Contenido

[Memoria del Proyecto Restaurante 3](#_Toc181979390)

[Sprint 1 - Fundamentos del Sistema y Roles 3](#_Toc181979391)

[1. Introducción 3](#_Toc181979392)

[2. Objetivos 3](#_Toc181979393)

[3. Estructura del Proyecto 4](#_Toc181979394)

[4. Proceso de Desarrollo 5](#_Toc181979395)

[Sprint 2 - Módulo Camarero 8](#_Toc181979396)

[1. Introducción 8](#_Toc181979397)

[2. Estructura del Módulo 8](#_Toc181979398)

[3. Funcionalidades Implementadas 9](#_Toc181979399)

[4. Mejoras en la Interfaz de Usuario 10](#_Toc181979400)

[5. Seguridad Implementada 11](#_Toc181979401)

[6. Funcionalidades de Gestión de Pedidos 12](#_Toc181979402)

[7. Mejoras en la Experiencia de Usuario 12](#_Toc181979403)

[8. Próximas Mejoras 13](#_Toc181979404)

[9. Conclusiones 13](#_Toc181979405)

[Sprint 3 - Gestión de Cuentas y Pagos 14](#_Toc181979406)

[1. Base de Datos 14](#_Toc181979407)

[2. Nuevas Funcionalidades 14](#_Toc181979408)

[3. Mejoras en la Interfaz 15](#_Toc181979409)

[4. Optimizaciones 16](#_Toc181979410)

[5. Archivos Modificados 16](#_Toc181979411)

[Próximos Pasos 16](#_Toc181979412)

# Memoria del Proyecto Restaurante

## 

## Sprint 1 - Fundamentos del Sistema y Roles

### 1. Introducción

Este proyecto se centra en el desarrollo de un sistema de gestión integral para un restaurante. El sistema abarca funcionalidades esenciales como la gestión de usuarios (con roles diferenciados), la administración de mesas, el procesamiento de pedidos y el control de productos. La modularidad del sistema permite adaptarlo a las necesidades específicas del restaurante, ofreciendo interfaces personalizadas para camareros y encargados.

### 2. Objetivos

* **Automatizar la gestión de pedidos:** Agilizar la toma de pedidos, la comunicación con la cocina y la generación de cuentas.
* **Controlar el inventario:** Mantener un registro actualizado de los productos disponibles, facilitando la gestión de stock y la previsión de compras.
* **Mejorar la atención al cliente:** Optimizar los tiempos de espera y reducir errores en las comandas.
* **Facilitar la administración del restaurante:** Proporcionar al encargado herramientas para la gestión de usuarios, el análisis de ventas y la generación de informes.

### 3. Estructura del Proyecto

La estructura de archivos y directorios del proyecto se organiza de la siguiente manera:

├── encargado  
│ └── index.php  
├── camarero  
│ ├── pedidos.php  
│ └── index.php  
├── BBDD  
│ ├── restaurante (con datos).sql  
│ └── restaurante.sql  
├── logout.php  
├── sesion.php  
├── login.php  
├── conexion.php  
├── index.php  
└── Instrucciones  
 └── leer.txt

#### 3.1. Descripción de Archivos

* **BBDD/restaurante.sql:** Define la estructura de la base de datos, incluyendo tablas, campos y relaciones.
* **BBDD/restaurante (con datos).sql:** Contiene la estructura de la base de datos con datos de ejemplo para facilitar las pruebas iniciales.
* **camarero/index.php:** Panel principal para camareros, con acceso a la gestión de mesas y pedidos.
* **camarero/pedidos.php:** Permite a los camareros tomar nuevos pedidos, modificar existentes y enviarlos a la cocina.
* **encargado/index.php:** Interfaz de administración para el encargado, con funcionalidades para la gestión de usuarios, productos, informes, etc.
* **conexion.php:** Centraliza la configuración de la conexión a la base de datos.
* **index.php:** Página principal del sistema, redirige a los usuarios según su rol.
* **login.php:** Gestiona la autenticación de usuarios, controlando el acceso al sistema.
* **logout.php:** Cierra la sesión del usuario.
* **sesion.php:** Implementa la gestión de sesiones para mantener el estado del usuario durante su interacción con el siste

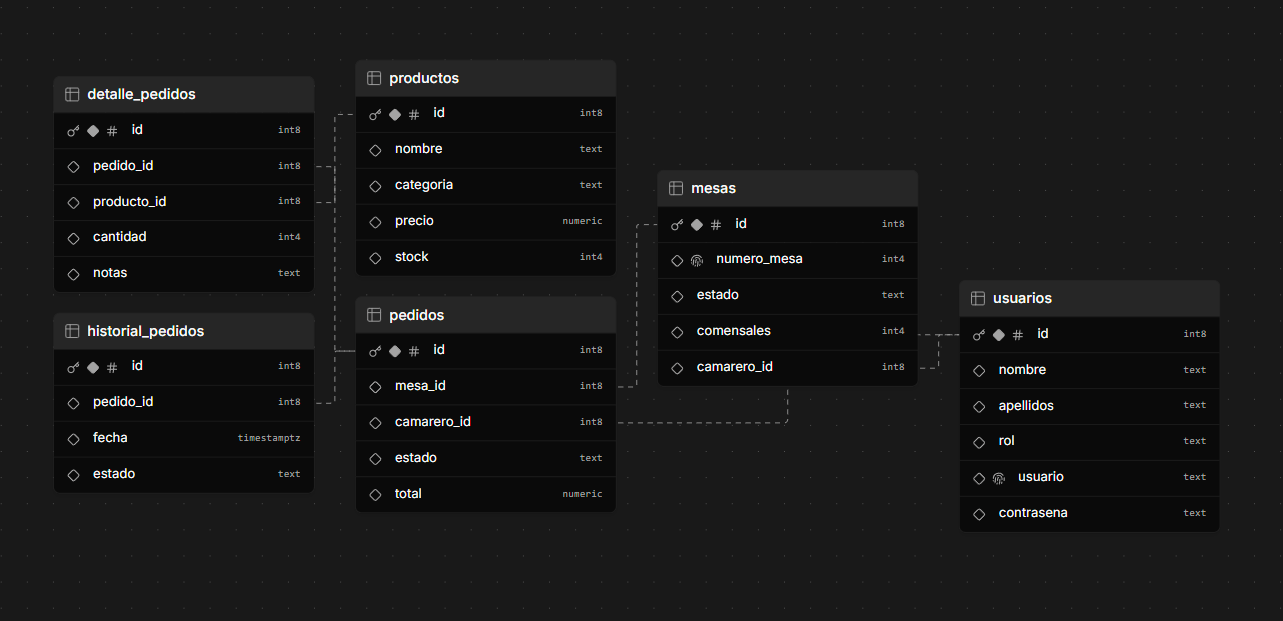
### 4. Proceso de Desarrollo

El desarrollo del proyecto se llevó a cabo siguiendo las siguientes etapas:

#### 4.1. Diseño de la Base de Datos

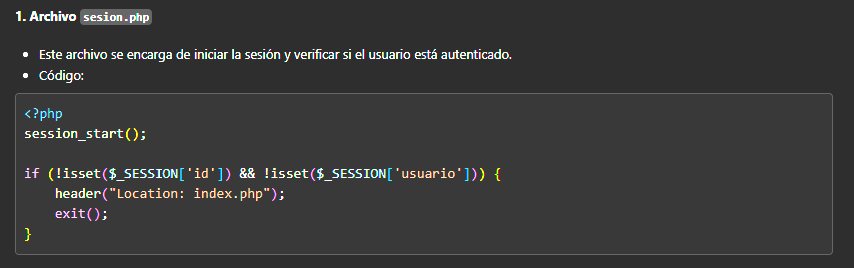
Se definieron las entidades necesarias para el sistema (usuarios, mesas, pedidos, productos) y se establecieron las relaciones entre ellas. Se utilizaron claves primarias y foráneas para asegurar la integridad referencial de los datos.

Primer esquema relacional:

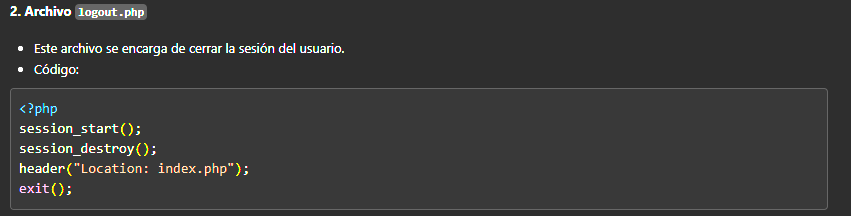


#### 4.2. Implementación del Sistema de Autenticación

Se implementó un sistema de autenticación de usuarios para restringir el acceso a las funcionalidades del sistema.

**Ejemplo de código PHP (sesion.php):**

**Ejemplo de código PHP (logout.php):**



**Ejemplo de código PHP (login.php):**

#### 4.3. Desarrollo de la Interfaz de Usuario

Se diseñaron interfaces intuitivas y fáciles de usar para cada tipo de usuario (camarero y encargado). Se utilizaron tecnologías web como HTML, CSS y JavaScript para crear las interfaces.

#### 4.4. Implementación de la Lógica de Negocio

Se programaron las funcionalidades del sistema, como la toma de pedidos, la gestión de mesas, el control de stock, etc. Se utilizó PHP y consultas SQL para interactuar con la base de datos.

#### 4.5. Pruebas y Validación

Se realizaron pruebas exhaustivas para asegurar el correcto funcionamiento del sistema y la integridad de los datos. Se probaron diferentes escenarios y casos de uso para identificar y corregir posibles errores.

## Sprint 2 - Módulo Camarero

### 1. Introducción

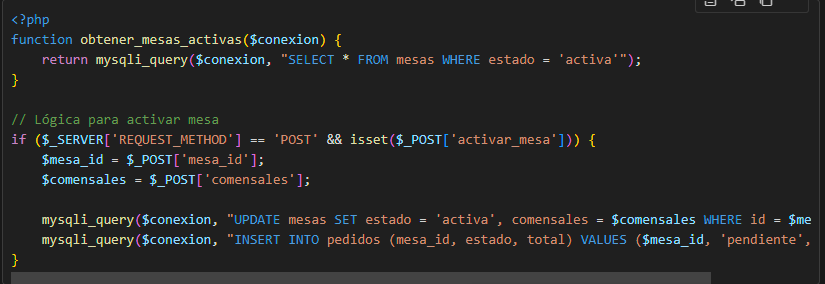
Durante el Sprint 2, se ha desarrollado y mejorado el módulo de camarero del sistema de gestión de restaurante. Este módulo permite la gestión eficiente de mesas, pedidos y productos, con una interfaz responsive y fácil de usar.

### 2. Estructura del Módulo

#### 2.1 Organización de Archivos

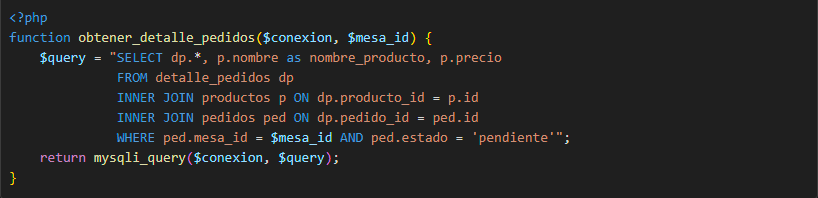
### 3. Funcionalidades Implementadas

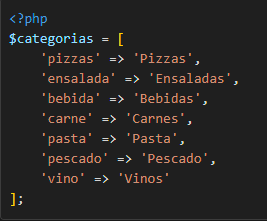
#### 3.1 Gestión de Mesas

****Ejemplo de código para la gestión de mesas activas:

#### 3.2 Gestión de Pedidos

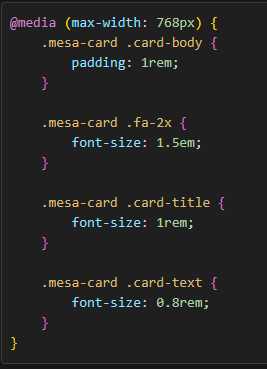
Se implementó un sistema completo de gestión de pedidos con las siguientes características:

**Código de productos añadidos:**

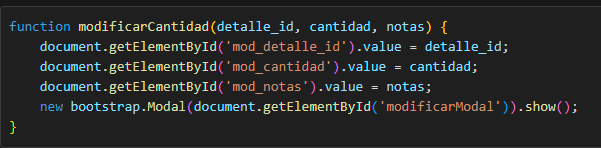
**Sistema de categorías de productos:**

### 4. Mejoras en la Interfaz de Usuario

#### 4.1 Diseño Responsive

****

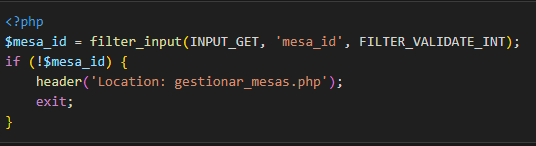
#### 4.2 Interactividad con JavaScript

****

### 5. Seguridad Implementada

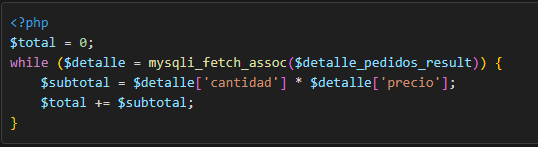
#### 5.1 Validación de Sesiones

#### 5.2 Sanitización de Datos

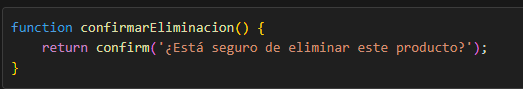


### 6. Funcionalidades de Gestión de Pedidos

#### 6.1 Cálculo de Totales



#### 6.2 Sistema de Confirmación



### 7. Mejoras en la Experiencia de Usuario

#### 7.1 Notificaciones y Feedback

* Mensajes de confirmación para acciones importantes
* Indicadores visuales de estado
* Transiciones suaves en la interfaz

#### 7.2 Optimización de Rendimiento

* Consultas SQL optimizadas
* Carga asíncrona de datos
* Minimización de recargas de página

### 8. Próximas Mejoras

* Implementación de sistema de notificaciones en tiempo real
* Mejora en la gestión de inventario
* Implementación de estadísticas de ventas
* Sistema de reservas

### 9. Conclusiones

El Sprint 2 ha logrado implementar un sistema robusto y fácil de usar para la gestión de mesas y pedidos, con especial énfasis en la experiencia de usuario y la adaptabilidad a dispositivos móviles. Las mejoras en la interfaz y la implementación de funcionalidades clave han resultado en un sistema más eficiente y fácil de usar para los camareros. He tenido que ajustar la base de datos de nuevo y tener que quitar el “id” de las mesas ya que me hacía tener que comprobar el camarero con su “id” y me creaba demasiados errores, por lo demás la base de datos creo que hacerte con ella y estoy teniendo muy buenos resultados.

## Sprint 3 - Gestión de Cuentas y Pagos

### 1. Base de Datos

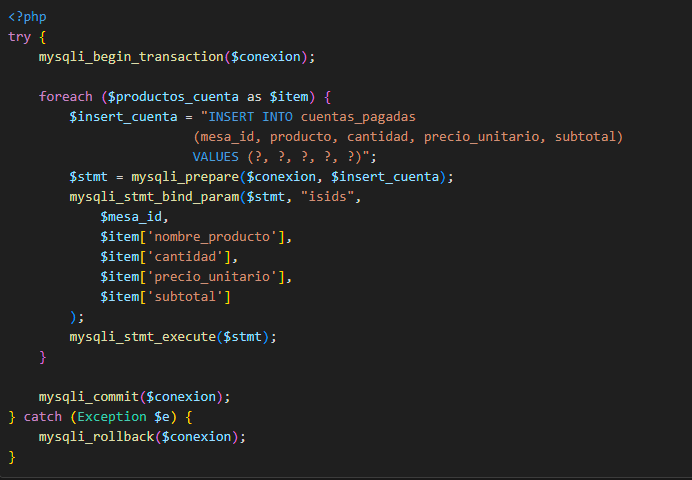
* Creación de nueva tabla cuentas\_pagadas para almacenar el historial de pagos
* Eliminación de tablas innecesarias (temp\_ticket, historial\_pedidos)
* Optimización de la estructura de la tabla cuenta actual

### 2. Nuevas Funcionalidades

#### 2.1 Gestión de Cuentas

* Implementación de vista detallada de cuenta por mesa
* Cálculo automático de subtotales y total
* Interfaz responsive para visualización de productos en la cuenta
* Funcionalidad para procesar pagos

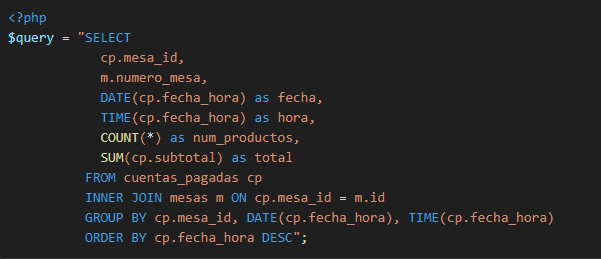
Ejemplo de procesamiento de pago:



#### 2.2 Historial de Cuentas

* Nueva sección de historial de cuentas pagadas
* Visualización de cuentas agrupadas por mesa y fecha
* Modal para ver detalles de cuentas históricas
* Sistema de consulta mediante AJAX para cargar detalles

Ejemplo de consulta para obtener historial:



#### 2.3 Dashboard del Camarero

* Actualización del panel principal con últimas cuentas pagadas
* Widget de resumen de actividad reciente
* Mejora en la navegación entre secciones

### 3. Mejoras en la Interfaz

#### 3.1 Diseño Responsive

* Optimización para dispositivos móviles
* Implementación de tablas responsive
* Mejora en la visualización de datos en pantallas pequeñas

#### 3.2 Experiencia de Usuario

* Nuevo sistema de navegación más intuitivo
* Botones de acción contextuales
* Feedback visual mejorado para acciones importantes

### 4. Optimizaciones

* Implementación de prepared statements para consultas seguras
* Mejora en el manejo de transacciones para pagos
* Optimización de consultas SQL para mejor rendimiento

### 5. Archivos Modificados

1. camarero/cuenta.php - Nueva implementación
2. camarero/cuentas\_pagadas.php - Nueva implementación
3. camarero/obtener\_detalle\_cuenta.php - Nuevo archivo
4. camarero/index.php - Actualizado con historial
5. BBDD/restaurante.sql - Actualización de estructura

### Próximos Pasos

1. Implementar sistema de tickets PDF
2. Añadir filtros en el historial de cuentas
3. Implementar estadísticas de ventas
4. Mejorar la gestión de errores
5. Añadir funcionalidad de búsqueda en el historial

Este sprint se ha centrado en mejorar la gestión financiera y el seguimiento de pagos, proporcionando herramientas más robustas para los camareros y una mejor experiencia de usuario.