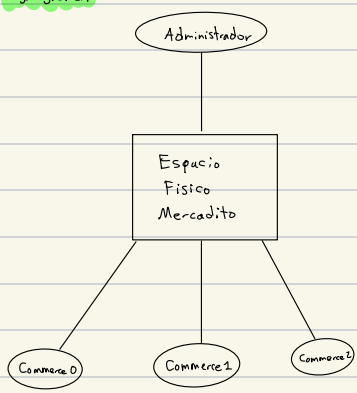
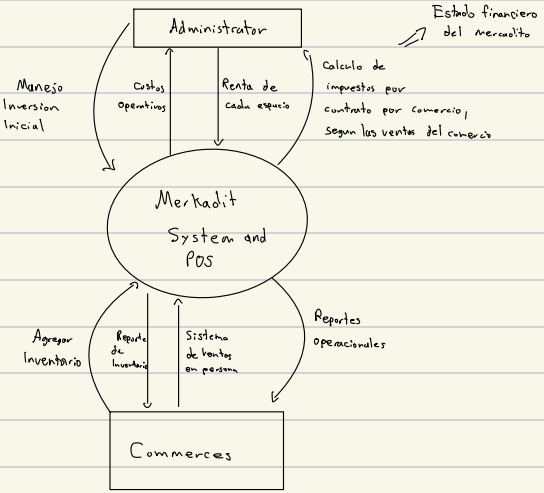
11/09/2025

Caso # 1 Bases de Datos

MerKadit = Sistema Interno de manejo de mercados gastronomicos y tiendas de conveniencia.

Organigrama:





Esto ayuda a definir los contratos o settlements por mes segun las ventas del negocio.

Nota: Todas las ventas y reportes del negocio independiente son vistas por el administrador para tomarlos en cuenta segun el contrato.

Requerimientos funcionales:

1. Manejo de Espacio y Contrato
2. Manejo financiero del administrador
3. Sistema de ventas (POS)
4. Fee and Revenue Calculation
5. Reportes y Monitoreo.
6. Roles de Usuario y control de acceso

Derivables

1. Investigar el dominio del problema
2. Diseño de la base de datos
3. Inserts en Queries
   - 2 Edificios => 2 Espacios (Alquiler) => 4 a 7 Comercios

4. Creación de un Rest API

Investigación:

- Que es una API? → Application Programing Interface → Reglas y herramientas que permiten que dos programas/sistemas se comuniquen entre sí "mensajero que lleva una solicitud al sistema y devuelve la respuesta sin tener que saber como funciona el sistema internamente"
- Que es un rest API? → Es un tipo de Api que usa HTTP y URLs claras para acceder, crear, modificar o eliminar datos de forma organizada
- Que es una tecnologia de API? Node Js + Express etc..
  ↳ Son herramientas, métodos o estilos que se utilizan para crear, usar y gestionar APIs

Architectural Layers:

- Handler Layer (API Routes/EndPoints): → Que son? Para que se utilizan? Como funcionan?
- Controller Layer (Business Logic Orchestration): → Que son? Para que se utilizan? Como funcionan?
- Service Layer (Complex Business Logic): → Que son? Para que se utilizan? Como funcionan?
- Repository Layer (Data Access): → Que son? Para que se utilizan? Como funcionan?

Stored Procedures → Que son? Para que se utilizan? Como funcionan?

- **Stored Procedure 1:** `registerSale`
  - Purpose: To record the sale of an item at a store.
  - Input Parameters must have at least:
    - Product name
    - Store name (`comercio`)
    - Quantity sold
    - Amount paid (`monto pagado`)
    - Payment method (`medio de pago`)
    - Payment confirmations (`confirmaciones de pago`)
    - Reference numbers (`numeros de referencia`)
    - Invoice number (`numero de factura`)
    - Customer (`cliente`)
    - Applied discounts (`descuentos aplicados`)
  - Implementation Requirements:
    - Must include comprehensive exception handling.
    - Must log details of the operation (e.g., computer, user, checksum).

- Esto como se conecta con mi BD? → Están dentro de la BD
- Los resultados de este procedure tienen un efecto en el diseño de mis tablas?

- **Stored Procedure 2:** `settlement`
  - Purpose: To settle the accounts for a store for the current month.
  - Input Parameters it must have at least:
    - Name of the store (`comercio`)
    - Name of the location/premises (`local`)
  - Logic it must perform:
    - Check if the store has already been settled for the current month.
    - If it has NOT been settled:
      - Calculate the total fees owed to the premises administrator based on all sales.
      - Calculate the remaining amount that belongs to the store.
      - Execute the respective financial transactions (e.g., adjusting balances).
      - Record the settlement to prevent it from being done a second time.
    - If it has been settled, it should handle this scenario appropriately.
  - Implementation Requirements:
    - Must include comprehensive exception handling and transaction management to ensure data integrity.
    - Must log details of the operation (e.g., computer, user, checksum).

Postman test → Que es un postman collection para testear la API?

5. Queries para reportes

5. Write SQL queries for reporting

- Create a query that generates a business report as of the current date.
- The report should include:
  - Business name
  - Store space name
  - Building name
  - Date of the first sale in the current month
  - Date of the last sale in the current month
  - Number of items sold
  - Total sales amount
  - Percentage and monetary amount due to the store space owner
  - Rental fee amount to be paid by the business
- Transform this query into a view and use it as data source for a report
- Create a professional-looking report that includes:
  - Title and subtitle
  - Current date
  - A table with the query results
  - Subtotals grouped by business
  - A final total at the end of the report
- You may use any of the following tools (or a similar one): Power BI, Crystal Reports, Tableau, Kibana.
- The report should demonstrate clear data visualization and professional presentation.

- Que es Power BI, como lo conecto con mi BD?

Diseño de la BD:

- Todas las tablas inician con la etiqueta "MK_" de Merkadit

- División de Tablas:

Perfiles

• Users ⟷ • User Roles ⟷ • Roles ⟷ • RolePermissions ⟷ • Permissions

Usuario x Mercado = Un mercado tiene varios user y un user puede administar varios mercados Caso empresas.

Espacio y Comercios

• Mercado (Un administrador puede tener varios mercados)

↕

• Building x Mercado (Multiplaza puede tener 2 mercados y 1 mismo mercado puede estar en varios malls)

↓

• Building (Ejemplo Multiplaza) => FK Addresses

↪

• Spaces (Un local que se alquila para el espacio del mercado)

↪

• Commerces (Comercios independientes que alquilan en el mercadito) n

• Commerce Category (Categoria del comercio del mercadito)

Localizaciones

• Countries • States • Cities • Address

Contratos con Comercios y Building

• Contract ↪ • Contract Per Commerce ↪ • Contract Per Building ↪ • Contract Renewals • Commerce Settlement • Commerce Settlement Detail

Logs

• Logs • Log Types • Log Sources • Log Stevenities

Handler Layer (API Routes / EndPoints)

Es la capa de la API que recibe y maneja las solicitudes que llegan desde el cliente. Cada EndPoint es una URL especifica que representa un recurso o una accion dentro de la API.

Sirven para → Definir puntos de entrada a tu API (URLs que el cliente puede usar)
→ Conectar la peticion del cliente con la lógica del negocio y la base de datos
→ Organizar el código en capas claras:
- Cliente → solicita algo
- EndPoint (handler) → recibe la solicitud
- Lógica del negocio → procesa
- Base de datos → devuelve los datos
- EndPoint → responde al cliente

} Asi funciona

Controller Layer (Business Logic Orchestration)

Es la capa de control dentro de la API que se encarga de gestionar la lógica del negocio. Recibe las solicitudes desde el Handler Layer y decide qué hacer.

Se utiliza para:

1. Orquestar la lógica del negocio: asegura que las reglas del sistema se cumplan
2. Separar responsabilidades: el handler solo recibe la solicitud, el controller decide qué hacer.
3. Reutilizar código: diferentes endpoints pueden usar la misma lógica del controller.

¿Cómo funciona?

1. El endpoint recibe la solicitud HTTP del cliente.
2. Llama a un controller específico para ese recurso o acción
3. El controller:
   - Valida la información recibida
   - Ejecuta la lógica de negocio (Service Layer)
   - Llama a la base de datos si es necesario
   - Devuelve los resultados al handler
4. El handler envía la respuesta al cliente.

Service Layer (Complex Business Logic)

Es la capa donde se maneja la lógica de negocio más compleja de la aplicación. El controller layer decide qué hacer con la solicitud, pero el Service Layer se encarga de cómo hacerlo: cálculos, reglas complejas, coordinación de varios recursos, integración con otros servicios, etc.

Se utiliza para:

1. Centralizar la lógica del negocio
2. Reutilizar procesos complejos
3. Mantener el código organizado

¿Cómo funciona?

1. El controller recibe la solicitud del handler
2. Llama a uno o varios servicios del Service Layer según lo que se necesite
3. El Service Layer:
    - Ejecuta la lógica del negocio
    - Interactúa con la base de datos o APIs externas
    - Aplica reglas del negocio
    - Devuelve el resultado al controller

Repository Layer (Data Access)

Es la capa de la API que se encarga de interactuar directamente con la base de datos. Su objetivo es aislar la lógica de negocio de los detalles de la base de datos.

Se utiliza para:

1. Separar responsabilidades
2. Reutilizar consultas y operaciones de datos
3. Facilitar cambios en la base de datos
4. Mantener código limpio y testeable

¿Cómo funciona?

1. El Service Layer necesita leer o guardar información
2. Llama a funciones del Repository Layer para acceder a la base de datos.
3. El Repository Layer ejecuta las consultas o llamadas al motor de datos.
4. Devuelve los resultados al service, que sigue con la lógica del negocio

Stored Procedures

Son programas o funciones que se guardan dentro de la base de datos. Contienen consultas SQL y lógica, la base de datos los almacena y ejecuta internamente.

Se utiliza para

1. Reutilizar código
2. Mejorar rendimiento
3. Seguridad (dar acceso a los procedimientos sin dar acceso a las tablas)
4. Centralizar lógica

¿Cómo funcionan?

1. Se crea un procedimiento en la base de datos con SQL
2. Se le pueden pasar parámetros de entrada y salida
3. Cuando se necesite, el usuario llama al procedimiento, y la base de datos lo procesa
4. Devuelve los resultados al llamador..