Memoria del Proyecto Restaurante

### 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 sistema.

### 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.

**Ejemplo de código SQL (restaurante.sql):**

SQL

CREATE TABLE usuarios (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 nombre VARCHAR(255) NOT NULL,  
 usuario VARCHAR(255) NOT NULL UNIQUE,  
 contrasena VARCHAR(255) NOT NULL,  
 rol ENUM('camarero', 'encargado') NOT NULL  
);  
  
CREATE TABLE mesas (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 numero INT NOT NULL UNIQUE,  
 capacidad INT NOT NULL,  
 estado ENUM('libre', 'ocupada') NOT NULL DEFAULT 'libre'  
);

#### 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):**

PHP

<?php  
session\_start();  
  
function verificar\_sesion() {  
 if (!isset($\_SESSION['usuario'])) {  
 header("Location: login.php");  
 exit();  
 }  
}  
?>

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

PHP

<?php  
session\_start();  
session\_destroy();  
header("Location: login.php");  
exit();  
?>

#### 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.

**Ejemplo de código PHP (camarero/pedidos.php):**

PHP

<?php  
require\_once '../conexion.php';  
verificar\_sesion();  
  
// Obtener el ID de la mesa desde la URL  
if (isset($\_GET['mesa'])) {  
 $id\_mesa = $\_GET['mesa'];  
  
 // Consulta para obtener los productos  
 $sql = "SELECT \* FROM productos";  
 $result = $conn->query($sql);  
} else {  
 // Redirigir si no se proporciona el ID de la mesa  
 header("Location: index.php");  
 exit();  
}  
?>

#### 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.

**Memoria 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**

camarero/

├── index.php                  # Panel principal

├── gestionar\_mesas.php        # Gestión de mesas

├── gestionar\_pedido.php       # Gestión de pedidos

├── seleccionar\_producto.php   # Selección de productos

├── productos\_anadidos.php     # Lista de productos en pedido

└── js/

    └── scripts.js            # Funciones JavaScript

**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:**



<?php

function obtener\_detalle\_pedidos($conexion, $mesa\_id) {

    $query = "SELECT dp.\*, p.nombre as nombre\_producto, p.precio

              FROM detalle\_pedidos dp

              INNER JOIN productos p ON dp.producto\_id = p.id

              INNER JOIN pedidos ped ON dp.pedido\_id = ped.id

              WHERE ped.mesa\_id = $mesa\_id AND ped.estado = 'pendiente'";

    return mysqli\_query($conexion, $query);

}

**Sistema de categorías de productos:**



<?php

$categorias = [

    'pizzas' => 'Pizzas',

    'ensalada' => 'Ensaladas',

    'bebida' => 'Bebidas',

    'carne' => 'Carnes',

    'pasta' => 'Pasta',

    'pescado' => 'Pescado',

    'vino' => 'Vinos'

];

**4. Mejoras en la Interfaz de Usuario**

**4.1 Diseño Responsive**



@media (max-width: 768px) {

    .mesa-card .card-body {

        padding: 1rem;

    }

    .mesa-card .fa-2x {

        font-size: 1.5em;

    }

    .mesa-card .card-title {

        font-size: 1rem;

    }

    .mesa-card .card-text {

        font-size: 0.8rem;

    }

}

**4.2 Interactividad con JavaScript**



function modificarCantidad(detalle\_id, cantidad, notas) {

    document.getElementById('mod\_detalle\_id').value = detalle\_id;

    document.getElementById('mod\_cantidad').value = cantidad;

    document.getElementById('mod\_notas').value = notas;

    new bootstrap.Modal(document.getElementById('modificarModal')).show();

}

**5. Seguridad Implementada**

**5.1 Validación de Sesiones**



<?php

if (!isset($\_SESSION['rol']) || $\_SESSION['rol'] !== 'camarero') {

    header('Location: ../index.php');

    exit;

}

**5.2 Sanitización de Datos**



<?php

$mesa\_id = filter\_input(INPUT\_GET, 'mesa\_id', FILTER\_VALIDATE\_INT);

if (!$mesa\_id) {

    header('Location: gestionar\_mesas.php');

    exit;

}

**6. Funcionalidades de Gestión de Pedidos**

**6.1 Cálculo de Totales**



<?php

$total = 0;

while ($detalle = mysqli\_fetch\_assoc($detalle\_pedidos\_result)) {

    $subtotal = $detalle['cantidad'] \* $detalle['precio'];

    $total += $subtotal;

}

**6.2 Sistema de Confirmación**



function confirmarEliminacion() {

    return confirm('¿Está seguro de eliminar este producto?');

}

**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.