SMARTORDER

Gestión de Pedidos en Restaurantes

IVÁN MARTÍNEZ OLMEDO

1. INTRODUCCIÓN

1.1.Presentación del Proyecto

Smartorder es una aplicación web para la gestión de pedidos en bares/restaurantes. Permite a los clientes ver el menú, seleccionar productos y hacer pedido asociados a mesas, los administradores además de eso también podrán gestionar productos y mesas, controlar el estado de estas últimas y supervisar los pagos.

1.2 OBJETIVOS

Podríamos poner como objetivo general el desarrollar una aplicación web que gestione mesas, pedidos y pagos de forma clara y eficiente. Alguno de los objetivos específicos son:

- Crear una interfaz simple y responsive.
- Diferenciar roles de usuario (cliente / admin).
- Permitir pagar productos por separado.
- Automatizar la liberación de mesas tras el pago.
- Simular entorno real de un bar/restaurante.

1.3 JUSTIFICACIÓN

Las apps actuales como Glovo o Uber Eats están pensadas para delivery, no para el servicio en mesa.

SmartOrder responde a una necesidad real de muchos locales pequeños.

2. Análisis de requerimientos

2.1. Identificación de necesidades y requerimiento:

- Sistema ágil para tomar pedidos desde la aplicación.
- Asociación de pedidos a mesas.
- Control por roles.
- Validaciones, seguridad y simplicidad.

Funcionalidad	SmartOrder	Glovo	Uber Eats
Pedido desde mesa	Sí	No	No
Gestión interna de productos/mesas	Sí	No	No
Pago por separado	Sí	No	No
Interfaz personalizada por negocio	Sí	No	No
Reparto a domicilio	No	Sí	Sí

2.2 PÚBLICO Y MERCADO

Público:

- Clientes del restaurante que hacen el pedio y pagan
- Personal administrativo que se encarga de gestionar mesas, productos y los pagos.

Competencia:

- Apps como Just Eat no permiten pagos internos en mesa.
- SmartOrder cubre ese hueco, permitiendo digitalizar la experiencia dentro del local,

3. Diseño y planificación

3.1 Definición de la arquitectura del proyecto:

- Laravel como backend (PHP + MVC).
- Vistas con blade.
- Estilos con Tailwind CSS y Bootstrap.
- Base de datos relacional con MySQL.
- Sesiones y middleware para controlar accesos.

3. Diseño y planificación

3.2. Diseño de la interfaz de usuario:

- Dashboard personalizado según el rol.
- Menú dividido por categorías con botones grandes.
- Vista de selección de mesa antes de comenzar.
- Vista de pedido por cuenta atras.
- Factura final clara.

3. Diseño y planificación

3.3. Planificación de tareas y recursos:

- Planificación de fases:
  - Fase 1: diseño de base de datos.
  - Fase 2: desarrollo de vistas y controladores.
  - Fase 3: implementación de pagos.
  - Fase 4: validaciones y pruebas.
  - Fase 5: mejora de estilo y mapa.

4. Implementación y pruebas

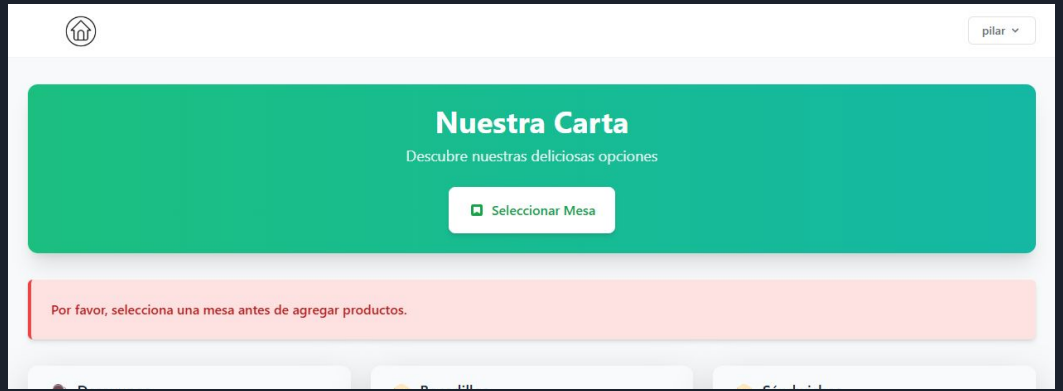
4.1. Desarrollo de funcionalidades:

- Autenticación con roles (cliente y admin).
- Gestión de productos y mesas (CRUD).
- Hacer pedidos (con cuenta atrás).
- Pagar por separado o todo.
- Liberar mesas automáticamente.
- Mostrar mapa con ubicación.

4. Implementación y pruebas

4.2. Pruebas unitarias y de integración:

- Errores si no hay mesa seleccionada.
- Validación de acceso por rol.
- Mensaje si se intenta pagar sin marcar productos.
- Mesa ocupada por otra sesión .
- Confirmación al eliminar productos.



4. Implementación y pruebas

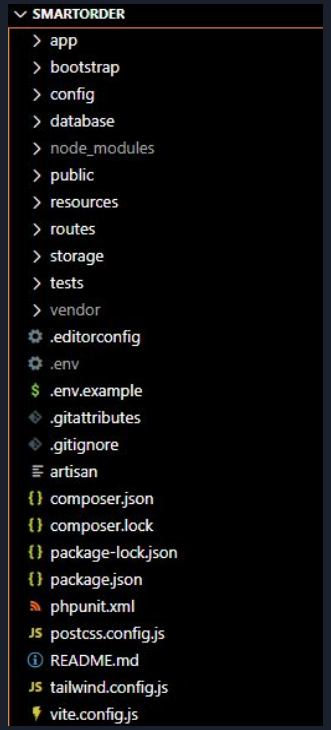
4.3. Corrección de errores y optimización:

- Solucionado bug ID fijo de mesa.
- Mejora en control de sesiones y limpieza tras pagos.
- Eliminadas recargas innecesarias.
- Carga de estilos via CDN y recursos mínimos para rapidez.

5. DOCUMENTACIÓN

5.1. Documentación técnica:

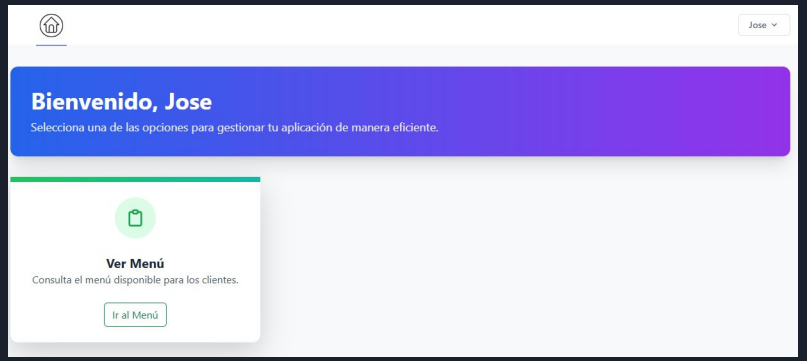
- Estructura de carpetas Laravel:
  - app/Http/Controller
  - resources/views
  - routes/web.php
- Roles definidos en controladores.
- Control de sesiones para lógica de pedidos.

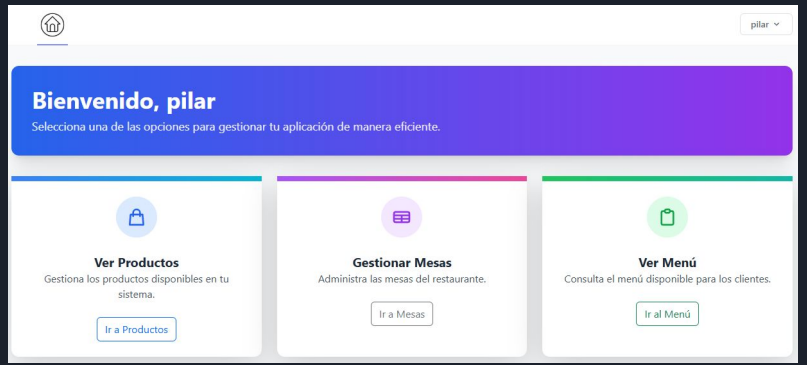


5. DOCUMENTACIÓN

5.2. Documentación de usuario:

- Cliente:
  - Ver menú.
  - Seleccionar mesa.
  - Hacer pedido.
  - Pagar.
  - Ver factura.
- Administrador:
  - Ver dashboard.
  - Gestionar productos y mesas.
  - Liberar mesas.





5. DOCUMENTACIÓN

5.3. Manual de instalación:

- Requisitos: XAMPP, PHP, Composer, Node.js, Laravel, navegador.
- Comandos:
  - composer install
  - Configurar .env
  - npm install
  - npm run dev
  - php artisan serve

6. MANTENIMIENTO Y EVOLUCIÓN

6.1. Plan de mantenimiento y soporte:

- Backup diario de base de datos.
- Revisión de errores reportados.
- Actualización de Laravel y dependencias si hay vulnerabilidades.

6. MANTENIMIENTO Y EVOLUCIÓN

6.2. Identificación de mejoras:

- Escaneo QR.
- Nuevo role: camarero/cocina.
- Control de stock por producto.
- Notificaciones en tiempo real con WebSockets.

6. MANTENIMIENTO Y EVOLUCIÓN

6.3. Actualizaciones futuras:

- App móvil nativa.
- Integración con Stripe/Bizum.
- Panel de estadísticas.
- Traducción a distintos idiomas.
- Gestión avanzada de mesas (reservas, comensales).

7. CONCLUSIONES

7.1. Evaluación del proyecto

- Se cumplió el objetivo de crear una app funcional.
- Superó expectativas al incluir validaciones, roles y diseño cuidado.
- Buen rendimiento y estabilidad.
- Experiencia de usuario cuidada.
- Implementación escalable

7. CONCLUSIONES

7.2. Cumplimiento de objetivos

- Todos los objetivos técnicos y funcionales logrados.
- Interfaz sencilla y responsive.
- Roles bien diferenciados.
- Pagos individuales o totales.
- Mejoras planificadas no necesarias para la funcionalidad básica.

7. CONCLUSIONES

7.3. Lecciones aprendidas y recomendaciones:

- Importancia de diseñar primero la base de datos.
- Mantener estructura limpia desde el inicio.
- Usar sesiones correctamente.
- Documentar bien y probar como si fueras el usuario.

8. BIBLIOGRAFÍA Y REFERENCIAS

Laravel Docs

Leaflet.js

Bootstrap/ TilwindCSS

StackOverflow

Laracasts

GitHub

ChatGTP

Video informativo de Píldoras Informáticas

VIDEO