ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal

Memoria de Proyecto Fin de CFGS

Desarrollo de Aplicaciones Web.

07/06/2024

IES Bernaldo de Quirós

Denís Camblor Caberos

ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal

Índice

ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal

1. Introducción

1.1 Presentación y Objetivos

BookMyMeal es una aplicación web diseñada para digitalizar y automatizar tareas en la gestión de reservas y pedidos en negocios hosteleros. Su objetivo principal es mejorar la eficiencia de estos negocios, optimizar la atención al cliente y aumentar las ventas mediante una plataforma moderna, accesible y funcional.

Objetivo General: Automatizar las reservas y pedidos para mejorar la gestión de restaurantes y bares, proporcionando una aplicación web de fácil acceso que ofrezca una experiencia fácil a usuarios y administradores.

Objetivos Específicos:
1. Automatizar reservas y pedidos.
2. Mejorar la visibilidad digital del negocio.
3. Gestionar eficazmente la capacidad del local.
4. Aumentar el volumen de ventas mediante pedidos online.

1.2 Contexto

En la actualidad, muchos negocios de hostelería aún gestionan sus reservas y pedidos de forma tradicional, lo que genera errores, pérdida de tiempo y una experiencia de cliente poco eficiente. La digitalización se ha convertido en una necesidad para mantenerse competitivo. BookMyMeal surge como una solución para aquellos locales que no cuentan con una plataforma digital o desean mejorar la que ya tienen.

1.3 Planteamiento del Problema (la idea)

La falta de digitalización en la gestión de reservas y pedidos limita la capacidad de respuesta de los negocios ante una alta demanda, reduce la eficiencia operativa y afecta negativamente la imagen del local. BookMyMeal propone una solución integral que automatiza estos procesos, mejora la experiencia del cliente y permite a los negocios aumentar sus ingresos sin necesidad de ampliar su infraestructura física.

1.4 Análisis de Costes

Concepto	Coste (€)
Salario del desarrollador	7.500
Ordenador para desarrollo	250
Dominio y certificado SSL (anual)	50
Alquiler, luz e internet (3 meses)	1.320
Total:	9.120

1.5 Plan de Financiación

El desarrollo y lanzamiento de la aplicación BookMyMeal requiere una inversión inicial estimada de 9.120 €. Para poder financiar el proyecto, se ponen tres fases.

Fase 1: Financiación Propia (30%)

Se destinará un 30% del presupuesto inicial (2.736 €) a través de recursos propios tanto de familiares y amigos. Esta fase permitira empezar el proyecto rapidamente.

Fase 2: Subvenciones y Ayudas Públicas (40%)

Se buscará obtener un 40% del presupuesto (3.648 €) mediante subvenciones como:
- Kit Digital.
- Ayudas del Gobierno del Principado de Asturias.
- Fondos europeos como Next Generation UE.

Fase 3: Crowdfunding (30%)

Se lanzará una campaña en plataformas tipicas con el objetivo de recaudar un 30% del presupuesto (2.736 €). A cambio, se ofrecerán recompensas como:
- Acceso anticipado a la app.
- Publicidad gratuita para los primeros.

1.6 Plan de Recursos Humanos

1. Objetivo del Plan de Recursos Humanos

Teniendo en cuenta que será desarrollado y gestionado por una única persona: análisis, diseño, desarrollo, pruebas, despliegue y mantenimiento, serán desarrolladas por la misma persona.

2. Estructura del Equipo de Trabajo

Denís Camblor Cabero:

- **Jefe de Proyecto**
- **Analista de Requisitos**
- **Diseñador UI/UX**
- **Desarrollador Full-Stack**
- **Administrador de Base de Datos**
- **Tester / QA**

3. Distribución de Fases de desarrollo del proyecto

1. Fase de análisis:

En esta fase se analizar todos los requisitos tanto técnicos como funcionales de la aplicación, esto incluye desarrollo de modelos entidad relación, tecnologías que se van a usar, estructuras de datos y arquitectura de la aplicación.

2. Diseño de la interfaz

Diseño de la landing page, del admin dashboard, carta, etc...

3. Desarrollo

Fase del desarrollo de software tanto backend(Node.js) y frontend(Angular), creación del script de la base de datos.

4. Pruebas (QA)

Realizar pruebas funcionales, validar reservas y pedidos, corregir errores.

5. Despliegue

En esta fase es el momento en el que la aplicación será visible al mundo, es decir, será publicada en la web.

1.7 Plan de Prevención de Riesgos

Un plan de riesgos laborales en este caso debe prevenir posibles riesgos asociados al trabajo prolongado frente a un ordenador y asegurar un entorno de trabajo seguro, saludable y eficiente durante el desarrollo del proyecto.

Riesgos Potenciales

1. **Riesgos Visuales**
   a. Fatiga ocular por las largas jornadas junto a pantallas.
   b. Enfocar durante mucho tiempo a una distancia fija sin descanso.
2. **Riesgos Ergonómicos**
   a. Mala postura durante el trabajo.
   b. Iluminación insuficiente o mal orientada.
3. **Riesgos Eléctricos**
   a. Uso inadecuado de equipos informáticos o electrónicos.
   b. Sobreutilización de regletas o enchufes múltiples.
4. **Riesgos Físicos**
   a. Dolor en espalda, cuello o muñecas por malas posturas.
   b. Sendentarismo puede causar lesiones musculares por falta de movimiento.
5. **Riesgos Psicosociales**
   a. Estrés debido a sobrecarga de tareas o mala gestión del tiempo.
   b. Ansiedad ante plazos estrictos o falta de organización.

Medidas Preventivas

ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal

1. Ergonomía

- Utilizar silla ajustable con soporte lumbar.
- Mantener la pantalla a la altura de los ojos y a unos 50-60 cm de distancia.
- Usar teclado y ratón colocados correctamente para evitar tensión muscular.

2. Pausas Activas

- Realizar pausas de 5-10 minutos cada hora para estirarse y descansar la vista.
- Cambiar de postura regularmente para evitar rigidez muscular.

3. Iluminación

- Trabajar en un lugar bien iluminado, preferiblemente con luz natural indirecta.

4. Riesgos Psicosociales

- Planificar las tareas del proyecto con un cronograma realista.
- Evitar jornadas de trabajo excesivamente largas.

5. Seguridad Eléctrica

- Evitar el uso de regletas sobrecargadas.
- Asegurar el correcto funcionamiento de cables, cargadores y enchufes.

2. ANÁLISIS. ESPECIFICACIÓN DE REQUISITOS.

2.1 Introducción

El análisis se enfoca en entender el problema que el proyecto intenta resolver y en establecer una base sólida para definir los requisitos funcionales y técnicos. Sobre todo, se trata de identificar y documentar las funcionalidades requeridas para satisfacer las necesidades de la aplicación.

Como resultado de este análisis, vamos a obtener los Requisitos (funcionales y no funcionales), Diagramas de Clases, Diagramas de Casos de Uso, Diagrama relacional (E/R o Lógico) de la BBDD, etc

2.2 Descripción general.

Una aplicación web que permite la automatización de las reservas y pedidos para mejorar la gestión de restaurantes y bares, proporcionando una aplicación de fácil acceso que ofrezca una experiencia fluida a usuarios y administradores.

2.3 Requisitos Específicos.

2.3.1 Requerimientos Funcionales.

1. Registro y Autenticación de Usuarios: Permite a los usuarios crear cuentas (clientes o superusuarios) y autenticarse de manera fácil y segura.

2. Visibilidad del Negocio: La aplicación en su conjunto es una parte informativa sobre el negocio, incluyendo menú, ubicación y datos de contacto, todo esto mejora la visibilidad e imagen del negocio.

3. Sistema de Reservas Automatizado: Calendario Interactivo: Un sistema de reservas que muestra la disponibilidad de mesas en tiempo real y permite reservar en horarios específicos. (Este sistema a la vez con lleva un control de capacidad del local).

4. Gestión de Pedidos: Pedidos para Llevar y a Domicilio: Los clientes pueden realizar pedidos para recoger o a domicilio. Mediante pagos online en este caso Stripe.

5. Gestión Administrativa: Panel de Administración: Permite a los trabajadores gestionar pedidos y reservas.

6.Diferentes acciones dependiendo del tipo de usuarios:

1. Administrador: Propietarios o gerentes del local con acceso total a la plataforma pueden hacer operaciones CRUD con usuarios, pedidos, reservas y productos.

2. Trabajadores/empleados: Personal del local que puede visualizar las reservas y pedidos, actualizar el estado de cada pedido (preparado, entregado, etc), con permisos limitados en comparación con el administrador.

3. Clientes (Usuario Registrado): Usuarios que se registran para realizar reservas y pedidos a través de su correo electrónico. Estos usuarios pueden consultar la disponibilidad de mesas, hacer reservas y realizar pedidos para llevar o a domicilio, proporcionando los datos necesarios para el pago y la entrega.

4. Usuario sin Registro: Para aquellos clientes que desean realizar pedidos sin necesidad de crear una cuenta. Solo se recopilarán los datos mínimos necesarios para completar el pedido y el pago. Esto permite realizar un pedido sin riesgo para el propietario, ya que el pedido se paga en el momento de la solicitud. (Estos usuarios solo tienen acceso a los pedidos y a visualizar el contenido de la página (menú, fotos...))

2.3.2 Requerimientos de Interfaces Externas.

2.3.2.1 Interfaces de los Usuario.

BokkMeal ofrece una interfaz moderna y responsiva, accesible desde navegadores actuales en ordenadores, tablets y dispositivos móviles. La plataforma está pensada para diferentes perfiles de usuario: Administrador, Cliente Registrado y Usuario sin Registro, cada uno con funcionalidades y vistas adaptadas a sus necesidades.

La navegación es sencilla e intuitiva, con secciones dedicadas al registro e inicio de sesión, búsqueda y exploración de platos, gestión de pedidos y reservas, perfil del usuario y un completo panel de administración para la gestión del negocio.

Se utilizan componentes interactivos como menús desplegables, formularios dinámicos, mensajes de confirmación y alertas que contribuyen a una experiencia de usuario fluida y agradable.

2.3.2.2 Interfaces Hardware.

Desarrollo: En cuando al desarrollo se ha utilizado un HP Elitebook 845 G11:

- AMD Ryzen 5 Pro
- 16 Gb ram
- 512 Gb SSD en Dropbox
- Windows 11 Pro.

Para uso de los usuarios: Lo podrán utilizar desde cualquier dispositivo que tenga un navegador y acceso a internet.

2.3.2.3 Interfaces Software

El stack que se ha usado para el desarrollo de toda la aplicacion BookMyMeal ha sido:

1. **Frontend:** Angular, Tailwind CSS y daisyui.
2. **Backend:** Nodejs – Express, JWT para tokens de autenticación, Stripe como pasarela de pagos y nodemailer como servicio de correo para enviar correos.
3. **Base de datos:** MySQL.
4. **Para desarrollo de pruebas:** Stripe CLI y postman.

2.3.2.4 Interfaces de Comunicaciones.

Para la comunicación entre backend y frontend se ha utilizado una arquitectura API Rest por petición HTTP.

2.3.3 Requerimientos de Rendimiento.

En caunto a los requerimientos de rendimiento son cuatro principios.

1. Tiempo de carga

- **Inicio y menú principal**: debe cargarse en menos de **2 segundos**.
- El usuario debe poder empezar a interactuar con la interfaz en menos de **3 segundos** tras acceder.

2. Interactividad y fluidez

- Las acciones clave (añadir al carrito, hacer una reserva, filtrar platos) deben responder en menos de **100-200 ms**
- El **panel de administración** debe mostrar listados y actualizaciones (platos, pedidos) en menos de **500 ms**.

3. Rendimiento del backend

- Las API REST deben responder en menos de **300 ms** bajo carga normal.
- Consultas como:
    - "Buscar platos por categoría"
    - "Obtener historial de pedidos"
    - "Reservas por fechas

4. Capacidad concurrente y escalabilidad

- Capacidad mínima para manejar:
    - **50-100 usuarios activos simultáneos** en restaurantes pequeños.
    - Escalabilidad progresiva para más de **500 usuarios** en eventos o promociones.

2.3.4 Obligaciones del Diseño.

2.3.4.1 Estándares Cumplidos.

El diseño de la aplicación BokkMeal sigue los siguientes estándares para garantizar accesibilidad, usabilidad y compatibilidad:

- **Responsive Web Design (RWD):** la interfaz se adapta a diferentes tamaños de pantalla (móviles, tablets, escritorio).
- **Estándares de HTML5 y CSS3:** código semántico y validado, compatible con todos los navegadores modernos.
- **Compatibilidad multi-navegador:** garantizado soporte mínimo en Chrome, Firefox, Safari y Microsoft Edge (últimas 2 versiones).
-

2.3.4.2 Limitaciones Hardware.

La aplicación BokkMeal está diseñada para funcionar correctamente con la mayoría de los dispositivos, estos son los requisotos mínimos y recomendados para una experiencia fluida:

En dispositivos del usuario (cliente):

- **Requisitos mínimos:**
  - Navegador moderno (Chrome, Firefox, Safari, Edge) actualizado.
  - Resolución mínima: **360x640 px**.
  - CPU equivalente a **1 GHz**, 2 GB de RAM.
  - Conexión estable a Internet (mínimo 3G o superior).

- **Requisitos recomendados:**
  - Dispositivo con **4 GB de RAM o más**, procesador de 2 núcleos.
  - Resolución estándar **HD (1280x720 px)** o superior.
  - Conexión de banda ancha o 4G/5G para mayor velocidad.

2.3.5 Atributos.

2.3.5.1 Seguridad.

La aplicación BooMyMeal ha sido desarrollada teniendo en cuenta medidas de seguridad robustas para proteger tanto los datos personales de los usuarios como las operaciones realizadas dentro del sistema (como pedidos, reservas y administración de platos).

Estándares de Seguridad Implementados:

- ***JWT (JSON Web Tokens):*** Se utiliza como sistema de autenticación y autorización, asegurando que solo usuarios válidos puedan acceder a recursos protegidos. Cada vez que un usuario inicia sesión correctamente, se genera un token que debe acompañar cada petición al backend para validar su identidad.
- ***Autenticación en dos pasos (2FA):*** Además del usuario y la contraseña, se exige tener acceso al correo electrónico asociado para validar el inicio de sesión. El sistema de doble factor está desarrollado de forma interna, añadiendo una capa extra de seguridad sin depender de servicios externos.
- ***Hash de contraseñas:*** Las contraseñas se almacenan de forma cifrada utilizando un algoritmo de hash en este caso bcrypt, evitando el almacenamiento de credenciales en texto plano.

Protección de Rutas:

- ***Lado cliente (Angular):*** Se emplean Guards personalizados para proteger rutas según el rol del usuario (Administrador, Cliente, Invitado). El acceso a secciones como el panel de administración o el historial de pedidos requiere autenticación y permisos adecuados.
- ***Lado servidor (Node.js):*** Todas las rutas sensibles están protegidas mediante validación del token JWT. Si el token no es válido o ha expirado, la solicitud se deniega inmediatamente.

2.3.5.2 Facilidades de Mantenimiento.

Para facilitar elo matenimiento se han utilizado las siguientes buenas prácticas y técnicas.

- **Gestión de dependencias:** Se utiliza npm para el frontend (Angular) y el backend (Node.js), lo que permite mantener las librerías actualizadas.
- **Separación clara entre cliente y servidor: La aplicación está dividida en dos partes:**
  El frontend en Angular y el backend en Node.js con Express, lo que permite trabajar en ambos de forma paralela y facilita la implementación de futuras mejoras o migraciones tecnológicas.
- **Modularidad del código:** En el frontend, componentes y servicios de Angular favorece la reutilización de código y una estructura limpia. En el backend, el código está organizado por rutas, controladores y servicios, siguiendo una arquitectura MVC (Modelo-Vista-Controlador) que mejora la legibilidad y mantenibilidad.
- **Preparado para ampliaciones:** La estructura modular y el diseño basado en servicios permiten añadir fácilmente nuevas funcionalidades con facilidad, integrando con APIs externas, o nuevas vistas en el panel de administración.

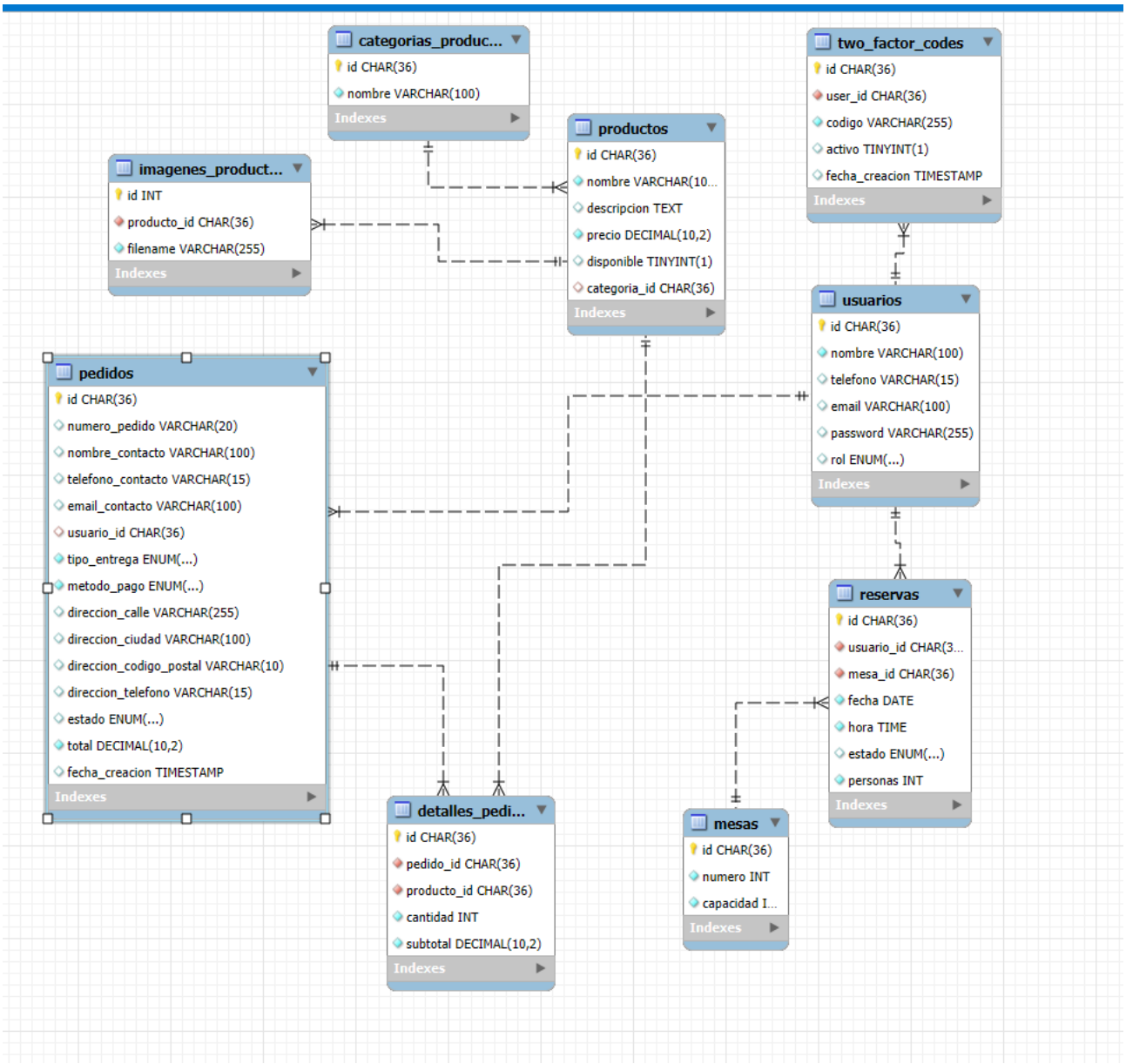
2.3.5.3 Portabilidad.

Gracias al stack tecnológico empleado Angular para el frontend, Node.js para el backend y MySQL como base de datos la aplicación es altamente portable y compatible con una amplia variedad de servicios de despliegue. Estas tecnologías son ampliamente utilizadas en el sector del desarrollo web, lo que garantiza su soporte en la mayoría de las plataformas, tanto en entornos locales como en la nube.

Por ello, migrar la aplicación a otro entorno de despliegue solo requiere que el sistema sea compatible con estas tecnologías, algo muy común en la mayoría de los proveedores actuales como Vercel, AWS, Firebase, Railway, entre otros.

ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal

2.4 Diagrama de Clases.



1. usuarios

Representa a los usuarios del sistema (clientes y administradores).

Atributos: id, nombre, telefono, email, password, rol (ENUM: cliente, admin, etc.).

Relaciones:

- Un usuario puede hacer muchos pedidos.
- Un usuario puede hacer muchas reservas.
- Se vincula con la entidad two_factor_codes para (2FA).

2. two_factor_codes

Sistema de doble autenticación (2FA) vinculado a cada usuario.

Relación: Cada código pertenece a un usuario (user_id).

3. productos

Representa los productos del menú (platos, bebidas, etc.).

Relaciones:

- Pertenece a una categoría de producto.
- Puede tener varias imágenes asociadas.
- Estan en los detalles_pedido.

4. categorias_productos

Clasifica los productos (ej. Bebidas, Entrantes, Postres...).

- **Relación:** Una categoría puede tener muchos productos.

5. imagenes_productos

Permite asociar múltiples imágenes a un producto.

- **Relación:** Cada imagen pertenece a un producto.

6. pedidos

Representa un pedido realizado por un usuario.

Relaciones:

- Un pedido pertenece a un usuario.
- Un pedido tiene varios detalles_pedido (líneas de productos comprados).

7. detalles_pedido

Es una tabla intermedia que detalla los productos de un pedido.

Relaciones:

- Cada registro vincula un producto con un pedido.
- Permite múltiples productos por pedido.

8. reservas

Representa una reserva de mesa hecha por un usuario.

Relaciones:

- Cada reserva pertenece a un usuario.
- Cada reserva está asociada a una mesa.

9. mesas

Define las mesas del restaurante.

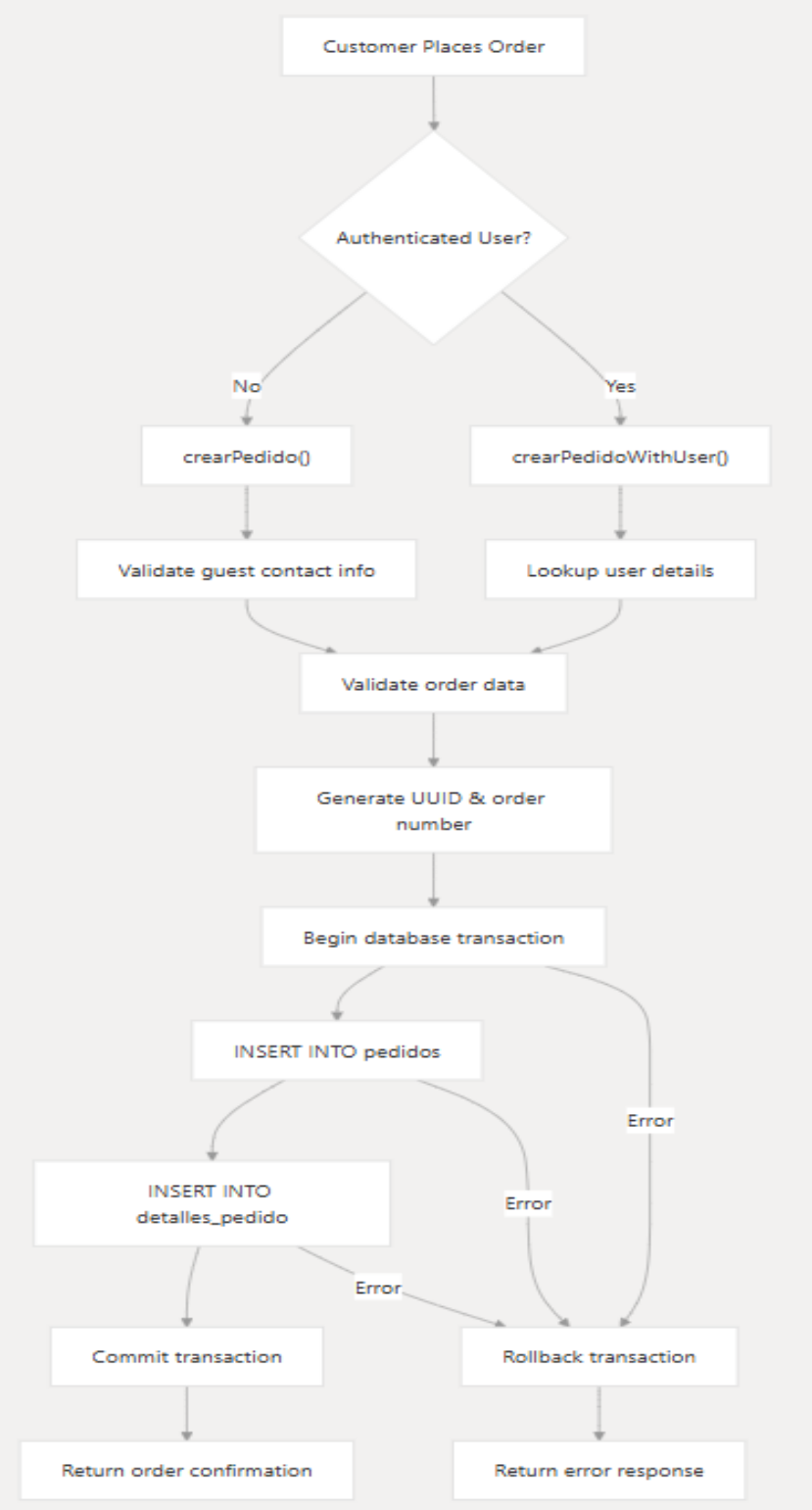
Relación: Una mesa puede estar vinculada a varias reservas.

2.5 Diagrama de Casos de Uso.

Debido a que son demasiados diagramas solo he puesto los dos más complejos que son el de orders y el de reservations. El resto de las operaciones sigue una estructura clara de MVC.

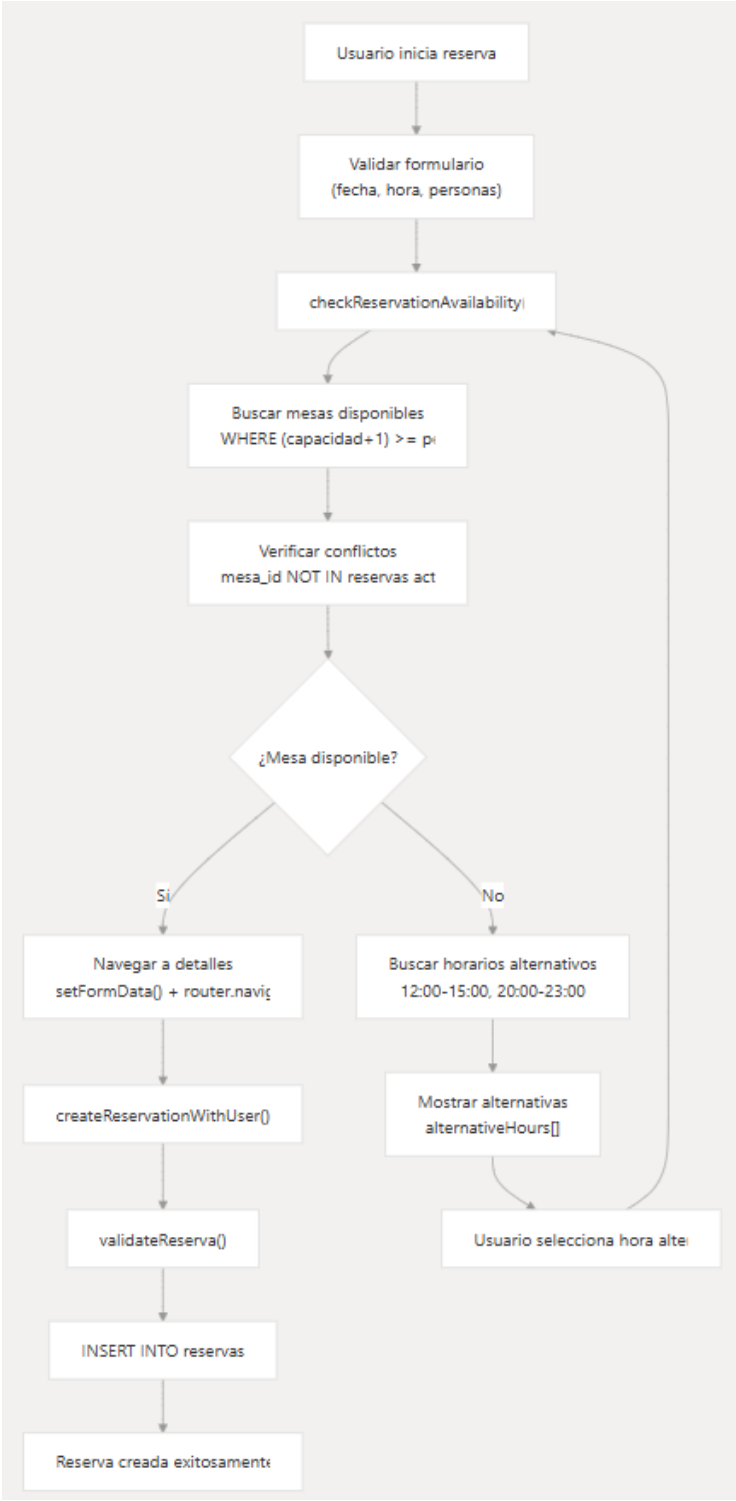
- **Orders:** En este diagrama se explica cual seria el flujo de un pedido sin tener en cuenta la logica de Stripe ya que no la sabemos

ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal

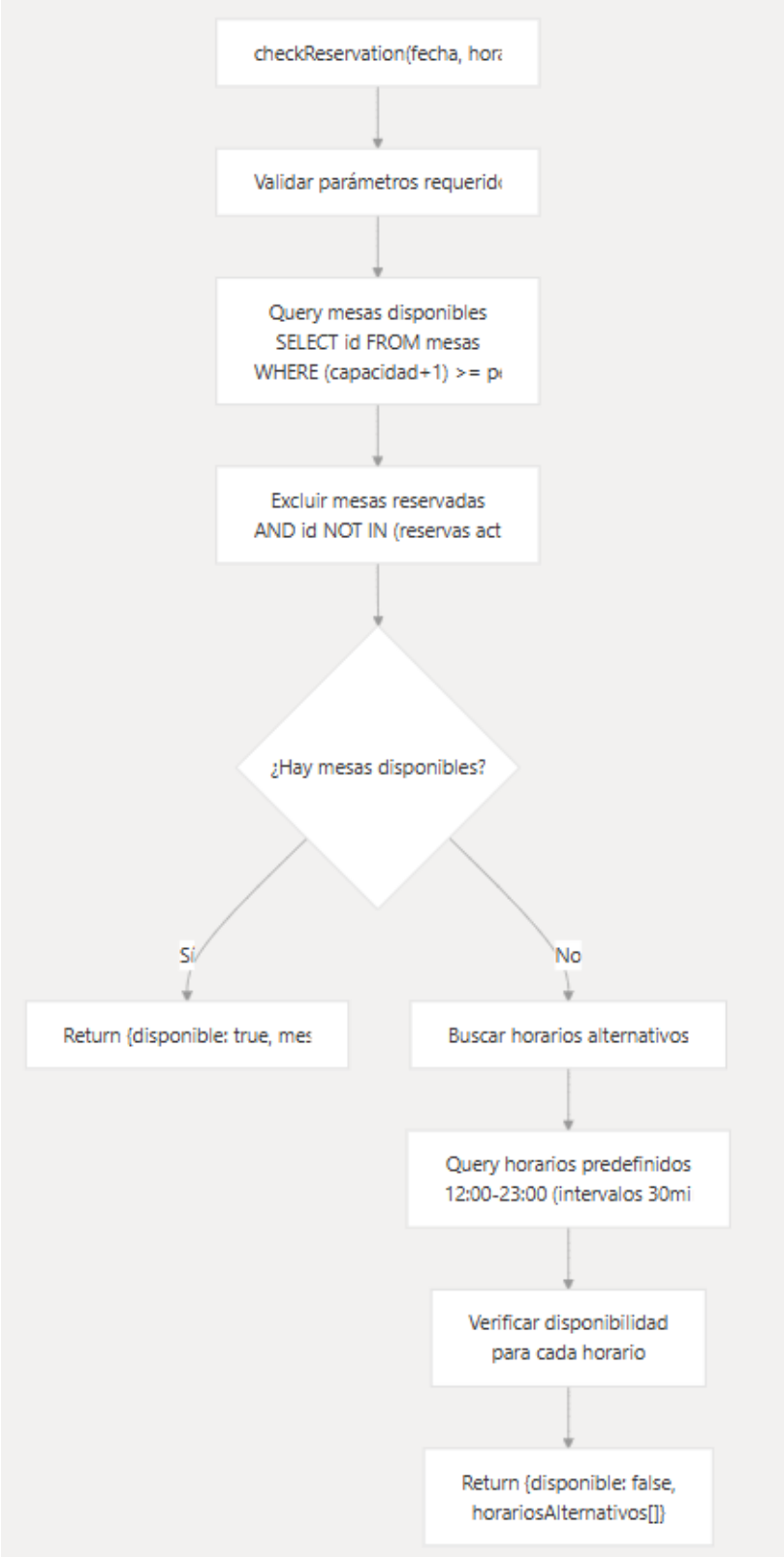


Reservations: En este diagrama se explica el flujo de una reserva.

ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal



ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal



3. DISEÑO

3.1 Introducción

En esta fase de Diseño vamos a planificar y estructurar cómo estarán organizadas las diferentes capas (Presentación, Negocio, Persistencia) y cómo interactuarán entre sí, sin entrar aún en detalles sobre las tecnologías concretas. Es decir, vamos a documentar como diseñaremos la implementación de la aplicación web de manera general, pero no al nivel técnico detallado.

3.2 Capa de Presentación.

La interfaz de usuario de **BookMyMeal** es el punto de contacto entre los usuarios y el sistema. Se ha diseñado pensando en claridad, accesibilidad y adaptabilidad, permitiendo su uso en una amplia variedad de dispositivos, incluyendo ordenadores, tablets y móviles.

Aquí se muestran elementos necesarios para que los usuarios puedan interactuar de forma sencilla y eficaz con la aplicación.

Funcionalidades según el tipo de usuario

- **Cliente sin cuenta**:
  Puede consultar la carta de productos, ver información de cada plato y acceder a hacer pedidos. También se le ofrece la posibilidad de iniciar sesión o crear una cuenta.
- **Cliente registrado**:
  Tiene acceso a funcionalidades adicionales como realizar pedidos online, realizar reservas, consultar su historial etc...
- **Empleado del restaurante**:
  Cuenta con una interfaz operativa simplificada para acceder al listado de pedidos y reservas activas, actualizar su estado, etc... Vista está optimizada para su uso en dispositivos de trabajo dentro del local.
- **Administrador del sistema**:
  Dispone de un panel de control completo desde el que puede gestionar productos, categorías, usuarios, empleados, reservas y pedidos

3.3. Capa de Negocio o Lógica de la Aplicación

La **capa de negocio** representa el corazón de **BookMyMeal**. Es la encargada de aplicar las reglas del sistema y coordinar todos los procesos que definen el comportamiento lógico de la aplicación.

La lógica de negocio en **BookMyMeal** sigue el patrón **Modelo–Vista–Controlador (MVC)**, ampliamente utilizado en aplicaciones desarrolladas con **Node.js** y **Express.js**:

- **Modelo (Model):**
  Representa las entidades del sistema (usuarios, pedidos, reservas, productos, etc.). Cada modelo define la estructura y relaciones de los datos almacenados en la base de datos (MySQL).
- **Controlador (Controller):**
  Gestiona las peticiones que llegan desde el cliente. Se encarga de validar datos, coordinar la lógica de negocio y construir las respuestas adecuadas. Los controladores sirven de puente entre las vistas (o APIs frontend) y los modelos.
- **Vista (View):**
  En el caso de BookMyMeal, la vista está implementada en Angular, que consume las APIs expuestas por el backend. Aunque el patrón MVC tradicional incluye plantillas HTML generadas desde el servidor, en esta arquitectura desacoplada, la "vista" se refiere al **cliente Angular** que interactúa con los controladores mediante solicitudes HTTP.

Esta capa gestiona:

- Validar acciones según el rol del usuario (cliente, empleado o administrador).
- Procesar la lógica de pedidos, desde su creación hasta la actualización de su estado.
- Controlar las reservas: verificar disponibilidad, asignar mesas y gestionar estados (pendiente, confirmada, cancelada, etc.).
- Administrar el catálogo de productos (platos): creación, edición, visibilidad y disponibilidad.
- Gestionar el proceso de autenticación con JWT y verificación en dos pasos (2FA) vía correo electrónico.
- Coordinar el envío de correos automáticos para confirmaciones de reserva/pedido, notificaciones y recuperación de cuentas.

3.4. Capa de Persistencia o Datos

La capa de persistencia en **BookMyMeal** es la responsable de interactuar directamente con la base de datos. Su misión es almacenar, recuperar, actualizar y eliminar la información necesaria para el funcionamiento del sistema, de forma segura y eficiente.

Entre las operaciones principales que realiza esta capa se encuentran:

- Guardar y recuperar datos relacionados con usuarios, productos, pedidos, reservas, mesas, códigos 2FA y otros elementos.
- Consultar y filtrar información solicitada por la lógica de negocio, como disponibilidad de mesas, historial de pedidos o productos por categoría.
- Garantizar la integridad de los datos y gestionar correctamente las relaciones entre entidades (por ejemplo, detalles de pedido asociados a un pedido).
- Utilizar repositorios para exponer interfaces limpias y evitar que otras capas deban interactuar directamente con SQL o estructuras complejas.

Esta capa proporciona una abstracción del acceso a los datos, de modo que el resto de la aplicación no necesita preocuparse por cómo se almacenan o recuperan.

4. IMPLEMENTACIÓN

4.1 Tecnologías utilizadas en el desarrollo del proyecto

Las tecnologías utilizadas son:

- **Backend:** Nodejs-Express y nodemailer.
- **Frontend:** Angular, Tailwind CSS y daisyui.
- **Base de datos:** MySQL.
- **Pasarela de pagos:** Stripe
- **Seguridad:** JWT, bcript para las contraseñas y cors origin de express para que solo se puedan hacer peticiones desde el frontend.

4.2 Descripción del Proyecto

4.2.1 Capa de Presentación.

La capa de presentación en BookMyMeal ha sido desarrollada en Angular con HTML y Talwind CSS para garantizar un diseño responsive.

Además, se han usado algunos componentes de Daisyui para agilizar el desarrollo.

Configuración Principal de Rutas

La configuración de rutas se define principalmente en el archivo **`app.routes.ts`**:

Rutas padre

- **`/`** → **`landing-page`** `lleva a las rutas de` **`landing-page.routes.ts`**
- **`/admin-dashboard`** → lleva al panel del admin (protegido por `adminGuard`).
- **`/login`** → Página de inicio de sesión.
- **`/register`** → Página de registro de usuarios.
- **`/**`** → Si se intenta ir a una ruta que no existe lleva a la landing page

Landing Page

El módulo principal para los usuarios utiliza `LandingPageLayoutComponent` como layout base. Su configuración de rutas se encuentra en `landing-page.routes.ts`:

- **`/`** → `HomeComponent (página principal).`
- **`/carta`** → `CartaComponent (visualización del menú de productos).`
- **`/confirmar-pedido`** → `PedidosComponent (proceso de finalización de pedido).`
- **`/reservas`** `(protegido por authGuard)` →
    - **`/reservas`** → `ReservasComponent`
    - **`/reservas/detalles`** → `DetallesReservaComponent`
    - **`/reservas/gracias`**`/` → `GraciasReservaComponent(comfirma la reserva)`

Layout Principal

El layout base está compuesto por:

- **Header fijo**: `landing-page/components/front-navbar` con posición `fixed` y `z-index` alto para estar siempre visible.

- **Contenido principal**: Sección central con margen superior que cambiara segun la ruta es decir si la ruta es **/carta** saldra el header, la carta y el footer el contenino principal es variable.

- **Footer**: `landing-page/components/footer-landing` ubicado al final de la página.

Admin Dashboard

El panel de administración tiene su propia estructura de rutas definidas en **`admin-dashboard.routes.ts`**:

`/admin-dashboard` → Lleva al panel de administración.

- `/admin-dashboard` → **DashboardHomeComponent** (vista principal del panel).
- `/admin-dashboard/reservas` →
  - `''` → **AdminReservationsComponent** (lista de reservas).
  - `crear` → **CreateReservationComponent** (formulario para añadir una nueva reserva).
  - `actualizar/:id` → **UpdateReservationComponent** (editar una reserva existente).
  - `detalles/:id` → **ReservationDetailsComponent** (ver detalles de una reserva).
- `/admin-dashboard/pedidos` →
  - `''` → **AdminOrdersComponent** (lista de pedidos).
  - `crear` → **CreateOrderComponent** (crear un nuevo pedido).
  - `actualizar/:id` → **UpdateOrderComponent** (modificar un pedido).
  - `detalles/:id` → **OrderDetailsComponent** (ver detalles del pedido).
- `/admin-dashboard/productos` →
  - `''` → **AdminProductsComponent** (gestión de productos).
  - `crear` → **CreateProductComponent** (agregar un producto nuevo).
  - `actualizar/:id` → **UpdateProductComponent** (editar un producto).
  - `detalles/:id` → **ProductDetailsComponent** (ver información de un producto).

Sistema de Autenticación y Guards

- `authGuard`: Protege rutas que requieren autenticación.
- `adminGuard`: Protege rutas exclusivas para administradores.

Configuración General (app.config.ts)

- Detección de cambios optimizada.
- Configuración del router.
- Cliente HTTP configurado con interceptor de autenticación para envío de tokens JWT.

4.2.2 Capa de Negocio o Lógica de la Aplicación.

El backend de BookMyMeal está desarrollado con Node.js y Express, siguiendo una arquitectura MVC (Modelo-Vista-Controlador). Esta arquitectura promueve una clara separación de responsabilidades entre las rutas, controladores y modelos. Todas las rutas están organizadas en app.js, lo que facilita su mantenimiento y escalabilidad. El back tiene autenticación con JWT, validaciones con Zod, integración con Stripe para pagos, y envío de notificaciones por email.

Rutas Principales del Backend (app.js)

- **/auth/** → Autenticación de usuarios
- **/users/** → Gestión de usuarios
- **/productos/** → Catálogo de productos
- **/mesas/** → Gestión de mesas
- **/pedidos/** → Procesamiento de pedidos
- /**detalles-pedidos/** → Detalles de pedidos
- **/reservas/** → Sistema de reservas
- **/images/** → Gestión de imágenes
- **/product-categories/** → Categorías de productos
- **/emails/** → Envío de emails

Rutas de Autenticación (/auth/)

Archivo: userlogController.js

- **POST /auth/login**
  - **Propósito:** Autenticar usuarios
  - **Recibe:** email, password
  - **Devuelve:** Token JWT y nombre del usuario
  - **Extra:** Envía email de notificación
- **POST /auth/register**
  - **Propósito:** Registrar nuevos usuarios
  - **Recibe:** nombre, email, password, telefono
  - **Devuelve:** Confirmación de registro
  - **Extra**: Envía email de bienvenida

Rutas de Gestión de Usuarios (/users/)

Archivo: userDataRoutes.js

GET:

- **/users** → Listar todos los usuarios
- **/users/id/:id** → Buscar usuario por ID
- **/users/tipo/:tipo** → Filtrar usuarios por rol
- **/users/email/:email** → Buscar usuario por email
- **/users/nombre/:nombre** → Buscar usuario por nombre

POST / PUT:

- **POST /users** → Crear nuevo usuario
- **PUT /users** → Actualizar datos del usuario
- **PUT /users/actualizarRol** → Cambiar rol del usuario

DELETE:

- **DELETE /users/:id** → Eliminar usuario

Validación de roles en userModel.js

ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal

Rutas de Productos (/productos/)

Archivo: productDataRoutes.js

GET:

- **/productos** → Todos los productos
- **/productos/productsImages** → Productos con imágenes
- /**productos/grouped** → Agrupados por categoría
- **/productos/id/:id** → Producto por ID
- **/productos/:limi**t → Productos destacados con límite

POST / PUT / DELETE:

- P**OST /producto**s → Crear producto con imágenes
- **PUT /productos** → Actualizar producto
- **DELETE /productos/:id** → Eliminar producto
- **DELETE /productos/:id/images/:filename** → Eliminar imagen específica

Rutas de Pedidos (/pedidos/)

Archivo: pedidoDataRoutes.js

GET:

- **/pedidos** → Todos los pedidos
- **/pedidos/pedidosProductos** → Pedidos con productos
- **/pedidos/id/:id** → Pedido específico
- /**pedidos/usuario/:username** → Pedidos por usuario

POST / PUT / DELETE:

- **POST /pedidos** → Crear pedido (invitado)
- **POST /pedidos/with-user** → Crear pedido (usuario registrado)
- **PUT /pedidos** → Actualizar pedido
- **DELETE /pedidos/:id** → Eliminar pedido

Rutas de Reservas (/reservas/)

Archivo: reservasDataRouter.js

GET:

- **/reservas** → Todas las reservas
- /**reservas/id/:id** → Reserva específica
- **/reservas/usuario/:id** → Reservas por usuario
- **/reservas/mesa/:id** → Reservas por mesa
- **/reservas/fecha/:date** → Reservas por fecha

POST / PUT / DELETE:

- **POST /reservas** → Crear reserva (invitado)
- **POST /reservas/with-user** → Crear reserva (usuario registrado)
- **POST /reservas/checkReservation** → Verificar disponibilidad
- **PUT /reservas** → Actualizar reserva
- **DELETE /reservas/:id** → Eliminar reserva

Rutas de Pagos (/payments/)

Archivo: stripeRoutes.js:

Stripe:

- **POST /payments/create-checkout-session** → Crear sesión de pago
- **GET /payments/verify-payment/:session_id** → Verificar estado de pago
- **POST /payments/webhook** → Webhook para eventos
- **POST /payments/refund** → Reembolsos

Rutas de Emails (/emails/)

Archivo: emailRoutes.js:7

- **POST /emails/send** → Enviar notificaciones por correo electrónico

4.2.3 Capa de Persistencia o de Datos.

La base de datos sigue una estructura relacional, donde las entidades clave están conectadas mediante referencias (foreign keys). Está diseñada para soportar funcionalidades de reservas, pedidos, productos, autenticación de usuarios, y pagos.

1. Usuarios (`users`)

- **Campos principales:**
  - id, nombre, email, password, telefono, rol
- **Roles**: cliente, admin, empleado
- **Relaciones:**
  - Tiene muchos pedidos
  - Tiene muchas reservas

2. Productos (`productos`)

- **Campos principales:**
  - id, nombre, descripcion, precio, categoriaId, stock, imagenes
- **Relaciones:**
  - Pertenece a una categoría (product-categories)
  - Se incluye en muchos detalles de pedido

3. Categorías de Producto (`product-categories`)

- **Campos:**
  - id, nombre, descripcion
- **Relaciones:**
  - Tiene muchos productos

4. Pedidos (pedidos)

- **Campos:**
  - id, usuarioId, fecha, estado, total, metodoPago
- **Relaciones:**
  - Pertenece a un usuario
  - Tiene muchos detalles-pedidos

5. Detalles de Pedido (detalles-pedidos)

- **Campos:**
  - id, pedidoId, productoId, cantidad, precioUnitario
- **Relaciones:**
  - Pertenece a un pedido
  - Hace referencia a un producto

6. Reservas (reservas)

- **Campos:**
  - id, usuarioId, mesaId, fecha, hora, numPersonas, estado
- **Relaciones:**
  - Pertenece a un usuario
  - Asignada a una mesa

7. Mesas (mesas)

- **Campos:**
  - id, numero, capacidad, ubicacion
- **Relaciones:**
  - Puede tener muchas reservas

8. Imágenes (images)

- **Campos:**
  - id, productoId, filename, url
- **Relaciones:**
  - Asociadas a productos

9. 2FA(two_factor_codes)

- **Campos** (posiblemente solo logs):
  - id, user_id, codigo, activo, fecha_creacion

10. Pagos (Stripe, no es una pero relacionado)

- Los pagos se gestionan externamente con Stripe, pero:
  - Se guarda el **session_id** para verificar el estado
  - Relacionado con un **pedido** para saber si está pagado, fallido o pendiente

4.3 Despliegue del Proyecto

1. Requisitos Previos

Antes de comenzar, es necesario tener instaladas las siguientes herramientas:

- Node.js (v22.14.0 o superior)
- MySQL Workbench
- Git
- Visual Studio Code (u otro editor de código)
- Postman (opcional, para pruebas de la API)

2. Clonar el Repositorio

Se debe clonar el repositorio del proyecto desde GitHub:

git clone https://github.com/Tr1pin/bookmymeal
cd bookmymeal

3. Instalación de Dependencias

Desde la raíz del proyecto, instalar las dependencias necesarias:

npm install

4. Configuración de la Base de Datos

1. Abrir MySQL Workbench.
2. Copiar el contenido del archivo bookmymeal.db.sql y ejecutarlo para crear la base de datos y las tablas necesarias.

5. Configuración de Variables de Entorno

Dentro de la carpeta /backend, crear un archivo .env con el siguiente contenido:

```
# URL base para imágenes de productos
BASE_URL=http://localhost:3001

# URL del frontend (para redirecciones)
FRONTEND_URL=http://localhost:4200

EMAIL_SENDER=tu_correo@example.com
EMAIL_SENDER_NAME=BookMyMeal
EMAIL_PASSWORD=tu_contraseña_de_correo
```

6. Configuración del Archivo `config.ts`

En el archivo config.js, ubicado dentro del backend, se debe actualizar la configuración de la conexión a la base de datos con tus credenciales locales:

```
export const DEFAULT_PORT = 3001;

export const DEFAULT_MYSQL_CONECTION = {
  host: 'localhost',
  user: 'root',       // Cambiar si se utiliza otro usuario
  password: '',       // Agregar tu contraseña de MySQL
  database: 'BookMyMeal',
  port: 3306
};

export const SALT_ROUNDS = 10;

export const JWT_SECRET =
'JhgjwIa5OhRkdqQdnPIPUiFXgYSdB7N17FmjbbkozVdOmUXIGZJsda1jdajnsdadast2peasJKq';
```

7. Ejecución del Proyecto

Desde la raíz del proyecto, ejecutar el siguiente comando:

npm run dev

Este comando iniciará tanto el backend como el frontend en modo desarrollo.
Por defecto, estarán disponibles en:

- Backend: http://localhost:3001
- Frontend: http://localhost:4200

6. CONCLUSIÓN

6.1 Valoración Personal del Trabajo Realizado

A continuación, presento un análisis DAFO que resume las principales fortalezas, debilidades, oportunidades y amenazas del proyecto, seguido del correspondiente análisis CAME, que establece las estrategias para mejorar y aprovechar mejor los resultados.

Análisis DAFO

Fortalezas (F)

- Uso de tecnologías modernas (Node.js, Angular, MySQL)
- Proyecto funcional y visualmente bueno
- Buena separación de frontend y backend
- Seguridad buena

Debilidades (D)

- Falta de pruebas
- Documentación clara y estructura organizada

Oportunidades (O)

- Escalabilidad del proyecto a producción
- Posibilidad de usar la API REST para app móviles con API REST

Amenazas (A)

- Posible obsolescencia de dependencias
- Requiere mantenimiento constante para asegurar estabilidad

Análisis CAME

Corregir (Debilidades)

- Implementar tests unitarios y de integración.
- Mejorar de algunos errores en la pasarela de pagos.

Afrontar (Amenazas)
- Establecer rutinas de actualización de paquetes.
- Uso de dependencias que puedes llegar a ser un problema

Mantener (Fortalezas)
- Continuar con una arquitectura clara y modular.
- Reforzar la separación entre capas de frontend/backend.

Explotar (Oportunidades)
- Añadir autenticación social (Google, Facebook).
- Desarrollar versión móvil.
- Ampliar el sistema de reservas con calendario dinámico.

6.2 Posibles Ampliaciones.

- Añadir un recordatorio email para las reservas
- Añadir en el panel de admin una sección de mesas y usuarios

ALUMNO/A: Denís Camblor Cabero
PROYECTO: BookMyMeal

7. BIBLIOGRAFÍA / WEBGRAFÍA

- https://nodejs.org/docs/latest/api/
- https://docs.stripe.com/
- https://nodemailer.com/
- https://expressjs.com/
- https://www.npmjs.com/package/bcrypt
- https://www.npmjs.com/package/jsonwebtoken
- https://www.npmjs.com/package/mysql2
- https://zod.dev/
- https://www.npmjs.com/package/multer
- https://v19.angular.dev/overview