

CLICK MUNCH
INGENIERÍA DE SOFTWARE

PRESENTADO POR:

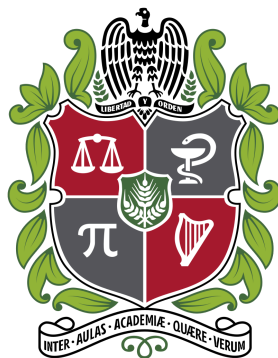
Michael Stiven Betancourt Gelves

Santiago Bejarano Ariza

Johan Sebastian Roa Rodriguez

PROFESOR

Oscar Eduardo Alvarez Rodriguez



Universidad Nacional de Colombia
Facultad de Ingeniería
2024

ÍNDICE

| | |
|---|-----------|
| 1. Origen de la Idea | 3 |
| 1.1 La Identificación del Problema | 3 |
| 1.2 El Proceso de Decisión: De la Idea al Proyecto | 4 |
| 1.3 Click Munch: La Solución Ideal | 5 |
| 2. Levantamiento de Requerimientos | 5 |
| 2.1. Usuarios Externos – Clientes | 6 |
| Requerimientos Funcionales | 6 |
| Requerimientos No Funcionales | 7 |
| 2.2. Usuarios Internos – Administradores de Restaurantes | 7 |
| Requerimientos Funcionales | 7 |
| Requerimientos No Funcionales | 8 |
| 2.3. Usuarios Internos – Cocineros | 8 |
| Requerimientos Funcionales | 8 |
| Requerimientos No Funcionales | 9 |
| 2.4 Requerimientos Principales | 9 |
| 3. Análisis de requerimientos | 10 |
| Análisis Gestión del software | 12 |
| Tiempo | 12 |
| Costos salariales | 13 |
| Alcance | 15 |
| Diseño y arquitectura | 16 |
| Arquitectura del sistema | 16 |
| Diseño de bases de datos | 16 |
| Patrones de diseño | 17 |

1. Origen de la Idea

El concepto de Click Munch nació de un problema cotidiano que muchos de nosotros hemos experimentado: la falta de una plataforma integral que permita reservar en cualquier restaurante y, al mismo tiempo, administrar los pedidos en tiempo real dentro del establecimiento. Aunque existen aplicaciones enfocadas en la reserva de mesas y otras dedicadas a pedidos a domicilio, no había una solución que reuniera ambas funcionalidades en un solo sistema eficiente y accesible para cualquier tipo de restaurante.

1.1 La Identificación del Problema

Todo comenzó cuando, en una de nuestras reuniones iniciales para la selección del proyecto, cada miembro del equipo propuso diferentes ideas basadas en experiencias personales y problemáticas que habíamos observado en distintos sectores. Fue en medio de este diálogo donde uno de los integrantes del equipo, quien había trabajado en el sector de restaurantes, expuso una serie de dificultades recurrentes que había identificado tanto desde la perspectiva de los clientes como desde la gestión interna del negocio.

Nos compartió su experiencia trabajando en un restaurante de mediana escala, donde la administración de las mesas y los pedidos generaba constantes problemas, especialmente en horas pico. A partir de esto, empezamos a debatir y a plantearnos preguntas clave:

- ¿Por qué todavía hay tantos restaurantes que dependen de reservas manuales por teléfono cuando la mayoría de las actividades se han digitalizado?
- ¿Por qué los restaurantes pequeños y medianos no tienen acceso a un sistema de reservas eficiente, dejando a los clientes sin certeza de disponibilidad?
- ¿Por qué el proceso interno de toma y administración de pedidos sigue dependiendo de métodos tradicionales como papel y comunicación verbal, lo que provoca errores y demoras?

- ¿Existe alguna plataforma que permita gestionar reservas y, al mismo tiempo, optimizar la administración de pedidos en tiempo real dentro del restaurante?

Al analizar el mercado, nos dimos cuenta de que ninguna aplicación existente ofrecía una solución completa. Algunas plataformas permitían hacer reservas, pero solo en ciertos restaurantes de lujo o cadenas grandes. Otras se especializaban en pedidos a domicilio, pero no resolvían la gestión de pedidos en mesa. Faltaba una herramienta que ofreciera a los restaurantes de cualquier tamaño la posibilidad de digitalizar ambos procesos de manera integrada.

1.2 El Proceso de Decisión: De la Idea al Proyecto

A medida que discutimos las distintas ideas en equipo, esta propuesta comenzó a destacar por su relevancia, aplicabilidad y potencial impacto en el sector gastronómico. Para asegurarnos de que realmente era una necesidad en el mercado, decidimos hacer una breve investigación:

1. Observamos restaurantes locales para entender cómo manejaban sus reservas y pedidos.
2. Entrevistamos clientes para saber si se sentían frustrados por la falta de un sistema que les permitiera reservar mesas en cualquier lugar.
3. Conversamos con empleados y administradores de restaurantes para validar si realmente la gestión de pedidos en tiempo real era un desafío operativo.

Los resultados fueron claros: tanto clientes como empleados coincidían en que la falta de digitalización en estos procesos generaba problemas innecesarios.

- Los clientes querían poder reservar una mesa sin llamadas telefónicas ni incertidumbre.
- Los restaurantes necesitaban una mejor forma de administrar sus pedidos y mesas sin depender del papel ni de largas explicaciones verbales.

Después de compartir estos hallazgos, tomamos la decisión final. Votamos por unanimidad que desarrollaremos Click Munch, una plataforma que no solo permitiera a los clientes reservar mesas de manera rápida y sencilla, sino que

también ofreciera a los restaurantes herramientas para optimizar la toma de pedidos y mejorar su eficiencia operativa.

1.3 Click Munch: La Solución Ideal

Así nació la idea de Click Munch, una aplicación que combina:

- Un sistema de reservas universal, donde cualquier restaurante puede registrarse y gestionar sus mesas en tiempo real.
- Un sistema de administración de pedidos en mesa, que digitaliza la comunicación entre meseros y cocina, reduciendo errores y tiempos de espera.
- Un panel de control para restaurantes, con estadísticas sobre ocupación, tiempos de atención y platillos más vendidos.

Con esta solución, no solo mejoramos la experiencia de los clientes, sino que también optimizamos el funcionamiento interno de los restaurantes, reduciendo costos operativos y mejorando la organización.

Nuestro objetivo con Click Munch es que cualquier restaurante, sin importar su tamaño, pueda modernizar su gestión sin necesidad de grandes inversiones en tecnología. Creemos que esta herramienta transformará la forma en que los restaurantes operan y cómo los clientes disfrutan de su experiencia gastronómica.

2. Levantamiento de Requerimientos

Después de identificar la necesidad de digitalizar la gestión de reservas y pedidos en restaurantes, el equipo realizó un análisis detallado para definir los requerimientos clave del sistema. Click Munch está diseñado para satisfacer las necesidades de tres tipos principales de usuarios:

1. Clientes (usuarios externos que hacen reservas y pedidos).
2. Administradores de restaurantes (encargados de gestionar reservas, pedidos y estadísticas).

3. Cocineros (responsables de recibir y procesar los pedidos en la cocina).

A continuación, se presentan los requerimientos específicos de cada usuario:

2.1. Usuarios Externos – Clientes

Los clientes son los usuarios principales del sistema. Su objetivo es encontrar un restaurante, reservar una mesa y hacer pedidos de manera eficiente. También buscan información sobre el restaurante y el estado de su pedido en tiempo real.

Requerimientos Funcionales

- Exploración de restaurantes: El cliente debe poder ver la lista de restaurantes registrados en la plataforma, filtrar por ubicación, tipo de comida y calificación.
- Visualización del menú digital: Cada restaurante debe mostrar su menú actualizado con precios y disponibilidad.
- Reservas en línea: Los clientes deben poder seleccionar un restaurante, elegir fecha, hora y número de personas para reservar una mesa.
- Confirmación de reserva: El sistema debe enviar una confirmación automática por correo electrónico o notificación en la aplicación.
- Modificación o cancelación de reservas: El cliente debe tener la opción de modificar o cancelar su reserva antes de la hora programada.
- Realización de pedidos: El cliente debe poder hacer un pedido desde la mesa, agregando platillos y personalizando ingredientes si el restaurante lo permite.
- Seguimiento del estado del pedido: El sistema debe mostrar en tiempo real el estado de preparación del pedido.
- Notificaciones en tiempo real: El sistema debe notificar al cliente cuando el pedido esté en preparación, listo para servir o entregado.
- Valoraciones y comentarios: Los clientes deben poder calificar la experiencia y dejar comentarios sobre el restaurante y la comida.
- Historial de reservas y pedidos: El cliente debe poder consultar sus reservas y pedidos anteriores.

Requerimientos No Funcionales

- Diseño intuitivo: La interfaz debe ser clara y fácil de usar.
- Rendimiento óptimo: La búsqueda de restaurantes y menús debe responder en menos de 3 segundos.
- Seguridad: Los datos personales deben estar protegidos y cumplir con normativas de privacidad.

2.2. Usuarios Internos – Administradores de Restaurantes

Los administradores son los encargados de gestionar la operación del restaurante en la plataforma. Su objetivo es administrar reservas, pedidos y obtener estadísticas sobre el rendimiento del negocio.

Requerimientos Funcionales

- Registro de restaurante: Los administradores deben poder crear una cuenta y registrar su restaurante en la plataforma.
- Gestión de disponibilidad de mesas: El sistema debe permitir configurar la cantidad de mesas y horarios disponibles para reservas.
- Administración del menú digital: El administrador debe poder agregar, modificar o eliminar platillos del menú, incluyendo imágenes, descripciones y precios.
- Visualización de reservas en tiempo real: El administrador debe poder ver todas las reservas activas, los clientes próximos a llegar y el estado de ocupación del restaurante.
- Confirmación automática o manual de reservas: El administrador debe poder configurar si las reservas se confirman automáticamente o si requieren aprobación manual.
- Gestión de pedidos internos: Debe permitir visualizar los pedidos de cada mesa y su estado de preparación.
- Gestión de notificaciones: El sistema debe permitir enviar notificaciones a los clientes sobre el estado de su pedido o cambios en la reserva.

- Estadísticas y métricas del negocio: El administrador debe poder consultar datos sobre:
 - Platos más vendidos.
 - Horarios de mayor demanda.
 - Tiempo promedio de atención por pedido.
 - Cancelaciones de reservas y pedidos.
- Control de usuarios: Debe permitir la asignación de roles y permisos dentro del restaurante (ejemplo: meseros, cocineros, administradores).

Requerimientos No Funcionales

- Seguridad de datos: Solo los administradores deben tener acceso a la información confidencial del restaurante.
- Escalabilidad: La plataforma debe soportar múltiples restaurantes y un alto número de usuarios simultáneos sin pérdida de rendimiento.
- Accesibilidad: La interfaz debe ser fácil de usar para cualquier administrador sin conocimientos técnicos avanzados.

2.3. Usuarios Internos – Cocineros

Los cocineros utilizan el sistema para recibir, organizar y gestionar pedidos en tiempo real. Su objetivo es evitar errores y mejorar la eficiencia de la cocina.

Requerimientos Funcionales

- Visualización de pedidos en tiempo real: El sistema debe mostrar en pantalla los pedidos que han sido enviados desde la mesa.
- Organización de pedidos por prioridad: Los pedidos deben clasificarse por orden de llegada y nivel de urgencia (ejemplo: platos que requieren mayor tiempo de preparación deben mostrarse con prioridad).
- Cambio de estado del pedido: Los cocineros deben poder actualizar el estado del pedido en tiempo real (ejemplo: "En preparación", "Listo para servir").
- Notificaciones a meseros o clientes: Una vez que el pedido esté listo, el sistema debe notificar automáticamente al mesero o al cliente.

- Historial de pedidos: Se debe poder consultar los pedidos anteriores en caso de reclamos o ajustes.

Requerimientos No Funcionales

- Interfaz optimizada para la cocina: La pantalla debe ser clara, con un diseño que facilite la lectura rápida de los pedidos.
- Sistema de alertas: Debe haber un sistema de alertas visuales o sonoras para pedidos prioritarios.
- Fiabilidad: Debe garantizarse que los pedidos no se pierdan o dupliquen por fallas en la conexión.

2.4 Requerimientos Principales

A partir de los requerimientos identificados, se definieron las siguientes funcionalidades clave para **Click Munch**:

| Requerimientos | Clientes | Administradores | Cocineros |
|---|----------|-----------------|-----------|
| Exploración de restaurantes | ✓ | ✗ | ✗ |
| Visualización de menús digitales | ✓ | ✓ | ✗ |
| Reservas en línea | ✓ | ✓ | ✗ |
| Modificación/cancelación de reservas | ✓ | ✓ | ✗ |
| Realización de pedidos | ✓ | ✓ | ✗ |
| Seguimiento del pedido | ✓ | ✓ | ✓ |
| Notificaciones en tiempo real | ✓ | ✓ | ✓ |
| Valoraciones y comentarios | ✓ | ✓ | ✗ |
| Historial de reservas/pedidos | ✓ | ✓ | ✓ |
| Gestión de disponibilidad de mesas | ✗ | ✓ | ✗ |
| Gestión del menú digital | ✗ | ✓ | ✗ |
| Gestión de pedidos internos | ✗ | ✓ | ✓ |
| Estadísticas y métricas | ✗ | ✓ | ✗ |
| Sugerencia de restaurantes por cercanía | ✓ | ✗ | ✗ |

3. Análisis de requerimientos

| Funcionalidad | Estimación MoSCoW | Días (Fibonacci) | Justificación |
|--------------------------------------|-------------------|------------------|---|
| Exploración de restaurantes | Must | 5 | Se requiere desarrollar un módulo de búsqueda con filtros que interactúe con la base de datos para recuperar información sobre los restaurantes. Se debe garantizar una consulta eficiente, utilizando índices y optimización para mejorar la velocidad de respuesta. La integración con otros módulos es mínima, pero debe asegurarse la sincronización con la disponibilidad de cada restaurante. |
| Visualización de menús digitales | Must | 5 | Se debe permitir que los restaurantes carguen menús actualizados, incluyendo imágenes y descripciones almacenadas en la base de datos. Se requiere integración con el módulo de pedidos para validar la disponibilidad de platillos y con la gestión de restaurantes para administrar los menús. Además, se debe optimizar el acceso a imágenes mediante almacenamiento en la nube. |
| Reservas en línea | Must | 8 | La lógica de disponibilidad de mesas requiere consultas en tiempo real a la base de datos, asegurando la consistencia de las reservas. Debe integrarse con el módulo de notificaciones para confirmar las reservas y con la gestión de disponibilidad para evitar sobreasignaciones. Se debe manejar concurrencia para prevenir reservas dobles en horarios de alta demanda. |
| Modificación/cancelación de reservas | Should | 5 | Se debe implementar una solución en base de datos para eliminar o modificar reservas en dado caso de algún imprevisto del restaurante o del cliente. Esta funcionalidad debe notificar a ambas partes del ajuste. |

| | | | |
|-------------------------------------|--------|----|---|
| Realización de pedidos | Must | 13 | Se debe implementar un sistema que permita a los clientes realizar pedidos interactuando con el menú digital. Este módulo debe integrarse con la base de datos para validar disponibilidad de platillos y con el módulo de cocina para enviar órdenes. |
| Seguimiento del estado del pedido | Must | 8 | Se necesita comunicación constante entre la cocina, el mesero y el cliente para actualizar el estado del pedido. La dificultad técnica radica en la integración con el módulo de gestión de pedidos y la necesidad de una actualización en tiempo real utilizando WebSockets o polling en la base de datos para reflejar cambios en los estados de los pedidos. |
| Notificaciones en tiempo real | Should | 5 | Se deben implementar eventos en tiempo real que notifiquen cambios en reservas y pedidos a los clientes. Esto requiere la integración con módulos de pedidos, reservas y estado del pedido. Técnicamente, es necesario usar Firebase Cloud Messaging o WebSockets para mantener la comunicación activa y eficiente. |
| Gestión de disponibilidad de mesas | Should | 5 | Se debe desarrollar un sistema de control que maneje la ocupación de mesas en tiempo real. Este módulo se integra con la base de datos de reservas y con la interfaz de administración del restaurante para permitir ajustes en la disponibilidad. Es necesario manejar concurrencia para evitar conflictos en la asignación de mesas. |
| Gestión de pedidos en tiempo real | Should | 8 | Este módulo requiere coordinación entre clientes, meseros y cocina. Debe integrarse con el módulo de pedidos, cocina y notificaciones en tiempo real. Se necesita una arquitectura eficiente que permita manejar múltiples pedidos simultáneamente sin retrasos, lo que implica el uso de colas de mensajes o tecnologías como Redis para optimizar la gestión. |
| Estadísticas y métricas del negocio | Could | 5 | Requiere la extracción y procesamiento de datos de pedidos, reservas y ventas. La dificultad técnica está en la optimización de consultas para manejar grandes volúmenes de información sin |

| | | | |
|--|-------|----|--|
| | | | afectar el rendimiento del sistema. Se recomienda el uso de bases de datos analíticas o almacenamiento en caché para mejorar la velocidad de respuesta. |
| Historial de reservas y pedidos | Could | 5 | Se debe permitir a los clientes y administradores acceder a registros anteriores. La dificultad técnica está en optimizar consultas para recuperar datos sin afectar el rendimiento del sistema. Se puede usar paginación o almacenamiento en caché para mejorar la eficiencia. |
| Sistema de valoraciones y comentarios | Could | 3 | CRUD básico de reseñas y puntuaciones. La dificultad técnica es mínima, pero se debe integrar con la base de datos de restaurantes y garantizar la moderación de comentarios para evitar contenido inapropiado. |
| Sugerencias de restaurantes por cercanía | Must | 8 | Se debe desarrollar un sistema de geolocalización que recomiende restaurantes según la ubicación del usuario. Requiere integración con APIs de mapas como Google Maps y con la base de datos de restaurantes para filtrar resultados. La dificultad técnica radica en manejar consultas eficientes para evitar sobrecarga en la base de datos. |
| | | 83 | |

Análisis Gestión del software

Tiempo

Dado que en el documento ya se tienen estimaciones en días para cada funcionalidad y se ha usado la serie de Fibonacci como base, podemos organizar el desarrollo en **sprints de 2 semanas (10 días hábiles)**.

Fases del Desarrollo

1. **Sprint 1 - Configuración Inicial y Base del Proyecto (10 días)**
 - a. Configuración del entorno de desarrollo
 - b. Diseño preliminar de la base de datos
 - c. Configuración de la arquitectura MVC (Spring Boot y React)

- d. Desarrollo del sistema de autenticación (Inicio de sesión y registro)
- 2. **Sprint 2 - Funcionalidades Principales del Cliente (10 días)**
 - a. Exploración de restaurantes (5 días)
 - b. Visualización de menús digitales (5 días)
- 3. **Sprint 3 - Gestión de Reservas y Pedidos (10 días)**
 - a. Implementación de reservas en línea (8 días)
 - b. Modificación y cancelación de reservas (5 días)
 - c. Gestión de disponibilidad de mesas (5 días)
- 4. **Sprint 4 - Sistema de Pedidos y Cocina (10 días)**
 - a. Realización de pedidos (13 días, podría extenderse a un sprint y medio)
 - b. Seguimiento del estado del pedido (8 días)
 - c. Gestión de pedidos en tiempo real (8 días)
- 5. **Sprint 5 - Notificaciones y Optimización de Interfaz (10 días)**
 - a. Implementación de notificaciones en tiempo real (5 días)
 - b. Historial de reservas y pedidos (5 días)
 - c. Sistema de valoraciones y comentarios (3 días)
- 6. **Sprint 6 - Panel de Administración y Métricas (10 días)**
 - a. Gestión del menú digital (13 días, podría extenderse a otro sprint)
 - b. Estadísticas y métricas del negocio (5 días)
- 7. **Sprint 7 - Sugerencias y Mejoras (10 días)**
 - a. Implementación de sugerencia de restaurantes por cercanía (8 días)
 - b. Integración de mejoras en UX/UI y corrección de errores detectados
- 8. **Sprint 8 - Pruebas Finales y Optimización (10 días)**
 - a. Pruebas de integración y estrés
 - b. Corrección de errores críticos
 - c. Optimización de rendimiento y seguridad

Costos salariales

| Categoría | Cantidad | Salario Mensual (COP) | Fuente |
|----------------------|----------|-----------------------|---|
| Desarrollador Junior | 2 | \$5.000.000 | https://talently.tech/herramientas/colombia/salario?utm_source=chatgpt.com |

| | | | |
|----------------------|---|--------------|---|
| Desarrollador Senior | 1 | \$11.500.000 | https://talently.tech/herramientas/colombia/salario?utm_source=chatgpt.com |
| Tester QA | 1 | \$2.900.000 | https://www.cooltesters.com/blog/salarios-en-software-testing-2022-colombia?utm_source=chatgpt.com |
| Diseñador UX/UI | 1 | \$4.000.000 | https://co.talent.com/salario?job=ux+ui&utm_source=chatgpt.com |

Infraestructura y Herramientas

| Elemento | Costo Mensual (COP) | Referencia |
|----------------------------|---------------------|---|
| Servicios en la Nube | \$645.000 | Estimación basada en proveedores como AWS y Azure. |
| Base de datos PostgreSQL | \$215.000 | aws.amazon.com |
| APIs Externas | \$860.000 | Estimación basada en proveedores comunes de APIs. |
| Herramientas de desarrollo | \$430.000 | Estimación basada en herramientas como GitHub, Postman, Figma, etc. |

| Mano de obra | Infraestructura | Costo por mes |
|--------------|-----------------|---------------|
| 28.400.000 | 2.150.000 | 30.550.000 |

Dado que el proyecto según la estimación fibonacci da un total de 83 días el costo total del proyecto aproximado a tres meses sería de 91.650.000.

Alcance

Funcionalidades Incluidas en el MVP

Para el MVP (Producto Mínimo Viable), priorizamos las funcionalidades clave para que los usuarios puedan buscar restaurantes, realizar pedidos y hacer reservas.

Incluimos:

1. Acceso a Menús Digitales
 - Búsqueda y filtros por ubicación, tipo de comida, palabras clave.
 - Optimización para conexiones lentas.
2. Reserva de Mesas
 - Registro de reservas con confirmaciones automáticas.
 - Notificaciones recordatorias para los usuarios.
3. Realización de Pedidos
 - Pedido en restaurante, para recoger y a domicilio.
 - Personalización de platillos y validación de disponibilidad.
4. Seguimiento del Estado del Pedido
 - Notificaciones en tiempo real sobre el estado del pedido.
 - Historial de pedidos.

Funcionalidades Post-MVP

Las siguientes funcionalidades se implementarán en versiones futuras:

5. Sugerencia de Restaurantes por Cercanía
 - Recomendaciones basadas en la ubicación del usuario.
 - Filtros según tipo de comida y valoraciones.
6. Valoraciones y Comentarios
 - Sistema de clasificaciones y reseñas de usuarios.
7. Gestión de Menús para Restaurantes
 - Interfaz para que los restaurantes editen y actualicen sus menús.
8. Gestión de Pedidos en Tiempo Real
 - Tablero de control para visualizar pedidos en vivo.
9. Estadísticas y Métricas de Ventas
 - Reportes de ventas, horarios de mayor demanda.

Diseño y arquitectura

Arquitectura del sistema

En cuanto a la arquitectura del sistema elegimos un modelo vista controlador (MVC), esto debido a que esta nos permite optimizar los recursos y nos deja tener escalabilidad sobretodo a largo plazo, también el MVC nos facilita la organización del código y poder separar mejor las responsabilidades en cuanto al modelo de la base de datos, el frontend que es la interfaz del usuario en react y la lógica del negocio y el controlador en springboot. También la escogimos por el hecho de que miramos este proyecto a largo plazo entonces el sistema debe ser capaz de poder crecer sin una pérdida de rendimiento teniendo escalabilidad fácil debido a que cada componente es independiente por lo que por ejemplo el front se puede cambiar sin alterar la base de datos, se pueden agregar funciones nuevas sin modificar o alterar todo el sistema y adams que springboot deja que el backend pueda escalar si es necesario en el futuro con microservicios sin afectar todo el funcionamiento de la página.

Diseño de bases de datos

En cuanto a las decisiones tomadas para el diseño del esquema de la base de datos tenemos esto:

Normalización

En cuanto a la normalización tratamos de eliminar redundancias dividiendo las entidades en tablas relacionadas, además de esto creamos relaciones entre entidades.

Índices

Para los índices tratamos establecer correctamente entre las claves primarias y foráneas para poder optimizar las consultas y mantenibilidad, también el email de los usuarios es único para evitar los duplicados en los datos, aún quedan añadir los índices pero es algo si se piensa implementar debido a que los índices son estructuras que optimizan mucho la búsqueda en las tablas de la base de datos, nos deja buscar la información sin tener que hacer todo el recorrido por los registros.

Claves primarias y foráneas

En cuanto a este punto cada tabla tiene su clave primaria serial y las relaciones están puestas con claves foráneas para que se pueda asegurar la integridad de los datos.

Restricciones

Para las restricciones establecimos valores no nulos en campos clave que lo necesitaban y pusimos un check en algunos campos importantes para que no puedan introducir valores inválidos.

Escalabilidad

Este apartado de la escalabilidad se evidencia en que por ejemplo hay una relación entre pedidos y detalles_pedido que hace que se puedan poner múltiples platillos en un pedido además que al tener los menús y platillos separados ayuda a que sea más fácil las actualizaciones en las respectivas categorías.

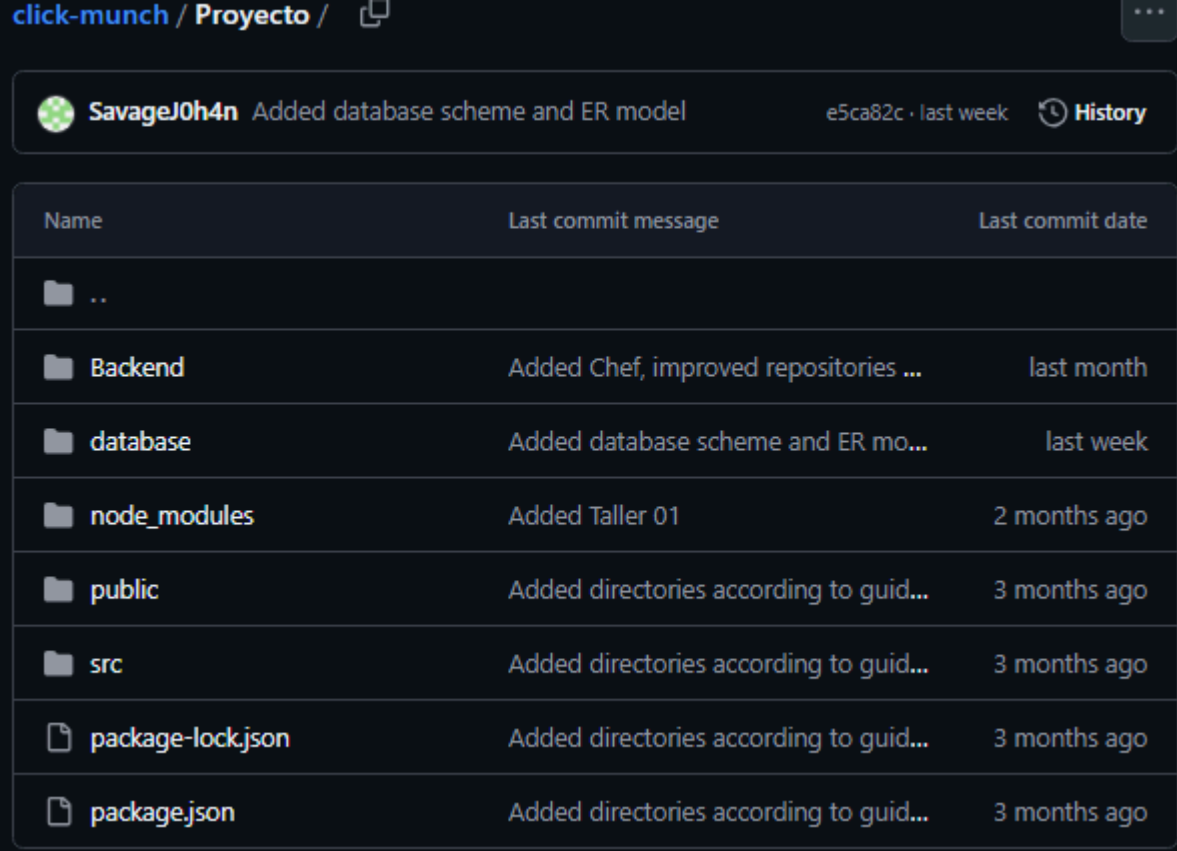
Para concluir este apartado de el diseño de la bases de datos hay que decir que escogimos una base de datos relacional debido a varios factores como que necesitábamos tener relaciones bien definidas entre entidades ya que en un restaurante hay muchas relaciones entre datos, necesitábamos usar la normalización y el uso de claves primarias y foráneas para evitar que haya datos huérfanos o inconsistentes, también es necesario que haya eficiencia y que se puedan realizar consultas complejas y por último también era necesario tener una escalabilidad y estabilidad en la base para manejar una gran cantidad de datos y conexiones simultaneas sin perder estabilidad.

Patrones de diseño

En cuanto al uso de algún patrón de diseño, el proyecto está aún en fases muy tempranas pero estamos tratando de implementar como patrón un Modelo Vista Controlador de tal forma que nos va a ayudar a separar la lógica del proyecto, la interfaz de usuario y el acceso a datos en capas diferentes. Esto nos ayuda a que el proyecto pueda tener fácil mantenimiento y escalabilidad, por otra parte también se asegura la reutilización del código en el proyecto y el poder hacer la separación de responsabilidades.

Tratamos de hacerlo de esta forma para que pudiéramos trabajar en el backend en Spring Boot sin tener que acoplar al frontend y que el front sólo tuviese que traer datos del back mediante una API REST, además también lo hicimos de esta forma para poder gestionar de una forma estructurada los datos en la base en Postgre.

Aún está muy en fase temprana el acoplar bien este patrón de desarrollo al proyecto pero más o menos estamos tratando de separar todo de esta manera:



The screenshot shows a file explorer interface with a dark theme. At the top, there is a breadcrumb navigation bar with the text "click-munch / Proyecto /" and a copy icon. Below this, a commit message "SavageJ0h4n Added database scheme and ER model" is displayed, along with a commit hash "e5ca82c" and the text "last week". A "History" button with a clock icon is also present. The main area contains a table with three columns: "Name", "Last commit message", and "Last commit date". The table lists the following items:

| Name | Last commit message | Last commit date |
|-------------------|--|------------------|
| .. | | |
| Backend | Added Chef, improved repositories ... | last month |
| database | Added database scheme and ER mo... | last week |
| node_modules | Added Taller 01 | 2 months ago |
| public | Added directories according to guid... | 3 months ago |
| src | Added directories according to guid... | 3 months ago |
| package-lock.json | Added directories according to guid... | 3 months ago |
| package.json | Added directories according to guid... | 3 months ago |

De tal forma que la Base de datos y las entidades en Spring Boot representan los datos y la lógica del negocio por ejemplo que las clases Usuario, Pedido, Restaurante tengan su mapeo en la base de datos, luego que el front sólo consuma mediante llamadas HTTP los datos del back así solo encargándose de todo lo que tiene que ver con la experiencia de usuario y la interfaz gráfica y todo esto funciona con una API REST que gestiona las solicitudes HTTP y comunica la vista con el modelo teniendo en el código varios controladores que expongan endpoints como en los pedidos o usuarios.