



Estudiantes

Jheinel Brown	20240017
Roniel Antonio Sabala Germán	20240212
Danuel Ezequiel Cuevas Tejeda	20240250

Materia

Base de Datos Avanzada

Profesor

Daniel Arturo Sánchez De Oleo

Fecha de elaboración

01 / 04 / 2025

Proyecto final.

Objetivo del Proyecto

Desarrollar un sistema de gestión geoespacial para la búsqueda y administración de puestos de comida, permitiendo a los usuarios localizar establecimientos cercanos de manera eficiente y precisa. Este sistema aprovechará una arquitectura híbrida que integra bases de datos relacionales, no relacionales y en memoria para optimizar el almacenamiento, procesamiento y acceso a la información.

La plataforma proporcionará una experiencia fluida al usuario mediante la consulta en tiempo real de información clave, como ubicaciones, horarios, promociones y valoraciones de los establecimientos.

Para ello, haremos uso de las siguientes tecnologías:

- **PostgreSQL** para gestionar datos estructurados y ejecutar consultas geoespaciales avanzadas a través de PostGIS.
- **MongoDB** para almacenar reportes dinámicos sobre estadísticas de establecimientos, visitas y platillos.
- **Redis** para optimizar la velocidad del sistema mediante el cacheo de datos frecuentemente consultados, como rankings de los locales más visitados y búsquedas recientes.

Estamos confiados en que este enfoque garantizará una arquitectura escalable, rápida y adaptable a las necesidades tanto de los usuarios como de los dueños de establecimientos, facilitando la toma de decisiones basada en datos y mejorando la accesibilidad a la información en cualquier momento.

Base de Datos Relacional

(implementada en PostgreSQL)

Nombre de la Base de Datos

ProyectoFinalDb

Descripción General

Esta base de datos está diseñada como una solución integral para gestionar y analizar información relacionada con usuarios, establecimientos gastronómicos (como restaurantes, cafeterías, heladerías, etc.), empleados, promociones, platillos, preferencias alimenticias, reseñas, y visitas de clientes. Además, se incorpora una dimensión geoespacial mediante el uso de PostGIS, lo cual permite realizar análisis de proximidad y visualización basada en ubicación geográfica.

Tablas

1. usuarios

Almacena la información de los usuarios registrados en el sistema.

Relaciones:

- Puede visitar múltiples establecimientos.
- Puede dejar reviews a establecimientos.
- Tiene preferencias alimenticias.



2. establecimientos

Representan restaurantes o negocios en los cuales los usuarios pueden realizar visitas.

Relaciones:

- Tiene múltiples empleados, promociones, platillos, visitas y reviews asociadas.

establecimientos	
id	INT
nombre	VARCHAR(100)
descripcion	TEXT
hora_apertura	TIME
hora_cierre	TIME
dias_laborables	SET(...)
telefono	VARCHAR(15)
categoria	VARCHAR(50)
rating	DECIMAL(3,2)
fecha_inauguracion	DATE
ubicacion	GEOMETRY
Indexes	

3. empleados

Personal que trabaja en un establecimiento, incluye información como nombre, puesto y fecha de contratación. Esto permite gestionar recursos humanos, identificar empleados con más antigüedad, o vincular personal con desempeño del local.

Relaciones:

- Le pertenece exclusivamente a un solo establecimiento.

empleados	
id	INT
nombre	VARCHAR(100)
puesto	VARCHAR(50)
sueldo	INT
fecha_contratacion	DATE
establecimiento_id	INT
Indexes	

4. promociones

Incluye ofertas y campañas activas o históricas asociadas a un establecimiento. Permite registrar promociones con fecha de inicio y expiración, útiles para estadísticas de marketing o para mostrar en una app a un cliente.

Relaciones:

- Le pertenece exclusivamente a un solo establecimiento.

promociones	
id	INT
oferta	TEXT
fecha_inicio	DATETIME
fecha_expiracion	DATETIME
establecimientos_id	INT
Indexes	

5. comidas

Define los diferentes tipos de platos o categorías alimenticias. Esta tabla actúa como base de clasificación para los platillos y para las preferencias de los usuarios.

Relaciones:

- Puede ser una preferencia para uno o mas usuarios.
- Puede servirse en uno o más establecimientos.

comidas	
id	INT
nombre	VARCHAR(100)
tipo_cocina	VARCHAR(50)
Indexes	

6. platillos

Incluye descripción, precio y disponibilidad. Se utiliza para componer menús y analizar ventas.

Relaciones:

- Un platillo es exclusivamente una comida.
- Un restaurante tiene muchos platillos.
- Un platillo puede ser consumido en un restaurante por muchos usuarios.

platillos	
id	INT
nombre	VARCHAR(100)
descripcion	TEXT
precio	DECIMAL(6,2)
esta_disponible	TINYINT
comida_id	INT
establecimiento_id	INT
Indexes	

7. preferencias

Registra los niveles de afinidad que un usuario tiene por ciertos tipos de comida. Esta tabla es esencial para construir sistemas de recomendación personalizados.

Relaciones:

- Le pertenece única y exclusivamente a un usuario y una comida.

preferencias	
usuario_id	INT
comida_id	INT
nivel_preferencia	TINYINT
Indexes	

8. reviews

Guarda las opiniones de los usuarios sobre su experiencia en un establecimiento. Incluye una calificación del 1 al 5, un comentario y la fecha. Permite evaluar la satisfacción del cliente, detectar problemas y construir rankings de calidad.

Relaciones:

- Es hecha exclusivamente por un usuario.
- Está dirigida exclusivamente hacia un restaurante en específico.

reviews	
id	INT
calificación	TINYINT
comentario	TEXT
fecha_publicacion	TIMESTAMP
usuario_id	INT
establecimiento_id	INT
Indexes	

9. visitas

Representa cada vez que un usuario acude a un establecimiento. Es clave para conocer la frecuencia de los usuarios y calcular indicadores como ticket promedio o visitas por semana.

Relaciones:

- En la visita se consumen múltiples platillos por un usuario en un establecimiento.

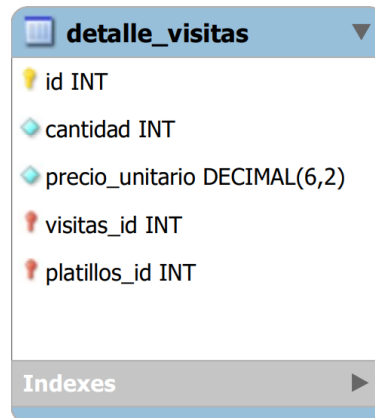
visitas	
id	INT
fecha_visita	DATETIME
consumo_total	DECIMAL(6,2)
usuario_id	INT
establecimiento_id	INT
Indexes	

10. detalle_visitas

Permite saber qué platillos se consumen más, qué combinaciones se hacen, y estimar el ingreso generado por platillo o categoría.

Relaciones:

- Asocia platillos específicos a visita, incluyendo cantidad y el precio unitario.



Indices

1. Índices comunes

Optimizan las consultas de las tablas SQL tradicionales.

- idx_reviews_calificacion
- idx_reviews_fecha
- idx_visitas_fecha

2. Índices geoespaciales

Optimizan consultas espaciales con PostGIS.

- idx_usuarios_ubicacion
- idx_establecimientos_ubicacion

Vistas

vista_establecimientos_rating

Muestra todos los establecimientos registrados con su promedio de calificación, calculado a partir de las reviews de los usuarios.

Campos:

- id: Identificador del establecimiento.
- nombre: Nombre del establecimiento.
- categoria: Categoría del establecimiento (ej. Cafetería, Pizzería).
- telefono: Número de contacto del establecimiento.
- ubicacion: Coordenadas geoespaciales del establecimiento.
- promedio_calificacion: Promedio de las calificaciones recibidas, o 0 si no tiene reviews.

vista_visitas_usuario

Presenta todas las visitas realizadas por usuarios a los distintos establecimientos.

Campos:

- id: Identificador de la visita.
- usuario: Nombre del usuario que realizó la visita.
- establecimiento: Nombre del establecimiento visitado.
- fecha_visita: Fecha y hora de la visita.
- consumo_total: Monto total consumido en la visita.
- Objetivo: Analizar el comportamiento de consumo y frecuencia de visitas de los usuarios.

vista_platillos_mas_vendidos

Lista los platillos con mayor número de unidades vendidas por establecimiento.

Campos:

- id: Identificador del platillo.
- nombre: Nombre del platillo.
- establecimiento: Nombre del establecimiento que ofrece el platillo.
- total_vendido: Número total de unidades vendidas del platillo.
- Objetivo: Identificar los productos más exitosos y populares por establecimiento.

vista_usuarios_preferencias

Detalla las preferencias alimenticias de los usuarios.

Campos:

- id: Identificador del usuario.
- nombre: Nombre del usuario.
- comida_preferida: Nombre del tipo de comida que prefiere.
- nivel_preferencia: Nivel de afinidad del usuario con esa comida (escala del 1 al 10).
- Objetivo: Ofrecer personalización en recomendaciones de menú y marketing según gustos de los usuarios.

establecimientos_mas_visitados

Muestra los establecimientos con mayor cantidad de visitas registradas.

Campos:

- id: Identificador del establecimiento.
- nombre: Nombre del establecimiento.
- total_visitas: Cantidad total de visitas que ha recibido.
- Objetivo: Detectar los locales más concurridos y evaluar su popularidad.

usuarios_mas_activos

Identifica a los usuarios más activos con base en la cantidad de visitas realizadas.

Campos:

- id: Identificador del usuario.
- nombre: Nombre del usuario.
- total_visitas: Total de visitas hechas por el usuario.
- Objetivo: Medir la fidelización de los usuarios y detectar clientes frecuentes para recompensas o análisis de comportamiento.

platillos_mas_vendidos

Lista los platillos más vendidos a nivel general (sin distinguir el establecimiento).

Campos:

- id: Identificador del platillo.
- nombre: Nombre del platillo.

- `total_vendido`: Suma total de unidades vendidas del platillo en todos los establecimientos.
- **Objetivo**: Obtener una visión general de los platillos más populares del sistema para análisis de demanda global.

Funciones

`recomendar_establecimientos(usuario_id INT)`

Recomienda establecimientos al usuario en base a sus preferencias alimenticias.

Parámetros:

- `usuario_id`: ID del usuario para el cual se generan las recomendaciones.

Retorna: Una tabla con el ID y nombre de establecimientos ordenados según el nivel de preferencia del usuario.

`calcular_distancia(usuario_id INT, establecimiento_id INT)`

Calcula la distancia geográfica entre un usuario y un establecimiento utilizando PostGIS.

Parámetros:

- `usuario_id`: ID del usuario.
- `establecimiento_id`: ID del establecimiento.

Retorna: Una distancia en metros (DOUBLE PRECISION).

`obtener_promedio_calificacion(est_id INT)`

Calcula el promedio de calificaciones que ha recibido un establecimiento.

Parámetros:

- `est_id`: ID del establecimiento.

Retorna: Promedio de calificaciones (DECIMAL(3,2)).

contar_visitas_usuario(user_id INT)

Cuenta la cantidad total de visitas registradas por un usuario.

Parámetros:

- user_id: ID del usuario.

Retorna: Número entero con la cantidad total de visitas.

obtener_gasto_total_usuario(user_id INT)

Calcula el total de dinero gastado por un usuario en todas sus visitas.

Parámetros:

- user_id: ID del usuario.

Retorna: Monto total gastado (DECIMAL(10,2)).

Procedimientos

agregar_usuario(nombre, email, telefono, fecha_registro, lat, lon)

Registra un nuevo usuario en el sistema, incluyendo su ubicación geoespacial.

Parámetros:

- nombre: Nombre del usuario.
- email: Correo electrónico único del usuario.
- telefono: Número de teléfono.
- fecha_registro: Fecha en la que se registra el usuario.
- lat: Latitud de la ubicación del usuario.
- lon: Longitud de la ubicación del usuario.

Retorna: No retorna ningún valor (VOID).

`registrar_visita(usuario_id, establecimiento_id, fecha, total)`

Inserta una nueva visita de un usuario a un establecimiento.

Parámetros:

- usuario_id: ID del usuario que realiza la visita.
- establecimiento_id: ID del establecimiento visitado.
- fecha: Fecha y hora de la visita.
- total: Monto total del consumo en esa visita.

Retorna: No retorna ningún valor (VOID).

`agregar_review(usuario_id, establecimiento_id, calificacion, comentario, fecha)`

Agrega una reseña o calificación hecha por un usuario sobre un establecimiento.

Parámetros:

- usuario_id: ID del usuario que hace la review.
- establecimiento_id: ID del establecimiento reseñado.
- calificacion: Calificación otorgada (1 a 5).
- comentario: Texto de la opinión del usuario.
- fecha: Fecha y hora de publicación de la reseña.

Retorna: No retorna ningún valor (VOID).

`actualizar_rating_establecimiento(est_id)`

Recalcula y actualiza el promedio de calificaciones de un establecimiento.

Parámetros:

- est_id: ID del establecimiento al que se desea actualizar el rating.

Retorna: No retorna ningún valor (VOID).

Triggers

trigger_validar_disponibilidad

Valida si el platillo seleccionado está disponible antes de insertar o actualizar un detalle de visita.

Evento: BEFORE INSERT OR UPDATE ON detalle_visitas

Función asociada: validar_disponibilidad_platillo()

trigger_actualizar_rating

Actualiza automáticamente el rating promedio del establecimiento cuando se inserta o modifica una review.

Evento: AFTER INSERT OR UPDATE ON reviews

Función asociada: actualizar_rating_trigger()

trigger_validar_fecha_visita

Impide que se registren visitas con fechas futuras.

Evento: BEFORE INSERT OR UPDATE ON visitas

Función asociada: validar_fecha_visita()

trigger_validar_fecha_promocion

Valida que la fecha de expiración de una promoción sea posterior a la fecha de inicio.

Evento: BEFORE INSERT OR UPDATE ON promociones

Función asociada: validar_fecha_promocion()

trigger_calcular_precio_total

Calcula automáticamente el precio total del platillo en base a su precio unitario y cantidad antes de insertar o actualizar en detalle_visitas.

Evento: BEFORE INSERT OR UPDATE ON detalle_visitas

Función asociada: calcular_precio_total()

Base de Datos No Relacional

(construida en MongoDB)

Nombre de la Base de Datos

ProyectoFinalDb

Descripción General

Esta base de datos almacena información generada a partir de la base de datos relacional en PostgreSQL, enfocándose en estadísticas y reportes que permiten acceder con más eficiencia y flexibilidad a ciertos datos. Esta también facilita el análisis de tendencias sin afectar el rendimiento de la base de datos relacional.

Colecciones

1. estadisticas_establecimientos

Esta colección almacena reportes mensuales de los establecimientos, incluyendo el número total de visitas y el consumo total de los clientes.

Estructura del documento:

- **establecimiento_id (Integer)**: Identificador del establecimiento.
- **mes (String)**: Mes y año en formato YYYY-MM.
- **visitas_totales (Integer)**: Cantidad total de visitas en ese mes.
- **consumo_total (Decimal)**: Suma del consumo total de los clientes en ese mes.
- **promedio_consumo (Decimal)**: Promedio del consumo total por cliente en ese mes.

ej.

```
{
  "establecimiento_id": 1,
  "mes": "2024-02",
  "visitas_totales": 1,
  "Consumo total": 21.99,
  "promedio_consumo": 21.99
},
```

2. estadisticas_reviews

Esta colección almacena estadísticas mensuales sobre las reseñas recibidas por cada establecimiento. Permite conocer la cantidad de reseñas, el promedio de calificación y la distribución de calificaciones.

Estructura del documento:

- establecimiento_id (**Integer**): Identificador del establecimiento.
- mes (**String**): Mes y año en formato YYYY-MM.
- total_reviews (**Integer**): Número total de reseñas en ese mes.
- promedio_calificacion (**Decimal**): Calificación promedio de las reseñas en ese mes.
- distribucion_calificaciones (**Objeto**): Contador de reseñas por calificación (1-5 estrellas).

ej.

```
{
  "establecimiento_id": 1,
  "mes": "2024-02",
  "total_reviews": 1,
  "promedio_calificacion": 3.0,
  "distribucion_calificaciones": {
    "1": 0,
    "2": 0,
    "3": 1,
    "4": 0,
    "5": 0
  }
},
```

3. platillos_estrellas

Esta colección guarda información sobre los platillos más vendidos en cada establecimiento durante un mes. Permite identificar qué productos son los más populares.

Estructura del documento:

- establecimiento_id (**Integer**): Identificador del establecimiento.
- mes (**String**): Mes y año en formato YYYY-MM.
- platillos (**Array**): Lista de platillos más vendidos.
 - platillo_id (**Integer**): Identificador del platillo.
 - nombre (**String**): Nombre del platillo.
 - cantidad (**Integer**): Número de veces vendido en ese mes.

ej.

```
{
  "establecimiento_id": 14,
  "mes": "2024-03",
  "platillos": [
    { "platillo_id": 6, "nombre": "Biryani", "cantidad": 1 },
    { "platillo_id": 17, "nombre": "Churrasco", "cantidad": 1 }
  ]
},
```

Base de Datos en Memoria

(construida en Redis)

1. Estructura de Datos en Redis

Redis se utiliza como una base de datos de tipo clave-valor con estructuras de datos avanzadas. Para este proyecto, se han implementado las siguientes estructuras:

1.1 Establecimientos por Hora de Cierre

Estructura: Conjunto ordenado (Sorted Set) **Clave:** establecimientos:por_hora_cierre

Valor: ID del establecimiento **Puntuación:** Hora de cierre convertida a minutos desde medianoche.



```
ZADD establecimientos:por_hora_cierre 1410 "1" # Burger Town - 23:30
```

Detalles del establecimiento:

Estructura: Hash **Clave:** establecimiento:{id} **Campos:** nombre, categoría, hora_apertura, hora_cierre, etc.



```
HSET establecimiento:1 nombre "Burger Town" categoria "Hamburguesería" hora_apertura "11:00" hora_cierre "23:30"
```

1.2 Promociones Activas

Estructura: Conjunto ordenado (Sorted Set) **Clave:** promociones:activas **Valor:** ID de la promoción **Puntuación:** Timestamp Unix de la fecha de expiración.



```
ZADD promociones:activas 1719792000 "1" # 2x1 en tacos al pastor los martes (30/06/2024)
```

Detalles de la promoción: Estructura: Hash **Clave:** promocion:{id} **Campos:** oferta, fecha_inicio, fecha_expiracion, establecimiento_id



```
HSET promocion:1 oferta "2x1 en tacos al pastor los martes" fecha_inicio "2024-03-01" fecha_expiracion "2024-06-30" establecimiento_id "1"
```

Promociones por establecimiento: Estructura: Conjunto (Set) **Clave:** establecimiento:{id}:promociones **Valor:** IDs de las promociones



```
SADD establecimiento:1:promociones "1"
```

1.3 Ranking de Establecimientos por Visitas

Estructura: Conjunto ordenado (Sorted Set) **Clave:** establecimientos:por_visitas **Valor:** ID del establecimiento **Puntuación:** Número total de visitas



```
ZADD establecimientos:por_visitas 42 "1" # Burger Town con 42 visitas
```

Visitas por mes: Estructura: Conjunto ordenado (Sorted Set) **Clave:** establecimientos:por_visitas:{YYYYMM} **Valor:** ID del establecimiento **Puntuación:** Número de visitas en ese mes



```
ZADD establecimientos:por_visitas:202404 18 "1" # 18 visitas en abril 2024
```

1.4 Platillos Más Consumidos por Establecimiento

Estructura: Conjunto ordenado (Sorted Set) **Clave:** establecimiento:{id}:platillos_por_consumo **Valor:** ID del platillo **Puntuación:** Número de veces consumido



```
ZADD establecimiento:1:platos_por_consumo 45 "1" # Hamburguesa Clásica consumida 45 veces
```

Detalles del platillo: Estructura: Hash **Clave:** platillo:{id} **Campos:** nombre, precio, descripcion



```
HSET platillo:1 nombre "Hamburguesa Clásica" precio "8.99" descripcion "Hamburguesa con carne, lechuga, tomate y cebolla"
```

2. Tipos de Consultas Implementadas

2.1 Establecimientos Abiertos Hasta Cierta Hora

Estas consultas permiten encontrar establecimientos que están abiertos hasta o después de una hora específica.

Ejemplo:



```
# Establecimientos abiertos hasta las 22:00 o más tarde  
ZRANGEBYSCORE establecimientos:por_hora_cierre 1320 +inf
```


Casos de uso:

- Usuarios buscando restaurantes abiertos tarde
- Filtrado de establecimientos por disponibilidad horaria
- Planificación de visitas en horarios específicos

2.2 Promociones Activas

Estas consultas permiten encontrar promociones que están actualmente vigentes o que expirarán después de una fecha específica.

Ejemplo:



```
# Promociones activas a partir del 1 de abril de 2024
ZRANGEBYSCORE promociones:activas 1712073600 +inf
```

Casos de uso:

- Mostrar promociones vigentes a los usuarios
- Notificar sobre promociones a punto de expirar
- Filtrar promociones por período de tiempo

2.3 Ranking de Establecimientos Más Visitados

Estas consultas permiten obtener los establecimientos más populares basados en el número de visitas.

Ejemplo:



```
# Top 5 establecimientos más visitados
ZREVRANGE establecimientos:por_visitas 0 4 WITHSCORES
```

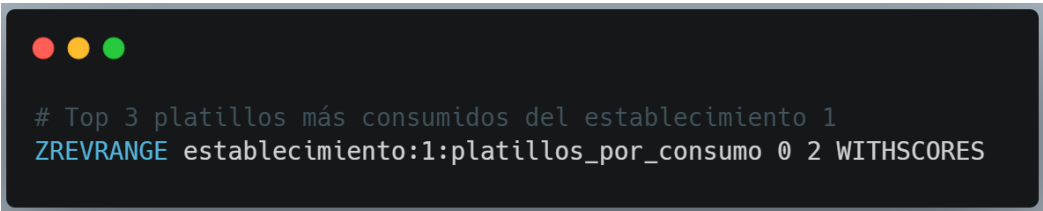
Casos de uso:

- Mostrar recomendaciones populares a los usuarios
- Análisis de tendencias de popularidad
- Identificación de establecimientos destacados

2.4 Platillos Más Consumidos por Establecimiento

Estas consultas permiten identificar los platillos más populares en cada establecimiento.

Ejemplo:



```
# Top 3 platillos más consumidos del establecimiento 1
ZREVRANGE establecimiento:1:platillos_por_consumo 0 2 WITHSCORES
```

Casos de uso:

- Recomendaciones de platillos a los usuarios
- Análisis de preferencias de consumo
- Optimización de inventario y preparación

3. Proceso de Sincronización de Datos

3.1 Extracción y Carga

El proceso de sincronización se realiza mediante un script Node.js que:

1. Conecta a PostgreSQL y extrae los datos relevantes.
2. Transforma los datos al formato optimizado para Redis.
3. Carga los datos en Redis utilizando las estructuras definidas.
4. Genera un archivo con todos los comandos Redis ejecutados para referencia.

3.2 Estrategias de Sincronización

Se pueden implementar diferentes estrategias de sincronización:

- **Sincronización periódica:** Ejecutar el script a intervalos regulares (por ejemplo, cada hora)
- **Sincronización basada en eventos:** Ejecutar el script cuando ocurren cambios significativos en PostgreSQL.
- **Sincronización incremental:** Actualizar solo los datos que han cambiado desde la última sincronización.

Diagrama de arquitectura

La arquitectura representada en el diagrama está diseñada para una aplicación que conecta a usuarios con dueños de establecimientos, permitiendo la interacción, consulta de información y generación de reportes.

Tanto los usuarios como los dueños de establecimientos acceden a la aplicación a través de Internet y sus solicitudes son procesadas por el servidor central, que se encarga de redirigir las operaciones hacia las bases de datos correspondientes. Su rol es crucial, ya que decide qué información se almacena, se consulta o se actualiza, y en qué tipo de base de datos debe realizarse cada operación.

PostgreSQL se encarga de almacenar la información estructurada del sistema, como:

- Datos de usuarios y dueños.
- Información de los establecimientos.
- Promociones y horarios.
- Relación entre visitas, ubicaciones y usuarios.

MongoDB se utiliza para almacenar datos no estructurados o semi-estructurados, como reportes de visitas a los establecimientos, estadísticas dinámicas, etc. Se mantiene actualizado gracias a una sincronización periódica con PostgreSQL.

Redis se utiliza para el cacheo de datos e información que se consulta con mucha frecuencia. Al guardar estos datos en memoria, Redis permite que el servidor los recupere de forma casi instantánea, mejorando así la velocidad de respuesta del sistema.