ESCUELA DE EDUCACIÓN SUPERIOR TECNOLÓGICA PRIVADA LA PONTIFICIA

PROGRAMACIÓN DE SERVICIOS WEB I

EN LA MODALIDAD DE PROYECTO PRODUCTIVO Y/O PROYECTO DE INNOVACIÓN

Informe Ejecutivo del avance del proyecto de la carrera profesional de Ingeniería de Sistemas de Información

Informe correspondiente al Módulo 2

CONSTRUCCIÓN DE SISTEMAS MULTIPLATAFORMA

desarrollado por el estudiante

MEJÍA CHUCHON CLINTON

AVENDAÑO AUCCATOMA JHONS

RAMOS PARIOMA ANJALI

Huamanga, abril del 2024

Introducción

En la era digital actual, las bases de datos juegan un papel fundamental en la gestión eficiente de los servicios y operaciones diarias de las empresas. En particular, la industria restaurantera se ha beneficiado enormemente del uso de bases de datos para gestionar inventarios, pedidos y facturación. La confiabilidad y la capacidad de respuesta de estos sistemas son esenciales para el buen funcionamiento del negocio, ya que las decisiones comerciales dependen en gran medida de la información precisa y actualizada.

Este proyecto aborda la necesidad de un sistema de gestión para restaurantes, donde los clientes puedan interactuar fácilmente con un menú, personalizar sus pedidos y realizar pagos de manera segura. Al mismo tiempo, los administradores tendrán la capacidad de gestionar los ingredientes, categorías de productos, menús y controlar el estado de los pedidos. A través de una plataforma web intuitiva, tanto compradores como dueños de restaurantes podrán acceder a un sistema eficiente, que optimiza la experiencia de usuario y mejora la administración operativa del restaurante.

Este sistema se implementará utilizando **MySQL** como base de datos relacional y **Spring Boot** como framework de desarrollo backend, lo que garantiza un manejo eficiente de los datos y una arquitectura escalable. Además, se integrará **Stripe** para el procesamiento seguro de pagos. La aplicación permitirá a los usuarios registrarse con roles distintos, ofreciendo distintos niveles de acceso y privilegios según su perfil (comprador o dueño de restaurante). A través de esta solución, se espera mejorar la gestión de pedidos, la personalización del menú y la eficiencia operativa del restaurante.

Descripción del Sistema

El sistema propuesto será una aplicación web interactiva que conectará a los clientes con el restaurante a través de una interfaz fácil de usar. Los clientes podrán ver el menú de platos, personalizar su pedido agregando ingredientes y elegir la ubicación para la entrega del pedido. A través de una integración con **Stripe**, los pagos serán procesados de forma segura, y los pedidos se actualizarán en tiempo real según su estado.

Por otro lado, los administradores tendrán acceso a una sección donde podrán crear y gestionar ingredientes, categorías de productos, y menús. Además, podrán supervisar los pedidos, cambiar su estado según su progreso y asegurarse de que se entreguen correctamente.

El sistema garantizará una experiencia de usuario fluida tanto para los clientes como para los administradores, utilizando tecnologías modernas y una arquitectura robusta que soporte el crecimiento y escalabilidad del negocio.

Objetivos y Alcances

- Diseñar e implementar una plataforma web que permita a los clientes navegar por el menú, personalizar sus pedidos y realizar pagos en línea.
- Proveer a los administradores con herramientas para gestionar ingredientes, categorías y menús, así como el seguimiento y control de los pedidos.
- Utilizar **Spring Boot** para el backend, **MySQL** para la base de datos y **Stripe** para el procesamiento de pagos.
- Garantizar una interfaz intuitiva y fácil de usar para ambos tipos de usuarios (clientes y administradores), optimizando la experiencia de compra y gestión.

Este sistema proporcionará una solución efectiva para mejorar la gestión de un restaurante y optimizar la experiencia del cliente.

Conceptualización del sistema

Casos de uso a Implementar

Se tienen los siguientes casos de uso que se implementarán en la aplicación:

Nombre del Caso de Uso	Realizar Pedido
Actores	Cliente
Precondiciones	El cliente debe estar registrado e iniciar sesión
Post-condiciones	El sistema registra el pedido y lo deja en estado "En Espera".
Flujo Básico	1. El cliente navega por la página y selecciona los platos deseados. 2. El cliente puede añadir ingredientes adicionales a los platos seleccionados. 3. Los platos se agregan al carrito de compras.

	4. El cliente proporciona su ubicación para el delivery. 5. El cliente hace clic en el botón "Pagar". 6. El sistema muestra un modal de Stripe para realizar el pago. 7. Una vez confirmado el pago, el pedido queda registrado en el sistema en estado "En Espera".
Flujo Alterno	Si el cliente no está registrado: 1. El sistema redirige al cliente a la página de registro. Si no se realiza el pago: 1. El sistema informa al cliente que el pago no fue exitoso y permite reintentar.

Resultado	El sistema registra el pedido con los detalles proporcionados y lo deja en estado inicial.

Nombre del Caso de Uso	Gestionar Pedidos
Actores	Administrador
Precondiciones	Debe iniciar sesión como administrador.
Post-condiciones	El estado del pedido puede ser modificado según corresponda
Flujo Básico	1. El administrador accede a la sección de "Órdenes". 2. Se muestra la lista de pedidos junto con sus estados.

	3. El administrador selecciona un pedido y cambia su estado (Pendiente, Pagado, En Proceso, Entregado, etc.). 4. El sistema actualiza el estado del pedido en la base de datos.
Flujo Alterno	Si el administrador no tiene permisos: 1. El sistema informa que no tiene acceso para realizar esta acción.
Resultado	El pedido cambia de estado según la gestión del administrador.

Nombre del Caso de Uso	Registrar Producto
Actores	Administrador
Precondiciones	El producto no debe estar registrado en el sistema.
Post-condiciones	El nuevo producto queda registrado en la base de datos.
Flujo Básico	1. El administrador accede a la sección de "Menú". 2. Se muestra un formulario con campos para ingresar nombre, precio, categoría e ingredientes. 3. El administrador completa el formulario y carga una imagen del producto. 4. Hace clic en el botón "Registrar". 5. El sistema guarda el producto y confirma su registro.
Flujo Alterno	Si el producto ya está registrado: 1. El sistema informa que el producto ya existe y solicita un nuevo ingreso.

Resultado	El producto queda registrado en el sistema y aparece en el menú

Nombre del Caso de Uso	Modificar Producto
Actores	Administrador
Precondiciones	El producto debe estar registrado en el sistema..
Post-condiciones	El producto queda actualizado en el sistema.

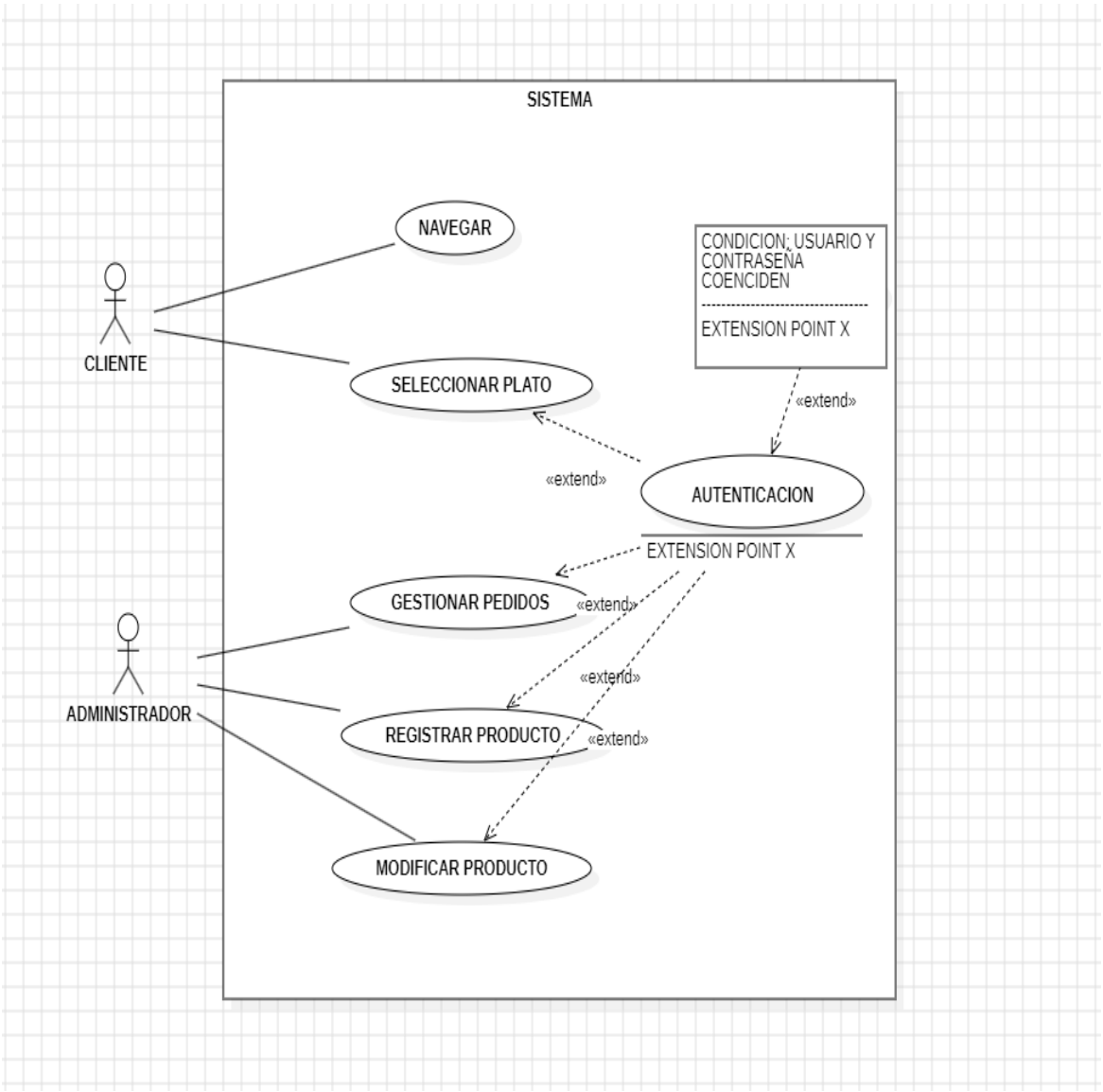
Flujo Básico	1. El administrador selecciona un producto de la lista en la sección de "Menú". 2. Hace clic en el botón "Modificar". 3. Se muestran los campos del producto actual para ser editados (nombre, precio, categoría, ingredientes). 4. El administrador realiza los cambios y sube una nueva imagen si es necesario. 5. Hace clic en "Guardar". 6. El sistema actualiza el producto y confirma los cambios.
Flujo Alterno	Si el producto no está registrado: 1. El sistema informa que el producto no existe.
Resultado	El producto queda actualizado con los nuevos detalles.

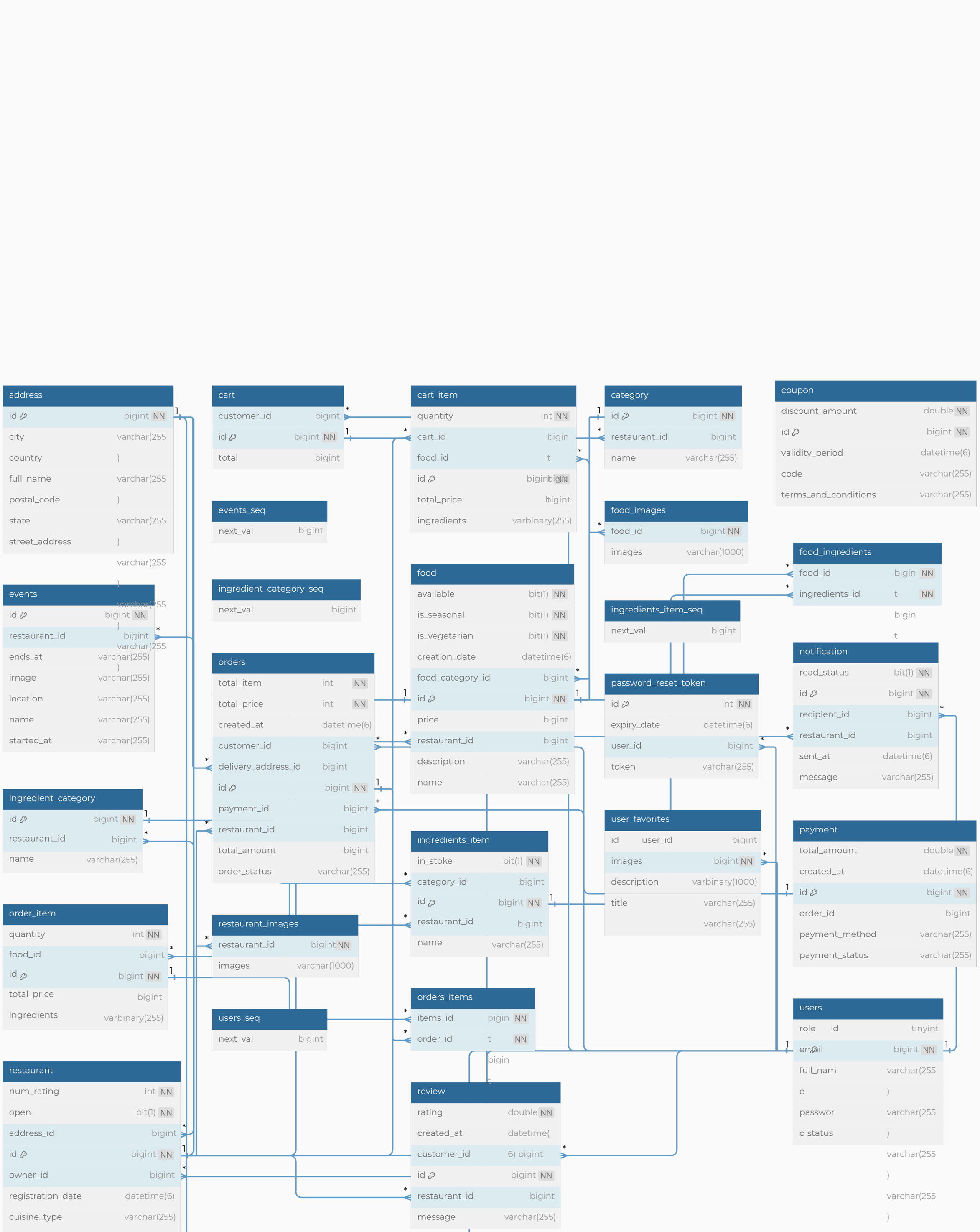
Nombre del Caso de Uso	Registrar Usuario
Actores	Cliente o Administrador
Precondiciones	El usuario no debe estar registrado.
Post-condiciones	El nuevo usuario queda registrado en el sistema con los privilegios correspondientes.

Flujo Básico	1. El cliente o administrador accede al formulario de registro. 2. Completa los campos obligatorios (nombre, correo, contraseña, tipo de usuario). 3. Hace clic en "Registrar". 4. El sistema guarda el usuario en la base de datos y confirma su registro.

Flujo Alterno	Si el correo ya está registrado: 1. El sistema informa que el correo ya está en uso.
Resultado	El usuario queda registrado con su tipo (comprador o dueño de restaurante).

Diagrama de cases de uso




address
id bigint NN
city varchar(255
country)
full_name varchar(255
postal_code)
state varchar(255
street_address)
varchar(255
)
varchar(255
)
cart
customer_id bigint
id bigint NN
total bigint
cart_item
quantity int NN
cart_id bigin
food_id t
id bigint NN
total_price bigint
ingredients varbinary(255)
category
id bigint NN
restaurant_id bigint
name varchar(255)
coupon
discount_amount double NN
id bigint NN
validity_period datetime(6)
code varchar(255)
terms_and_conditions varchar(255)
events_seq
next_val bigint
food_images
food_id bigint NN
images varchar(1000)
food_ingredients
food_id bigin NN
ingredients_id t NN
bigin
t
ingredient_category_seq
next_val bigint
food
available bit(1) NN
is_seasonal bit(1) NN
is_vegetarian bit(1) NN
creation_date datetime(6)
food_category_id bigint
id bigint NN
price bigint
restaurant_id bigint
description varchar(255)
name varchar(255)
ingredients_item_seq
next_val bigint
events
id bigint NN
restaurant_id bigint
ends_at varchar(255)
image varchar(255)
location varchar(255)
name varchar(255)
started_at varchar(255)
orders
total_item int NN
total_price int NN
created_at datetime(6)
customer_id bigint
delivery_address_id bigint
id bigint NN
payment_id bigint
restaurant_id bigint
total_amount bigint
order_status varchar(255)
password_reset_token
id int NN
expiry_date datetime(6)
user_id bigint
token varchar(255)
notification
read_status bit(1) NN
id bigint NN
recipient_id bigint
restaurant_id bigint
sent_at datetime(6)
message varchar(255)
ingredient_category
id bigint NN
restaurant_id bigint
name varchar(255)
ingredients_item
in_stoke bit(1) NN
category_id bigint
id bigint NN
restaurant_id bigint
name varchar(255)
user_favorites
id user_id bigint
images bigint NN
description varbinary(1000)
title varchar(255)
varchar(255)
payment
total_amount double NN
created_at datetime(6)
id bigint NN
order_id bigint
payment_method varchar(255)
payment_status varchar(255)
order_item
quantity int NN
food_id bigint
id bigint NN
total_price bigint
ingredients varbinary(255)
restaurant_images
restaurant_id bigint NN
images varchar(1000)
orders_items
items_id bigin NN
order_id t NN
bigin
t
users_seq
next_val bigint
users
role id tinyint
email bigint NN
full_nam varchar(255
e)
passwor varchar(255
d status)
varchar(255
)
varchar(255
)
restaurant
num_rating int NN
open bit(1) NN
address_id bigint
id bigint NN
owner_id bigint
registration_date datetime(6)
cuisine_type varchar(255)
description varchar(255)
email varchar(255)
instagram varchar(255)
mobile varchar(255)
name varchar(255)
opening_hours varchar(255)
twitter varchar(255)
review
rating double NN
created_at datetime(
customer_id 6) bigint
id bigint NN
restaurant_id bigint
message varchar(255)
users_addresses
addresses_id bigint NN
user_id bigint NN

Auditoría de Diseño de Base de Datos

Aspectos Positivos

1. **Organización por Categorías y Jerarquías**
    - La base de datos está bien estructurada con tablas relacionadas que reflejan jerarquías lógicas, como:
        - category → Categoriza los alimentos por restaurante.
        - ingredient_category → Agrupa los ingredientes por categorías específicas.
    - Esto facilita la búsqueda y clasificación de información relevante.
2. **Relaciones Definidas Claramente**
    - Las relaciones entre tablas están bien diseñadas con claves foráneas (FOREIGN KEY) y claves primarias (PRIMARY KEY) que aseguran la integridad referencial:
        - Ejemplo:
            - cart_item se relaciona con cart a través de cart_id y con food a través de food_id.
            - food está relacionado con category y restaurant, lo que permite identificar rápidamente el origen y clasificación de cada alimento.
        - Las relaciones bien definidas ayudan a prevenir datos huérfanos y redundantes.
3. **Soporte para Funcionalidades Específicas**
    - Tablas como coupon y notification añaden funcionalidades clave que enriquecen el sistema:
        - **coupon** permite gestionar descuentos con atributos como validity_period y terms_and_conditions.
        - **notification** facilita la comunicación con los usuarios, diferenciando entre mensajes leídos y no leídos mediante read_status.
4. **Escalabilidad**

- Las tablas de secuencias (events_seq, ingredient_category_seq, etc.) permiten una gestión eficiente de identificadores para futuros registros, preparando el sistema para escalar.

5. **Flexibilidad en las Categorías**
   - Tablas como food_ingredients permiten una relación de muchos a muchos entre food e ingredients_item, lo que ofrece flexibilidad para definir alimentos con múltiples ingredientes.
6. **Optimización del Rendimiento**
   - Índices en claves foráneas y columnas importantes, como en:
     - KEY FK1uobyhgl1wvgt1jpccia8xxs3 para cart_id en cart_item.
     - KEY FKm9xrxt95wwp1r2s7andom1l1c para restaurant_id en food.
   - Estos índices mejoran las consultas relacionadas con tablas clave.
7. **Atributos Específicos para Casos de Uso**
   - Tablas como food tienen atributos específicos como is_vegetarian, is_seasonal, y available, lo que facilita filtrar alimentos según preferencias y temporadas.
8. **Compatibilidad y Estandarización**
   - El uso de codificaciones modernas como utf8mb4 asegura soporte para caracteres especiales y emojis, mejorando la experiencia en idiomas diversos.

Relaciones Clave

1. **users ↔ cart**
   - Relación **uno a uno**, donde cada usuario tiene un carrito único mediante la clave customer_id.
     Ventaja: Facilita el seguimiento personalizado de los pedidos.
2. **cart ↔ cart_item ↔ food**
   - Relación **uno a muchos** entre cart y cart_item, y **uno a muchos** entre cart_item y food.
     Ventaja: Permite registrar múltiples artículos en un carrito mientras asocia cada uno con un alimento específico.

3. **restaurant ↔ category ↔ food**
    - Relación **uno a muchos** entre restaurant y category, y otra relación similar entre category y food.
      Ventaja: Permite identificar fácilmente los alimentos disponibles en un restaurante y su categorización.
4. **food ↔ food_ingredients ↔ ingredients_item**
    - Relación **muchos a muchos** que conecta alimentos con sus ingredientes.
      Ventaja: Flexibilidad para asignar múltiples ingredientes a un alimento y viceversa.
5. **restaurant ↔ events**
    - Relación **uno a muchos**, donde un restaurante puede organizar múltiples eventos.
      Ventaja: Posibilita gestionar eventos promocionales o culturales asociados a restaurantes específicos.
6. **notification ↔ users / restaurant**
    - Relación **uno a muchos** con usuarios y restaurantes, enviando notificaciones relevantes a cada grupo.
      Ventaja: Mejora la comunicación personalizada.

El diseño de esta base de datos destaca por su claridad, flexibilidad y orientación a funcionalidades clave. Las relaciones están bien definidas, lo que asegura integridad y facilidad para escalar en futuros desarrollos. Su estructura modular y atributos específicos permiten cubrir casos de uso complejos con eficiencia y precisión.