# Sistema Merkadit

Sistema de Gestión de Mercados Comerciales

Caso de Estudio #1 - Base de Datos I

---


**Presentado por:**

**Lindsay Nahome Marín Sánchez**

Código: **2024163904**

**Lee Sangcheol**

Código: **2024801079**

Profesor: **Rodrigo Núñez Núñez**


---

23 de septiembre de 2025


Resumen Ejecutivo

Merkadit es un sistema integral de gestión de mercados comerciales que optimiza la administración de espacios gastronómicos y de retail. El sistema permite a los administradores transformar espacios físicos en múltiples locales comerciales rentados por negocios independientes, integrando funcionalidad POS para inquilinos y control financiero centralizado. Este documento presenta la implementación completa incluyendo diseño de base de datos, API REST, procedimientos almacenados, análisis de datos y reporte visual profesional en Power BI.

# Índice

# 1. Introducción

## 1.1. Descripción del Sistema

Merkadit es una plataforma diseñada para optimizar la gestión de mercados gastronómicos y de retail, donde un administrador invierte en remodelar un espacio físico y lo transforma en múltiples locales o kioscos rentados por negocios independientes. La plataforma permite gestionar la inversión inicial, gastos operacionales continuos, la renta base de cada espacio, y el cálculo automático de comisiones basadas en ventas según el contrato de cada inquilino.

## 1.2. Objetivos del Proyecto

1. Diseñar una base de datos robusta que soporte la gestión completa de mercados comerciales
2. Implementar procedimientos almacenados para operaciones críticas de ventas y liquidaciones
3. Desarrollar una API REST con arquitectura de 4 capas
4. Crear consultas SQL avanzadas para reportes de negocio
5. Desarrollar un reporte visual profesional para análisis de datos
6. Demostrar el funcionamiento del sistema mediante pruebas exhaustivas

## 1.3. Funcionalidades Principales

- **Gestión de Espacios:** Registro y administración de espacios físicos disponibles
- **Control de Contratos:** Configuración de rentas base y porcentajes de comisión
- **Sistema POS Integrado:** Punto de venta para inquilinos con gestión de inventario
- **Liquidaciones Automáticas:** Cálculo de comisiones y rentas mensuales
- **Reportes Financieros:** Análisis de ingresos, gastos y flujo de caja
- **Control de Acceso:** Roles diferenciados para administradores e inquilinos
- **Dashboard Ejecutivo:** Visualizaciones interactivas en Power BI

# 2. Análisis y Diseño

## 2.1. Investigación del Dominio

La investigación se centró en identificar todas las entidades relevantes para el problema de Merkadit, incluyendo:

- **Gestión de Usuarios:** Individuos y empresas que administran mercados
- **Información de Mercados:** Direcciones, tamaños, tipos de locales disponibles

- **Contratos:** Negocios que rentan espacios y sus acuerdos comerciales
- **Inventarios:** Productos que los comerciantes venderán en sus espacios
- **Ventas:** Registro de transacciones a través del sistema POS
- **Aspectos Financieros:** Comisiones, pagos, liquidaciones y reportes

## 2.2. Diseño de la Base de Datos

El diseño de la base de datos se estructuró en módulos funcionales:

**Módulos del Sistema**

- **Sistema de Direcciones:** Countries, States, Cities, Addresses
- **Sistema de Usuarios:** Users, Roles, Permissions, Contacts
- **Sistema de Comercios:** Buildings, Floors, Spaces, Commerces
- **Sistema de Contratos:** Contracts, Schedules, Settlements
- **Sistema de Inventario:** Products, Categories, Price History, Inventory Movements
- **Sistema de Ventas:** Sales, Customers, Payment Methods, Sale Details
- **Sistema Financiero:** Investments, Expenses, Reports
- **Sistema de Auditoría:** Logs con checksums para integridad de datos

## 2.3. Retroalimentación Académica Incorporada

Tras la presentación del diagrama ERD inicial, se recibió retroalimentación del profesor que fue incorporada en el diseño final:

> *"Todo super, con esta revisión estamos más que bien, creo que podrían avanzar ya sin mayor problema con el resto del trabajo. Solo hay dos cositas que podrían tomar en cuenta..."*

### 2.3.1. 1. Comisiones Negociadas por Categoría de Producto

Se implementó la recomendación de permitir comisiones diferenciadas por categoría de producto. Como mencionó el profesor:

> *"Los fees que pagan los comercios podrían ser negociados por categoría de producto, por ejemplo como mieles, vinagres y eso de un precio bajo, al rato la comisión sería algo algo digamos no se un 7%, pero también podría ser que también venden vinos caros importados, de unos 40 mil colones, entonces ahí se negocie con el administrador que para la categoría de licores el fee sea 3% o como cuando hablamos de drones o equipo muy caro, la comisión debe ser más baja."*

**Implementación realizada:**

- **Productos básicos** (mieles, vinagres): 7 % de comisión
- **Licores importados** (vinos de ₡40,000): 3 % de comisión
- **Electrónicos** (drones, equipo caro): 2 % de comisión

La lógica de cálculo sigue un patrón de fallback inteligente:

1. Busca comisión específica negociada en `ContractCategoryCommissions`
2. Si no existe, usa la comisión por defecto de la categoría
3. Como último recurso, usa la comisión general del contrato

### 2.3.2. 2. Reportes Dinámicos vs Tablas Estáticas

Siguiendo la recomendación académica sobre reportes dinámicos:

> *"Los reportes, estos normalmente son calculados al vuelo, es decir, se usa una herramienta de reportería que haciendo la consulta calcula la información dinámicamente que va en el reporte, entonces no es necesario tener esas tablas de reportes, pues en una empresa pueden haber 10, 20, 80 reportes diferentes, para los cuales no se hace tabla."*

**Decisión de diseño:** Se optó por la **opción (b): crear una vista para cada reporte**, ya que:

- Los reportes se calculan dinámicamente al ejecutar las consultas
- Las empresas pueden requerir múltiples reportes diferentes sin crear tablas adicionales
- Las vistas SQL permiten consultas en tiempo real con datos actualizados
- Es compatible con herramientas de reporting profesionales como Power BI
- Mantiene la integridad y consistencia de los datos

**Vistas implementadas:**

- **ReportCommercesSales:** Ventas por comercio con comisiones dinámicas
- **ReportBuildingFinancials:** Finanzas consolidadas por edificio
- **ReportCategoryPerformance:** Performance por categoría de producto
- **BusinessReportView:** Vista principal para herramientas BI

## 2.4. Características Avanzadas del Diseño

# 3. Implementación

## 3.1. Base de Datos

### 3.1.1. Esquema Principal

La base de datos `merkadit` contiene más de 60 tablas organizadas de manera modular. Las tablas principales incluyen:

| Tabla | Propósito | Registros |
|---|---|---|
| Buildings | Edificios comerciales | 2 |
| Spaces | Espacios disponibles para renta | 3 |
| Commerces | Negocios históricos y activos | 15 |
| Contracts | Contratos activos | 5 |
| Products | Catálogo de productos | 14 |
| Sales | Transacciones de venta | 240 |
| Settlements | Liquidaciones mensuales | Variable |

### 3.1.2. Datos de Muestra

El script `sample_data.sql` genera datos realistas que incluyen:

- 2 edificios comerciales con ubicaciones en Costa Rica
- 4 meses de ventas aleatorias (mayo-agosto 2024)
- Entre 50-70 ventas por mes por comercio activo
- Inventario con productos de diferentes categorías
- Contratos con diferentes estructuras de comisión

## 3.2. Procedimientos Almacenados

### 3.2.1. SP_registerSale

Este procedimiento maneja el registro completo de ventas con las siguientes características:

**Funcionalidades de SP_registerSale**

- Validación completa de datos antes de la transacción
- Verificación de stock disponible para todos los productos
- Soporte para múltiples productos en una sola venta
- Actualización automática de inventario
- Creación de movimientos de inventario para auditoría
- Generación automática de número de factura
- Manejo de transacciones con rollback automático en errores
- Logging completo de éxitos y errores con checksums

**Parámetros de entrada:**

- `commerceID`: Identificador del comercio
- `customerID`: Identificador del cliente
- `paymentMethodID`: Método de pago utilizado
- `productID1-3, quantity1-3, unitPrice1-3`: Hasta 3 productos por venta
- `discountAmount`: Descuentos aplicados
- `cashierName`: Nombre del cajero

**Parámetros de salida:**

- `saleID`: ID de la venta creada
- `totalAmount`: Monto total de la venta
- `result`: Mensaje de resultado con detalles de la factura

### 3.2.2. SP_settleCommerce

Este procedimiento gestiona las liquidaciones mensuales con lógica sofisticada:

Funcionalidades de SP_settleCommerce

- Cálculo dinámico de comisiones por categoría de producto
- Validación de períodos y contratos activos
- Prevención de liquidaciones duplicadas
- Cálculo de renta base más comisiones variables
- Creación de registros de liquidación detallados
- Transacciones seguras con manejo de errores
- Logging detallado para auditoría financiera

**Parámetros de entrada:**

- `commerceID`: Comercio a liquidar
- `month, year`: Período de liquidación
- `processedBy`: Usuario que procesa la liquidación

**Parámetros de salida:**

- `settlementID`: ID de la liquidación creada
- `totalAmount`: Monto total a pagar
- `result`: Resumen detallado con ventas, comisión y renta

## 3.3. API REST

### 3.3.1. Arquitectura de 4 Capas

La API implementa una arquitectura clara y mantenible:

1. **Handler Layer (Endpoints HTTP):** Manejo de peticiones y respuestas HTTP
2. **Controller Layer (Orquestación):** Coordinación de lógica de negocio
3. **Service Layer (Lógica de Negocio):** Validaciones y reglas de negocio
4. **Repository Layer (Acceso a Datos):** Llamadas a procedimientos almacenados

### 3.3.2. Endpoints Implementados

| Método | Endpoint | Funcionalidad |
|---|---|---|
| GET | /health | Verificación de estado de la API |
| POST | /api/sales/register | Registro de ventas (llama SP_registerSale) |

| | | |
|---|---|---|
| GET | /api/sales/:saleID | Obtener detalles de una venta específica |
| POST | /api/settlements/process | Procesar liquidación (llama SP_settleCommerce) |
| GET | /api/settlements/commerce/:id | Obtener liquidaciones de un comercio |

#### 3.3.3. Tecnologías Utilizadas

- **Node.js + Express:** Framework web y servidor
- **MySQL2:** Driver de base de datos con soporte para procedimientos almacenados
- **Arquitectura RESTful:** Principios REST para API escalable
- **JSON:** Formato de intercambio de datos
- **Error Handling:** Manejo comprehensivo de errores y transacciones

### 3.4. Reporte Visual Profesional

#### 3.4.1. Implementación en Power BI

Se desarrolló un reporte interactivo en Power BI Desktop que utiliza la vista Business-ReportView como fuente de datos. El reporte incluye:

- **Título profesional:** "Merkadit Business Report - September 2025"
- **Tabla detallada:** Datos completos de todos los comercios activos con business_name, space_name, building_name, rental_fee_amount y total_amount_due
- **Gráfico comparativo:** Visualización de rentas por edificio comercial
- **Análisis de diferencias:** Comparación entre Mercado Central San José (₡1,000,000) vs Plaza Comercial Cartago (₡900,000)
- **Formato exportable:** Capacidad de exportar a PDF para distribución ejecutiva

#### 3.4.2. Insights del Reporte

El reporte demuestra la situación actual del negocio para septiembre 2025:

- Los comercios no han registrado ventas durante el período actual
- Las liquidaciones se basan únicamente en renta base acordada contractualmente
- El sistema está preparado para calcular comisiones cuando inicien las ventas
- La diferenciación de precios por ubicación refleja estrategia comercial efectiva

# 4. Consultas SQL Avanzadas

El proyecto incluye 6 consultas SQL que demuestran el uso de técnicas avanzadas requeridas:

## 4.1. Técnicas Implementadas

**Técnicas SQL Demostradas**

- **ORDER BY:** Ordenamiento de resultados por múltiples criterios
- **LIMIT (TOP):** Limitación de resultados para obtener top performers
- **Nested Queries:** Subconsultas complejas para cálculos dinámicos
- **EXISTS:** Verificación de existencia de registros relacionados
- **IN:** Filtrado por conjuntos de valores
- **Calculated Fields:** Campos calculados dinámicamente

## 4.2. Consultas Principales

### 4.2.1. Query 1: Reporte de Negocio Principal

Esta consulta cumple con los requisitos específicos del deliverable, proporcionando:

- Nombre del negocio, espacio y edificio
- Fechas de primera y última venta del mes actual
- Número de artículos vendidos
- Monto total de ventas
- Porcentaje y monto de comisión debido al propietario del espacio
- Monto de renta a pagar por el negocio

### 4.2.2. Query 2: TOP Productos por Volumen de Ventas

Utiliza LIMIT y ORDER BY para identificar los 15 productos más exitosos, incluyendo:

- Cantidad total vendida y revenue generado
- Precio promedio de venta y número de transacciones
- Revenue por venta como métrica de eficiencia

#### 4.2.3. Query 3: Comercios con Ventas de Alto Valor

Emplea EXISTS y subconsultas para identificar comercios con ventas superiores a ₡50,000:

- Valor promedio de venta por comercio
- Conteo de ventas de alto valor
- Análisis de performance de comercios premium

#### 4.2.4. Query 4: Análisis de Performance por Categoría

Utiliza IN con subconsultas para analizar categorías de productos:

- Productos totales y comercios activos por categoría
- Volumen de ventas y revenue total
- Comisiones generadas por categoría

#### 4.2.5. Query 5: Análisis de Liquidaciones

Subconsultas complejas para evaluar tendencias financieras:

- Meses con liquidaciones y revenue promedio mensual
- Tendencias de performance (mejorando/declinando/estable)
- Análisis comparativo entre períodos

#### 4.2.6. Query 6: Análisis de Inventario

Combina múltiples técnicas para gestión de inventario:

- Estado de stock (bajo/normal/sobrestock)
- Días estimados de inventario restante
- Valor monetario del stock actual

### 4.3. Vista BusinessReportView

Se creó una vista optimizada que transforma la Query 1 para herramientas de reportería como Power BI o Tableau, mejorando el performance mediante subconsultas optimizadas.

# 5. Pruebas y Validación

## 5.1. Colección de Postman

Se desarrolló una colección completa de Postman para probar todos los endpoints:

- **Health Check:** Verificación de estado de la API
- **Register Sale - Complete:** Venta con múltiples productos
- **Register Sale - Single:** Venta con un producto y descuento
- **Get Sale Details:** Consulta de detalles de venta
- **Process Settlement:** Procesamiento de liquidaciones
- **Get Commerce Settlements:** Consulta de historial de liquidaciones

## 5.2. Resultados de Pruebas

### 5.2.1. Pruebas de Registro de Ventas

**Resultados Exitosos - Register Sale**

- **Status:** 201 Created
- **Sale ID:** 243
- **Total Amount:** ₡3,390.00
- **Invoice:** FAC-2025-09-001-6486
- **Inventory Update:** Automático y correcto

### 5.2.2. Pruebas de Liquidaciones

**Resultados Exitosos - Process Settlement**

- **Status:** 201 Created
- **Settlement ID:** 17
- **Total Amount:** ₡500,000.00
- **Breakdown:** Ventas: ₡0.00, Comisión: ₡0.00, Renta: ₡500,000.00
- **Business Logic:** Correcto para período sin ventas

### 5.2.3. Pruebas de Consultas y Reportes

- **Get Sale Details:** Respuesta 200 OK con datos completos
- **Get Commerce Settlements:** Historial completo de liquidaciones
- **Queries SQL:** Todas las consultas ejecutan sin errores

- **BusinessReportView:** Vista creada exitosamente
- **Power BI Report:** Reporte funcional con visualizaciones correctas

# 6. Arquitectura y Tecnologías

## 6.1. Stack Tecnológico

- **Base de Datos:** MySQL 8.0
- **Modelado:** MySQL Workbench
- **Backend API:** Node.js + Express
- **Testing:** Postman
- **Business Intelligence:** Power BI Desktop
- **Control de Versiones:** Git + GitHub
- **Documentación:** LaTeX + Overleaf

## 6.2. Patrones de Diseño Aplicados

- **Repository Pattern:** Abstracción de acceso a datos
- **Service Layer Pattern:** Encapsulación de lógica de negocio
- **MVC Pattern:** Separación de responsabilidades
- **Transaction Script:** Para operaciones de base de datos complejas

## 6.3. Principios SOLID Aplicados

- **Single Responsibility:** Cada clase tiene una responsabilidad específica
- **Open/Closed:** Código abierto para extensión, cerrado para modificación
- **Dependency Inversion:** Dependencia de abstracciones, no de concreciones

# 7. Conclusiones

## 7.1. Objetivos Cumplidos

El proyecto Merkadit ha cumplido exitosamente todos los deliverables requeridos:

1. **Investigación y Diseño:** Base de datos robusta con más de 60 tablas
2. **Datos de Muestra:** Script completo con 4 meses de datos realistas
3. **Procedimientos Almacenados:** SP_registerSale y SP_settleCommerce funcionando

4. **API REST:** Arquitectura 4 capas completamente funcional
5. **Consultas SQL:** 6 queries avanzadas con todas las técnicas requeridas
6. **Reporte Visual:** Dashboard profesional en Power BI completamente funcional

## 7.2. Características Destacadas

- **Comisiones Dinámicas:** Sistema sofisticado de comisiones por categoría
- **Transacciones Seguras:** Manejo completo de errores y rollbacks
- **Auditoría Completa:** Logging con checksums para integridad
- **Escalabilidad:** Arquitectura preparada para crecimiento
- **Reportes Dinámicos:** Vistas SQL optimizadas para herramientas BI
- **Visualización Profesional:** Dashboard ejecutivo en Power BI

## 7.3. Lecciones Aprendidas

- La importancia de validar datos antes de operaciones críticas
- El valor de las transacciones para mantener consistencia
- La efectividad de las vistas SQL para reportes dinámicos
- La necesidad de logging completo para auditoría
- La importancia de pruebas exhaustivas en APIs
- El poder de las herramientas de visualización para insights de negocio

## 7.4. Trabajo Futuro

- Implementación de interfaz web para usuarios finales
- Integración con sistemas de pagos externos
- Módulo de analytics avanzado con machine learning
- Aplicación móvil para comerciantes
- Sistema de notificaciones en tiempo real
- Dashboards interactivos en tiempo real con Power BI Service

# 8. Anexos

## 8.1. Anexo A: Estructura de Archivos del Proyecto

```
Caso-1-Merkadit/
 README.md
 EntityDocumentation.docx
 EntityDocumentationv2.docx
 diagrama_merkadit_v1.pdf
 merkadit_schema_v1.sql
 merkadit_v1.mwb
 sample_data.sql
 stored_procedures.sql
 advanced_sql_queries.sql
 Merkadit_API_Collection.json
 Merkadit_Business_Report.pbix
 merkadit-api/
     package.json
     .env
     src/
         app.js
         config/database.js
         handlers/
         controllers/
         services/
         repositories/
```

## 8.2. Anexo B: Instrucciones de Instalación

### 8.2.1. Prerrequisitos

- MySQL 8.0 o superior
- Node.js 18.0 o superior
- npm 8.0 o superior
- Power BI Desktop (para visualización de reportes)

### 8.2.2. Instalación de Base de Datos

```
1. mysql -u root -p
2. SOURCE merkadit_schema_v1.sql;
3. SOURCE sample_data.sql;
4. SOURCE stored_procedures.sql;
5. SOURCE advanced_sql_queries.sql;
```

### 8.2.3. Instalación de API

```
1. cd merkadit-api
2. npm install
```

```
3. Configurar .env con credenciales de MySQL
4. npm run dev
```

#### 8.2.4. Visualización de Reportes

```
1. Abrir Power BI Desktop
2. Cargar Merkadit_Business_Report.pbix
3. Actualizar conexión de datos si es necesario
4. Explorar visualizaciones interactivas
```

### 8.3. Anexo C: URLs de Recursos

- **GitHub Repository:** https://github.com/CholiRat/Caso-1-Merkadit
- **API Health Check:** http://localhost:3000/health
- **Postman Collection:** Merkadit_API_Collection.json
- **MySQL Workbench Model:** merkadit_v1.mwb
- **Power BI Report:** Merkadit_Business_Report.pbix