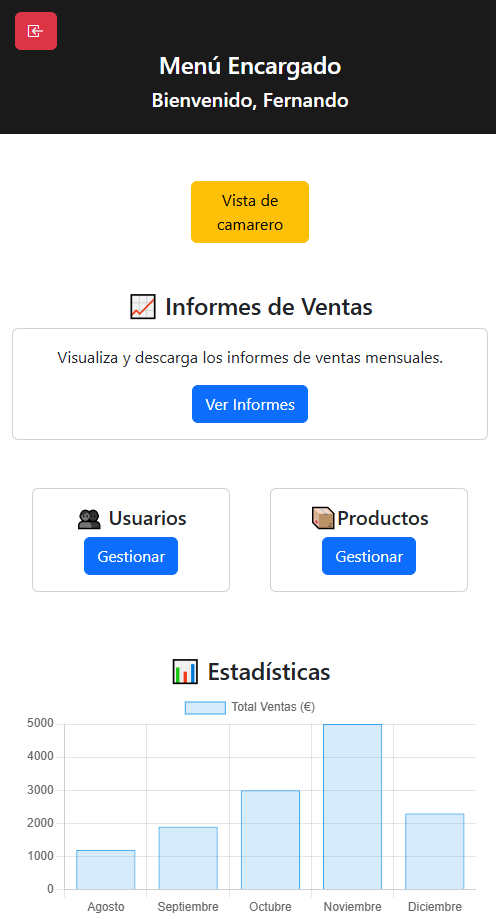
## Auditoría de fechas

Añado columna fecha para el pedido, y hacer búsqueda de pedidos de ese día. Esta opción ya está añadida en las capturas donde creamos el pedido, pero fue de los últimos cambios que realicé en el código.



Este código cambia la columna “Hora” que usábamos para mostrarla en el ticket, y almacena la fecha completa para permitir la auditoría de los pedidos. No hay cambios al crear el pedido.

**Sprint 4**



Para la vista del encargado he desarrollado el siguiente menú.

En el primer apartado podemos acceder a la sección de los camareros, para realizar labores relacionadas con los pedidos actuales (poder modificarlos y eliminarlos).

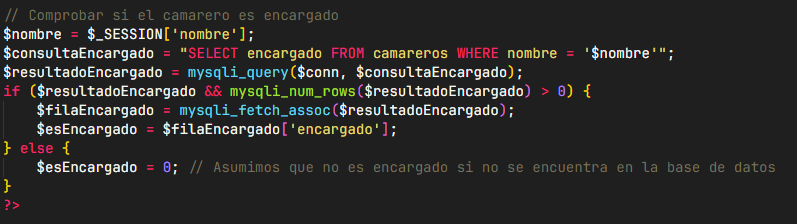
En informe de ventas podremos seleccionar la fecha en la que se realizaron pedidos, y nos generará una factura detalla con información sobre ese día

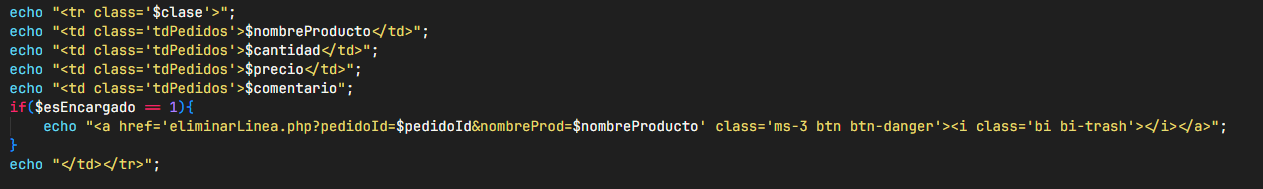
Se despliegan distintos modales donde podremos dirigirnos al apartado donde se muestran los usuarios y su gestión, y otro con un formulario para añadir nuevos.

Además de incluir el apartado de gestión de productos donde modificamos el stock, añadimos y eliminamos productos.

## Gestionar pedidos:

Cuando accedemos a la vista del camarero y posteriormente a una mesa con el pedido activo, el archivo comprueba que realmente es encargado.

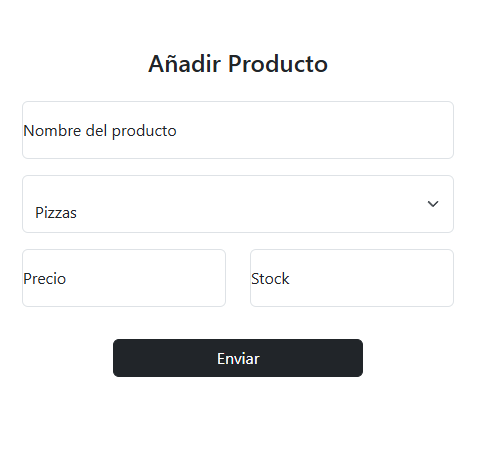


Esta opción nos permite generar junto al producto, un botón que elimine el mismo del pedido, accediendo por el enlace a la consulta donde se genera la modificación:

El código de eliminar línea va a utilizar esos dos parámetros para eliminar el producto. Es similar a los archivos que utilizo para eliminar camareros y productos, los cuales muestro abajo.

**Gestionar productos:**

En este apartado encontramos un formulario para insertar nuevos productos, según su categoría y el resto de parámetros necesarios (precio, stock).  La inserción en la base de datos requiere rellenar todos los campos.



Podremos ajustar el stock por defecto que tenga el producto, aunque más tarde sea modificado.

La consulta que realiza es la siguiente:



Junto al formulario encontramos el listado de productos, con la siguiente visual:

Los productos están organizados por categorías, cada una tiene una tabla con un color distinto.

Podemos restablecer stock de todos los productos (a 100), eliminar productos (pone stock a 0) o añadir (+) 50 de stock al producto.

Los archivos de modificación de stock contienen estas consultas:



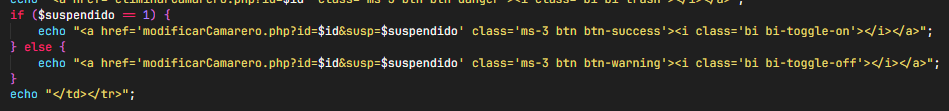


**Gestionar camareros**

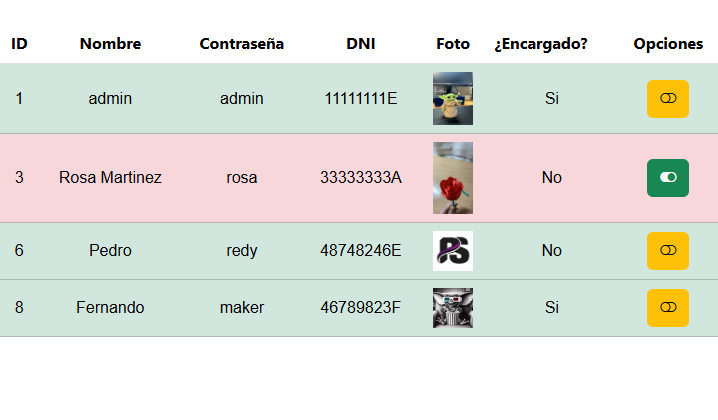
### Agregar camareros, suspender o eliminar.

Añadir camareros tiene la misma dinámica que el formulario de añadir productos, con la peculiaridad de que por defecto el usuario añadido está desactivado.



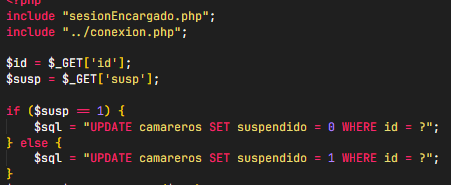
Así mostramos los botones para administrar el estado de los camareros:

El listado de camareros tiene esta interfaz. El fondo rojo aparece cuando el usuario está desactivado, y encontramos los botones de activar/desactivar a la derecha.



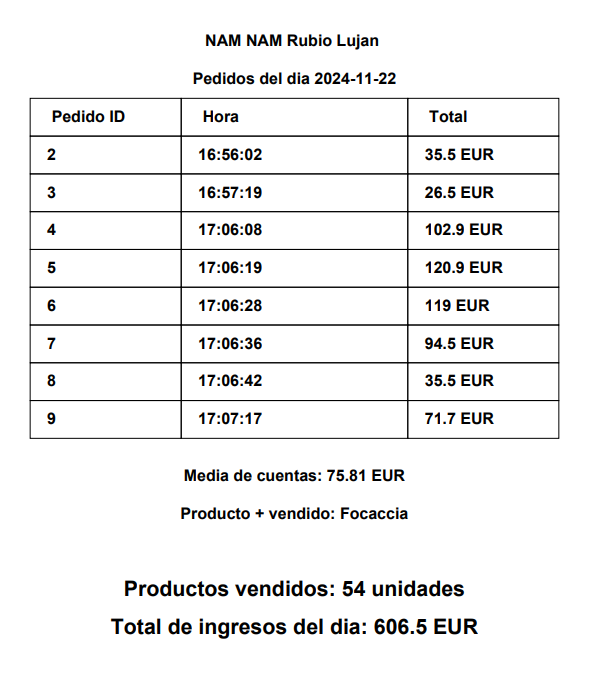
La opción de eliminar de la base de datos **no está contemplada**, para evitar problemas en futuras auditorías. Como mejora, propondría hacer otra tabla de antiguos camareros para guardar sus perfiles.

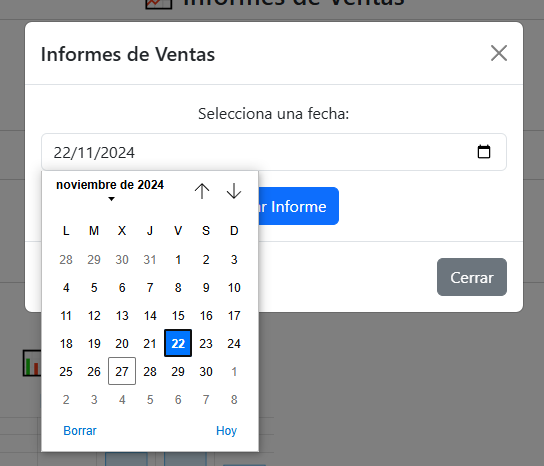
El archivo donde hacemos modificación comprueba primeramente qué vamos a modificar. Si está suspendido (1) lo activa, de lo contrario lo desactiva.



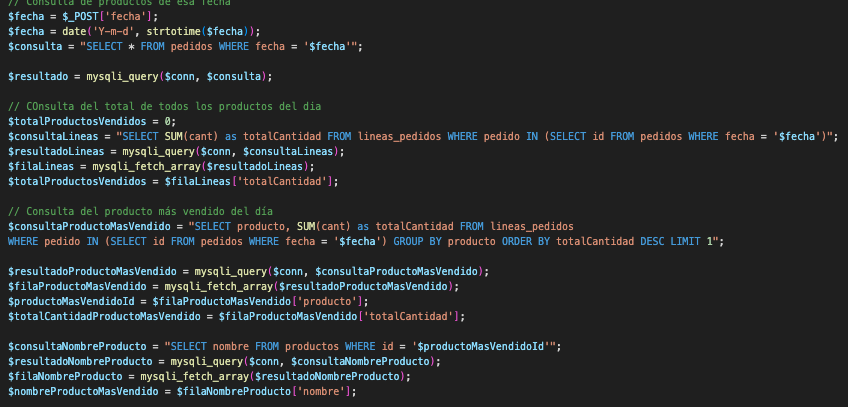
**Informes de ventas e historial de pedidos**

Volvemos al menú. Al pulsar en “ver informes” nos saltará un formulario donde podemos insertar el día del informe que estamos buscando. Automáticamente nos redirige al pdf generado, tomando por atributo indispensable la fecha elegida.





**Consultas a la base de datos que requiere esta funcionalidad:**

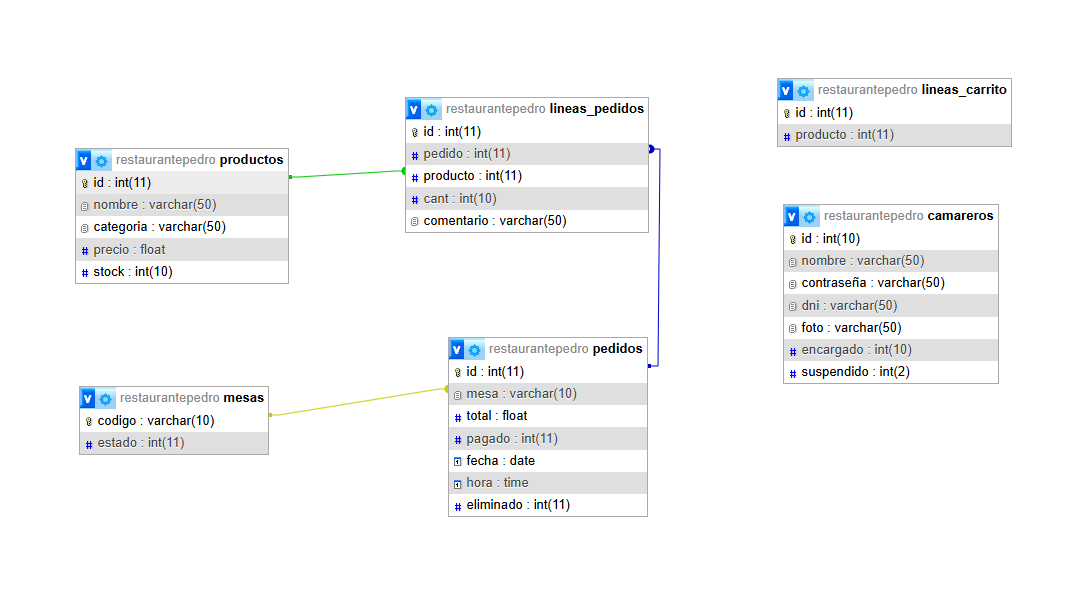
****

Realizamos subconsultas sencillas haciendo uso de lenguaje SQL y las relaciones entre tablas. Calculamos la media, el producto que más aparece en las líneas de pedido (más vendido), el total de productos que se han pedido y el total de ingresos (suma del total de todos los pedidos).

# Sprint 5

## Cambios en el proyecto

La base de datos a lo largo del proyecto ha sufrido cambios, culpa del mal diseño y planificación de proyecto y los requisitos del mismo.

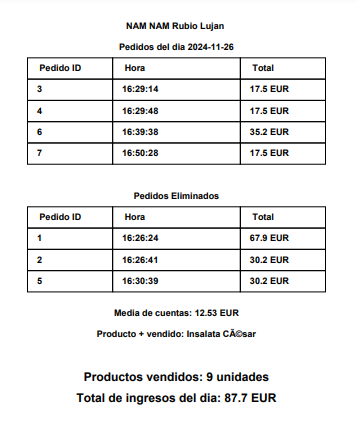


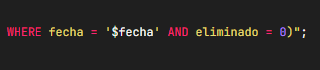
Como modificaciones importantes encontramos el atributo “eliminado” en pedidos, y “suspendido” en camareros, que ya hemos implementado en los apartados de gestión.

### Referente a los pedidos:

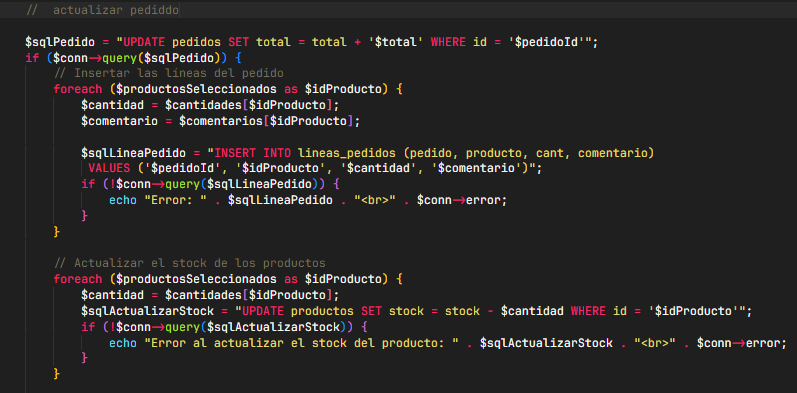
El encargado tiene opción de borrar artículos del pedido o (imaginemos en caso de error) eliminar el pedido al completo. Interesa llevar un registro de estos pedidos también, por eso lo añadimos en forma de atributo.



Esta opción la implementamos también en la auditoría de ventas, donde reflejamos que efectivamente hay pedidos eliminados. Estos no cuentan de cara a los productos vendidos ni al total de la caja, como se ve reflejado en el total de las cuentas.

Para no tener en cuenta los pedidos eliminados en los resultados, añadimos esta sencilla cláusula a las consultas:

### Añadir al pedido

Añado en este punto la opción de añadir más productos al pedido una vez creado, por un malentendido tuve que crear la funcionalidad más tarde. Creamos el archivo “añadirPedido”, que incluye las mismas inserciones que “crearPedido” salvo estas modificaciones:

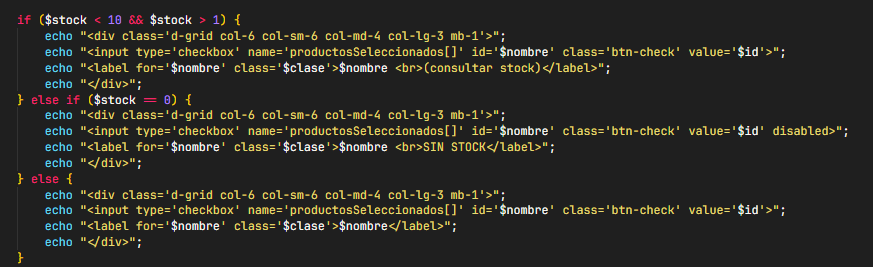
En este código observamos la actualización del total del pedido teniendo en cuenta los nuevos productos. Los añadimos a las líneas del pedido y además actualizamos el stock de la tabla productos para su correspondiente auditoría.

La interfaz sigue siendo igual, salvo que apreciamos el formulario debajo del listado del pedido para añadir esta funcionalidad.

### Productos sin stock en el formulario

Implementamos el stock de forma dinámica a la hora de tomar nota, esta funcionalidad es muy útil para los camareros, ya que saben a tiempo real los productos disponibles sin necesidad de preguntar o mirar en el listado.

Comprobamos el stock antes de mostrar el formulario con los productos, y ajustamos los parámetros con las condiciones:



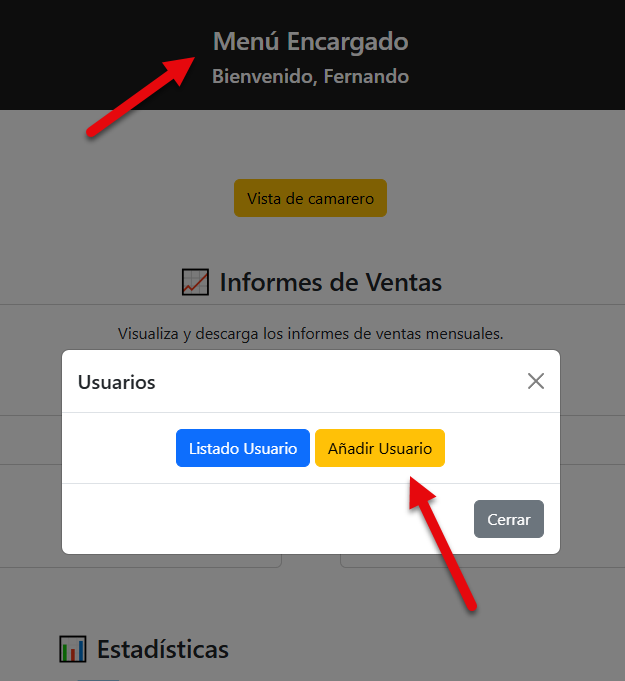
Si el stock del producto es menor que 10 (y mayor que 1), avisa que no quedan muchas unidades. Si directamente es 0, el input es “disabled”, no nos dejaría pedirlo. Finalmente, si no se cumple ninguna de esas condiciones, se muestra normalmente.

Hasta aquí las alteraciones en el código. Después de muchas horas de trabajo, decidí terminar aquí. Tras mucho esfuerzo e ideas, esto me ha servido de base para futuros proyectos, y me queda pendiente implementar todas aquellas mejoras en la aplicación para hostelería que quiero diseñar.

# Guía de uso

Vamos a realizar una guía de uso sencilla para los supuestos trabajadores que utilicen la aplicación. Primero comenzamos con el registro e inicio de sesión.

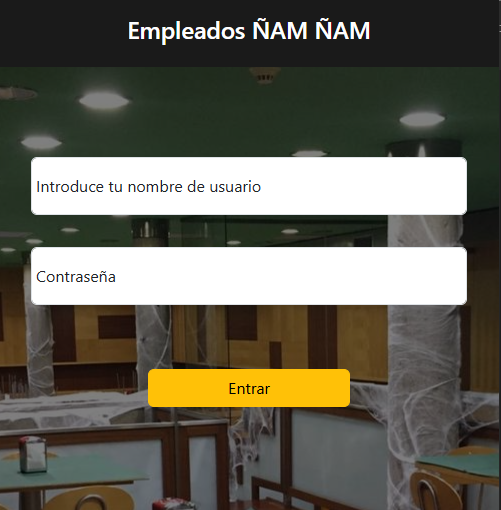
**Primer paso**

El encargado debe dar de alta a los camareros a través de su perfil.

Esto nos llevará al formulario que debe completarse con los datos del camarero.

**Segundo paso**

Acceder a la web principal e introducir tus credenciales.



**Menú del camarero**

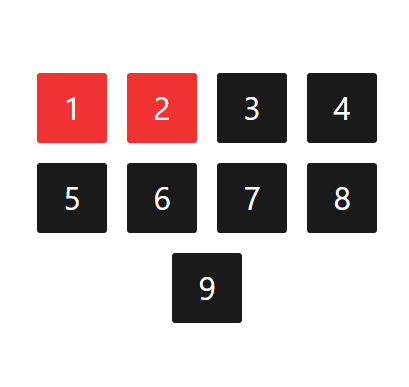
Nos aparecerán dos secciones, la primera de ellas de acceso al salón, donde tramitaremos las comandas, y la segunda a un listado de los productos y sus cantidades, por si requerimos consultar esa información antes del servicio.



Abajo podremos observar el día del servicio al que estamos accediendo.

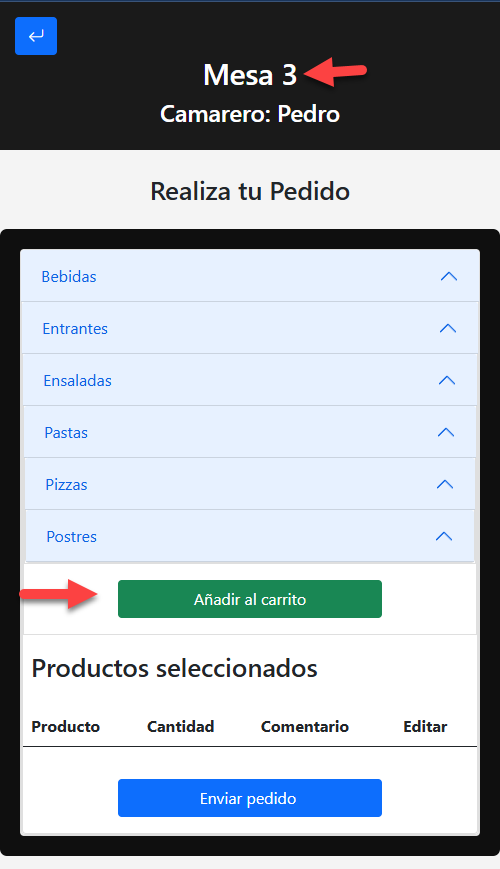
**Salón**

Una vez accedemos al salón, veremos el diseño de las mesas y su distribución. Las que se encuentren marcadas en rojo significa que están ocupadas (ya hay un pedido en ellas).

En cambio, las de color negro están libres. Hasta que no se realice el pago de la mesa o se elimine el pedido (solo el encargado puede eliminar), no se liberará la mesa.

**Toma de pedido**

Cuando accedemos a una mesa libre, nos aparecerá la interfaz para tomar un pedido.

Arriba nos aparece la mesa en la que estamos realizándolo.

Abajo tenemos menús desplegables divididos en secciones donde encontraremos los productos.

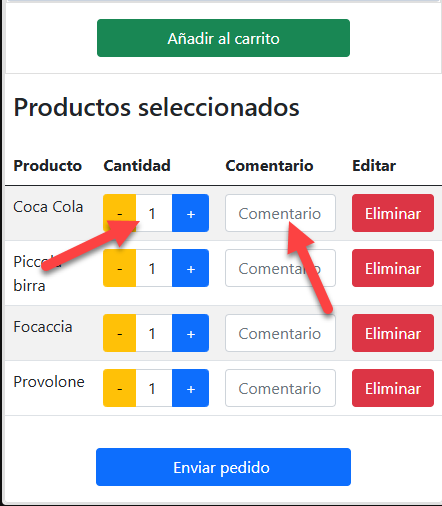
El botón verde añade las cosas al carrito, donde confirmaremos la cantidad del producto seleccionado y el comentario (si tiene).



Cuando pulsamos sobre una categoría nos aparecen los productos.

Atento a las indicaciones, nos indican si hay stock o si debemos consultar porque quedan pocos.

Cuando pulsamos sobre el botón del producto y añadimos al carrito, nos aparecerá el listado de productos seleccionados.

Aumentamos la cantidad si es necesario y añadimos un comentario.

Atención antes de pulsar el botón para enviar el pedido, confirma que el cliente ha pedido esos productos para evitar errores.

Recuerda que sólo el encargado puede eliminar los artículos una vez enviado a cocina.