Logotipo

El contenido generado por IA puede ser incorrecto.

**PROYECTO DE BASE DE DATOS:  
PLATAFORMA DE BARBERÍAS DEL MUNICIPIO CERCADO - COCHABAMBA**

**Integrantes:**

Enrique Lujan Arispe

William Lujan Arispe

**Fecha:**

12/09/2025

# ÍNDICE GENERAL

1. INTRODUCCIÓN

    1.1. Antecedentes

    1.2. Planteamiento del problema

    1.3. Objetivos

        1.3.1. Objetivo general

        1.3.2. Objetivos específicos

2. MARCO CONCEPTUAL

    2.1. Introducción

    2.2. Base de Datos Relacionales

    2.3. Modelo de Datos

    2.4. Lenguaje SQL

        2.4.1. DDL (Data Definition Language)

        2.4.2. DML (Data Manipulation Language)

        2.4.3. DQL (Data Query Language)

        2.4.4. DCL y TCL

    2.5. Seguridad en Bases de Datos

        2.5.1. Principios de Hardening

        2.5.2. Usuarios y Privilegios

        2.5.3. Roles y Buenas Prácticas

    2.6. Elementos Avanzados de Bases de Datos

        2.6.1. Vistas

        2.6.2. Procedimientos Almacenados

        2.6.3. Triggers

    2.7. Backups

    2.8. Migración

        2.8.1. Big Bang

        2.8.2. Incremental

3. MARCO PRÁCTICO

    3.1. Diseño de la Base de Datos

    3.2. Implementación de la Base de Datos

    3.3. Objetos de la Base de Datos

    3.4. Planificación de Migraciones

4. CONCLUSIONES Y RECOMENDACIONES

5. BIBLIOGRAFÍA

ANEXOS

# ÍNDICE DE TABLAS

[No hay tablas en este documento]

# ÍNDICE DE ILUSTRACIONES

Figura 1: Árbol de Problemas

# 1. INTRODUCCIÓN

## 1.1. Antecedentes

El presente proyecto surge como un emprendimiento propio orientado a resolver una problemática identificada en el sector de servicios de barbería en el municipio Cercado de la ciudad de Cochabamba, Bolivia.

Actualmente, el mercado de barberías en la zona opera de manera tradicional y desorganizada. Un estudio preliminar realizado con 4 barberías locales (1 ubicada en el centro de la ciudad y 3 en zonas periféricas) reveló un interés unánime en adoptar una solución tecnológica que modernice sus operaciones. Estas barberías representan una muestra inicial del mercado potencial, evidenciando que:

* **No existe competencia directa:** No hay plataformas digitales especializadas en el sector de barberías en Cochabamba.
* **Gestión rudimentaria:** La totalidad de los establecimientos consultados gestionan sus citas y comunicación con clientes a través de WhatsApp o Messenger, sin un sistema formal de organización.
* **Demanda latente:** Los clientes buscan servicios principalmente a través de redes sociales, recomendaciones personales y búsquedas en Google, sin una fuente centralizada de información.
* **Oportunidad de mercado:** Existe una clara necesidad de digitalización y profesionalización del sector.

## 1.2. Planteamiento del problema

## 

## 1.3. Objetivos

### 1.3.1. Objetivo general

Implementar una base de datos relacional que permita la gestión eficiente de servicios de barbería, reservas, usuarios y promociones, garantizando la integridad, seguridad y facilidad de acceso a los usuarios del sistema, facilitando la conexión entre barberos y clientes en el municipio Cercado de Cochabamba.

### 1.3.2. Objetivos específicos

* Realizar el modelado de la base de datos mediante diagramas entidad-relación (ER) y modelo relacional normalizado.
* Implementar la Base de Datos en un SGBD adecuado para el proyecto considerando escalabilidad y compatibilidad.
* Implementar los objetos necesarios en la Base de Datos incluyendo tablas, índices, vistas, procedimientos almacenados, triggers y funciones.
* Planificar la migración de datos estableciendo protocolos de carga inicial y actualización de información.
* Diseñar mecanismos de seguridad con control de acceso basado en roles y encriptación de datos sensibles.
* Establecer indicadores de rendimiento para monitorear y optimizar el funcionamiento del sistema.

1

# 2. MARCO CONCEPTUAL

## 2.1. Introducción

Según Silberschatz, Korth y Sudarshan (2014) en su libro "Fundamentos de Bases de Datos", "un sistema de gestión de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos" (p. 1). Esta definición establece la base fundamental para comprender la importancia de implementar un sistema robusto que gestione la información de las barberías en Cochabamba.

## 2.2. Base de Datos Relacionales

Elmasri y Navathe (2017) definen en "Fundamentos de Sistemas de Bases de Datos" que "el modelo relacional representa la base de datos como una colección de relaciones. Una relación se asemeja a una tabla de valores" (p. 63). Asimismo, Date (2012) señala que "el modelo relacional se ocupa de tres aspectos de los datos: estructura de datos, integridad de datos y manipulación de datos" (p. 127).

Para este proyecto, Coronel, Morris y Rob (2011) en "Bases de Datos: Diseño, Implementación y Administración" establecen que "una base de datos relacional almacena tanto los datos como las relaciones entre ellos en forma de tablas bidimensionales" (p. 31), lo cual resulta ideal para gestionar las relaciones entre barberías, servicios, clientes y reservas.

## 2.3. Modelo de Datos

Según Marqués (2011) en "Bases de Datos", "un modelo de datos es un conjunto de conceptos que permiten describir la estructura de una base de datos, es decir, los datos, las relaciones entre los datos y las restricciones" (p. 15). El proceso de modelado, como indica Piattini et al. (2006), "permite obtener una visión abstracta del mundo real, eliminando detalles superfluos y resaltando aquellos relevantes al problema" (p. 89).

Para el diseño conceptual, Batini, Ceri y Navathe (1994) establecen que "el objetivo principal es producir un esquema conceptual que represente correctamente todos los requisitos de datos de los usuarios" (p. 43).

## 2.4. Lenguaje SQL

Beaulieu (2009) en "SQL Práctico" define SQL como "el lenguaje estándar para trabajar con bases de datos relacionales [...] permite crear estructuras de bases de datos, añadir datos, modificarlos y consultarlos" (p. 3).

### 2.4.1. DDL (Data Definition Language)

Según Groff, Weinberg y Oppel (2010), "DDL incluye comandos para definir, alterar y eliminar tablas y otros objetos de base de datos como CREATE TABLE, ALTER TABLE y DROP TABLE" (p. 371). Celma, Casamayor y Mota (2003) añaden que "estas sentencias permiten definir el esquema de la base de datos" (p. 145).

### 2.4.2. DML (Data Manipulation Language)

Date (2012) explica que "DML es el subconjunto de SQL utilizado para insertar, actualizar, eliminar y consultar los datos: INSERT, UPDATE, DELETE y SELECT" (p. 89). Pérez (2014) complementa indicando que "estas operaciones constituyen las operaciones básicas CRUD (Create, Read, Update, Delete)" (p. 201).

### 2.4.3. DQL (Data Query Language)

Ramez y Shamkant (2017) destacan que "SELECT es la sentencia más poderosa del SQL, permitiendo realizar consultas simples y complejas mediante proyección, selección y combinación de tablas" (p. 187). Como señala Connolly y Begg (2015), "la cláusula SELECT especifica las columnas a recuperar, FROM indica las tablas y WHERE establece las condiciones" (p. 142).

### 2.4.4. DCL y TCL

Según Mullins (2012), "DCL controla los aspectos de autorización mediante GRANT y REVOKE, determinando qué usuarios pueden realizar qué operaciones" (p. 456). Respecto a TCL, Garcia-Molina et al. (2009) indican que "las transacciones garantizan la consistencia mediante COMMIT para confirmar cambios y ROLLBACK para deshacerlos" (p. 315).

## 2.5. Seguridad en Bases de Datos

### 2.5.1. Principios de Hardening

Litchfield et al. (2005) en "Manual de Seguridad en Bases de Datos" establecen que "el hardening implica configurar el sistema para minimizar vulnerabilidades: deshabilitar servicios innecesarios, aplicar parches y configurar auditorías" (p. 78). Natan (2005) refuerza que "la seguridad debe implementarse en capas: red, sistema operativo, SGBD y aplicación" (p. 112).

### 2.5.2. Usuarios y Privilegios

Fernández, Piattini y Peso (2008) señalan que "el principio de menor privilegio dicta que cada usuario debe tener únicamente los permisos mínimos necesarios para realizar su trabajo" (p. 234). Como indica Knox (2004), "nunca se debe utilizar la cuenta de administrador para operaciones rutinarias" (p. 89).

### 2.5.3. Roles y Buenas Prácticas

Según Gertz y Jajodia (2008), "los roles agrupan privilegios relacionados y simplifican la administración de permisos" (p. 167). Las buenas prácticas incluyen, según Ben-Natan y Saraph (2005): "contraseñas fuertes, auditoría continua, encriptación de datos sensibles y validación de entrada para prevenir inyección SQL" (p. 203).

## 2.6. Elementos Avanzados de Bases de Datos

### 2.6.1. Vistas

Según Ramakrishnan y Gehrke (2003) en "Sistemas de Gestión de Bases de Datos", "una vista es una tabla virtual cuyo contenido está definido por una consulta. Una vista no existe como conjunto de valores almacenados, sino como tabla derivada" (p. 98). Silberschatz et al. (2014) añaden que "las vistas proporcionan un mecanismo para ocultar ciertos datos a algunos usuarios y simplificar consultas complejas" (p. 137).

### 2.6.2. Procedimientos Almacenados

Celma et al. (2003) definen que "un procedimiento almacenado es un programa que se almacena físicamente en la base de datos y que generalmente está escrito en un lenguaje propietario del SGBD" (p. 423). Como indica Oppel (2009), "los procedimientos almacenados mejoran el rendimiento al reducir el tráfico de red y permiten encapsular la lógica del negocio" (p. 289).

### 2.6.3. Triggers

Date (2012) explica que "un trigger o disparador es un procedimiento que se ejecuta automáticamente cuando ocurre un evento específico en la base de datos" (p. 367). Según Coronel et al. (2011), "los triggers garantizan que ciertas condiciones se cumplan automáticamente cuando se modifican los datos, manteniendo la integridad referencial compleja" (p. 448).

## 2.7. Backups

Mullins (2012) establece que "el backup es la piedra angular de cualquier estrategia de recuperación de desastres. Sin backups adecuados, es imposible recuperar datos perdidos" (p. 589). Según Alapati (2008) en "Expert Oracle Database 11g Administration", "existen tres tipos principales de backup: completo, incremental y diferencial, cada uno con sus ventajas en términos de tiempo y espacio" (p. 734).

Piattini et al. (2006) recomiendan que "la política de copias de seguridad debe definir claramente la frecuencia, el tipo de copia, el medio de almacenamiento y el lugar de custodia" (p. 512).

## 2.8. Migración

Morris (2012) en "Practical Data Migration" define que "la migración de datos es el proceso de transferir datos entre sistemas, formatos de almacenamiento o bases de datos" (p. 15). Hudicka (2014) añade que "una migración exitosa requiere planificación detallada, mapeo de datos y pruebas exhaustivas" (p. 67).

### 2.8.1. Big Bang

Según Bisbal et al. (1999), "la migración Big Bang implica completar toda la migración en una ventana de tiempo muy limitada, generalmente durante un fin de semana o periodo de inactividad" (p. 103). Como señala Morris (2012), "aunque es más rápida, esta estrategia conlleva mayor riesgo porque no hay marcha atrás una vez iniciado el proceso" (p. 89).

### 2.8.2. Incremental

Wu et al. (1997) definen la migración incremental como "un proceso por fases donde los datos se migran en subconjuntos manejables a lo largo del tiempo" (p. 234). Brodie y Stonebraker (1995) indican que "la migración incremental reduce el riesgo al permitir la coexistencia de sistemas antiguos y nuevos durante el periodo de transición" (p. 156).

**2.9. Sistemas de Reservas y Gestión de Citas**

Según Kendall y Kendall (2011) en "Análisis y Diseño de Sistemas", "los sistemas de reservas en línea representan una evolución natural de los procesos manuales de programación de citas, permitiendo optimizar el uso del tiempo tanto de proveedores como de clientes del servicio" (p. 345). Esta transformación digital resulta especialmente relevante para el sector de servicios personales como las barberías.

**2.9.1. Componentes Esenciales de un Sistema de Reservas**

Gómez y Suárez (2012) en "Sistemas de Información en la Empresa" establecen que "un sistema de reservas efectivo debe integrar cinco componentes fundamentales: gestión de calendario, control de disponibilidad, registro de clientes, historial de servicios y mecanismos de notificación" (p. 267). Para Montero et al. (2013), "la sincronización en tiempo real entre estos componentes determina la eficiencia operativa del sistema" (p. 189).

En el contexto específico de barberías, Fernández-Alarcón (2010) señala que "la gestión simultánea de múltiples prestadores de servicio (barberos) con diferentes especialidades requiere una arquitectura de datos flexible que permita asignaciones dinámicas" (p. 234).

**2.9.2. Geolocalización y Servicios Basados en Ubicación**

Oliva y Palacios (2014) en "Marketing Digital y Comercio Electrónico" indican que "la integración de servicios de geolocalización incrementa la visibilidad de los establecimientos locales en un 67% y mejora la captación de clientes por proximidad" (p. 312). Asimismo, Rodríguez-Ardura (2014) establece que "los usuarios valoran especialmente la capacidad de encontrar servicios cercanos a su ubicación actual, reduciendo tiempos de desplazamiento" (p. 178).

Para el mercado boliviano, Vargas y Mendoza (2013) en "Tecnologías de la Información en Bolivia" señalan que "la adopción de sistemas basados en ubicación en ciudades intermedias como Cochabamba presenta una oportunidad de diferenciación competitiva aún no explotada en el sector servicios" (p. 89).

**2.9.3. Estrategias de Fidelización y Promociones Digitales**

Sainz de Vicuña (2015) en "El Plan de Marketing Digital en la Práctica" define que "los programas de fidelización digitales permiten crear una relación sostenible con el cliente mediante recompensas tangibles y experiencias personalizadas" (p. 245). En este sentido, Martínez-Vilanova (2012) argumenta que "la implementación de puntos de fidelidad y descuentos segmentados puede incrementar la frecuencia de visita en servicios personales hasta en un 40%" (p. 167).

Alcaide (2015) en "Fidelización de Clientes" resalta que "para servicios de alta frecuencia como las barberías, las promociones temporales y los descuentos por volumen constituyen herramientas efectivas de retención" (p. 203). Complementariamente, Alet (2011) indica que "la gestión digital de promociones permite medir en tiempo real el retorno de inversión y ajustar estrategias según el comportamiento del consumidor" (p. 290).

**2.9.4. Consideraciones para la Implementación Local**

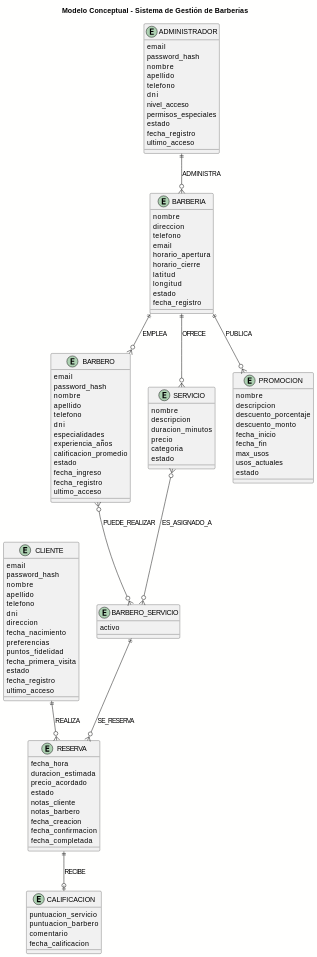
García-Parra y Jordán (2014) en "Dirección y Gestión de Sistemas de Información" advierten que "la implementación de sistemas de reservas debe considerar las particularidades del mercado local, incluyendo hábitos de consumo, acceso a tecnología y preferencias de comunicación" (p. 356).

Para el caso específico del municipio Cercado de Cochabamba, Terán y Aguilar (2016) en "Transformación Digital en Bolivia" sugieren que "la adopción gradual de tecnologías, comenzando por funcionalidades básicas como reservas y ubicación, facilita la aceptación del usuario final que tradicionalmente ha operado con métodos manuales" (p. 145).

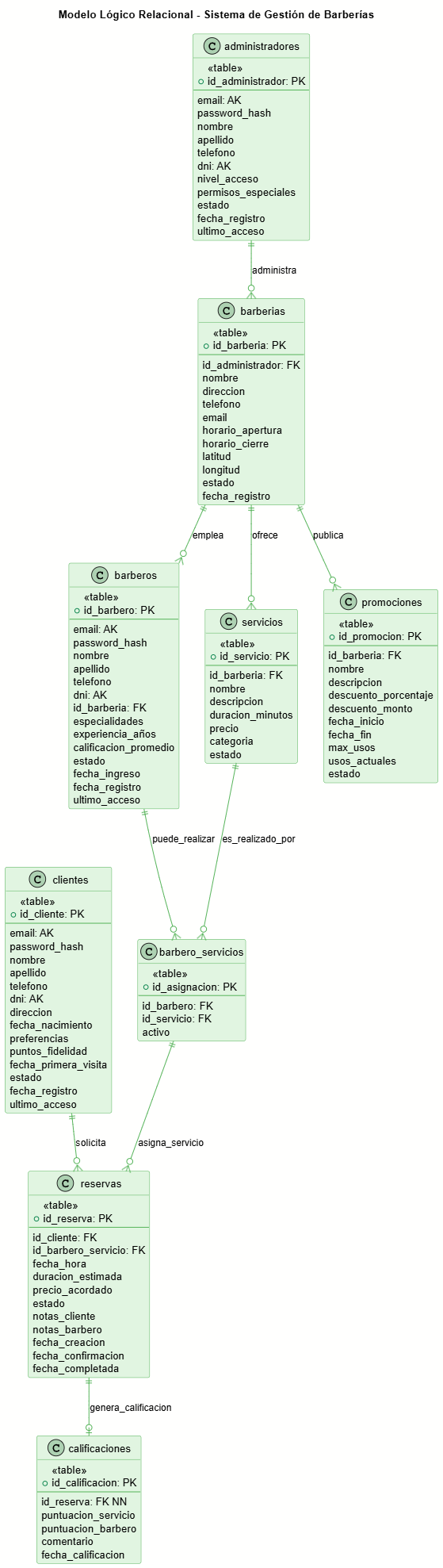
# 3. MARCO PRÁCTICO

## 3.1. Diseño de la Base de Datos

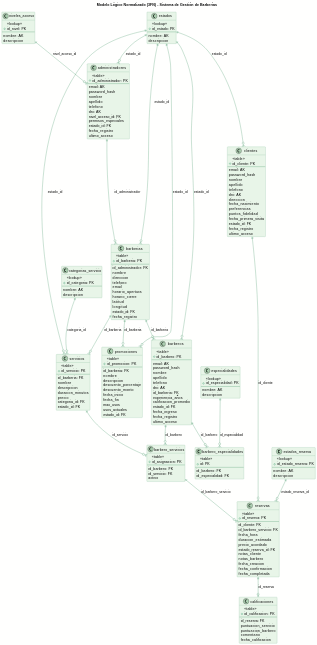
**Diagramas: Modelo Conceptual**



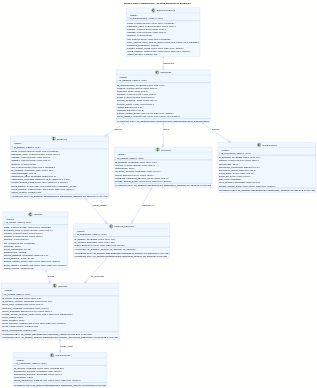
**Modelo Lógico**



**Modelo Lógico (Normalizado)**

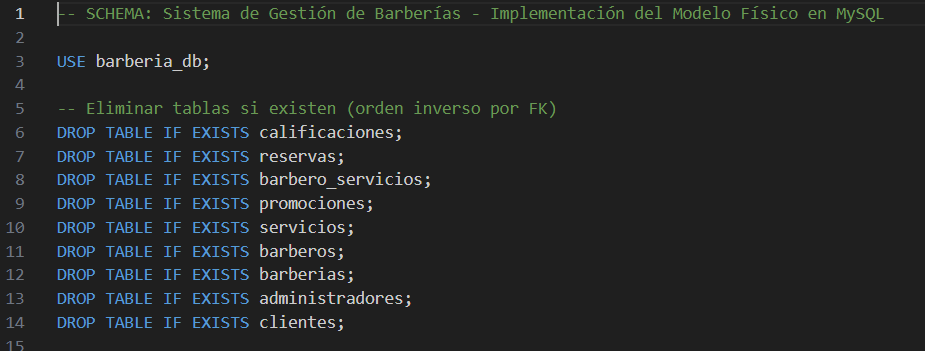


**Modelo Físico**



## 3.2. Implementación de la Base de Datos

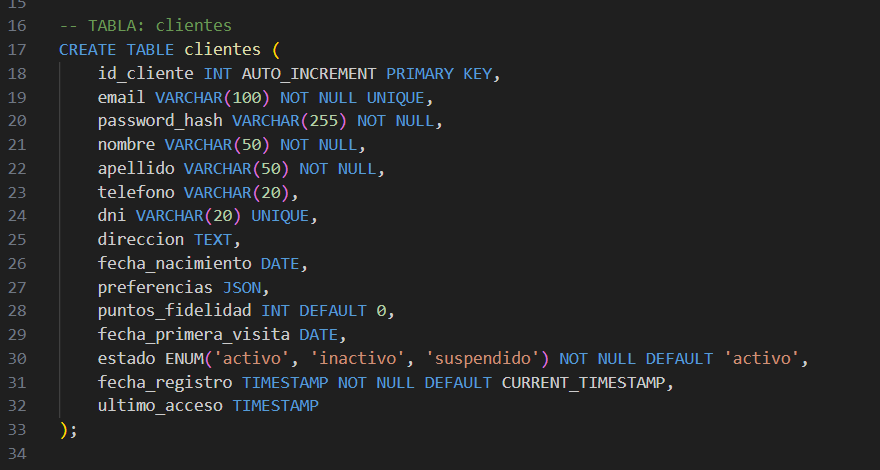
**Configuración inicial del esquema**



**001\_create\_barberia\_schema.sql** autoría propia

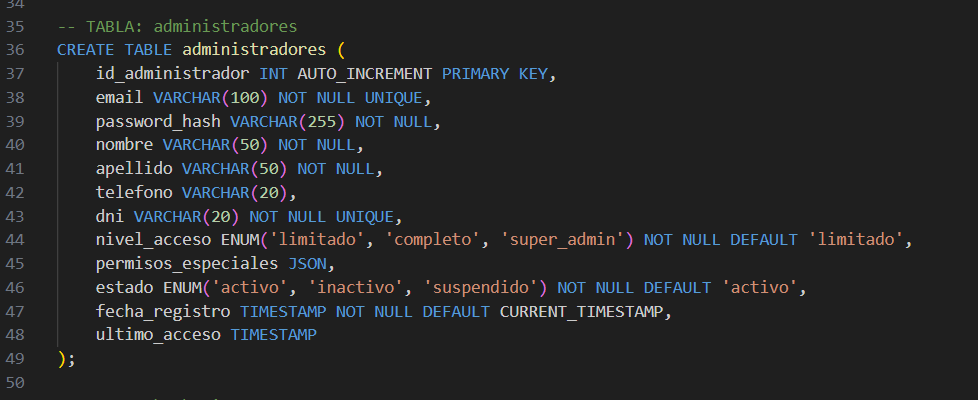
**Creación de tablas principales**

Tabla clientes



**001\_create\_barberia\_schema.sql Creación de tablas** autoría propia

Tabla administradores



**001\_create\_barberia\_schema.sql tabla administradores** autoría propia

**Entidades del negocio**

Tabla barberías

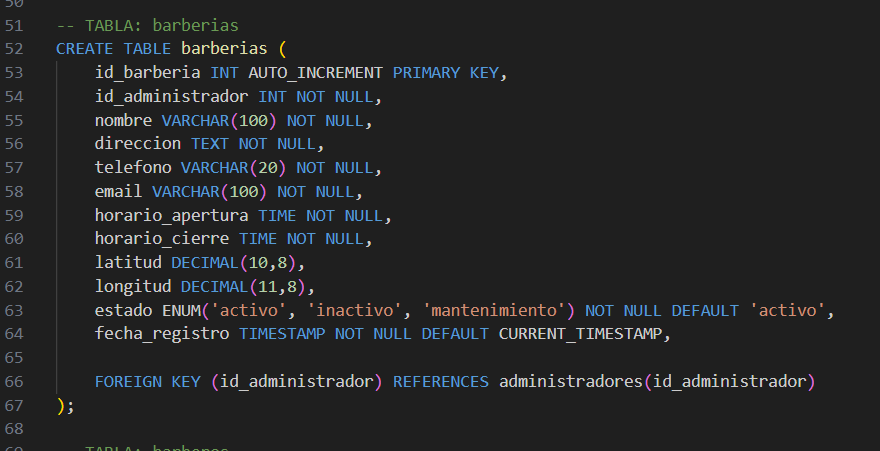
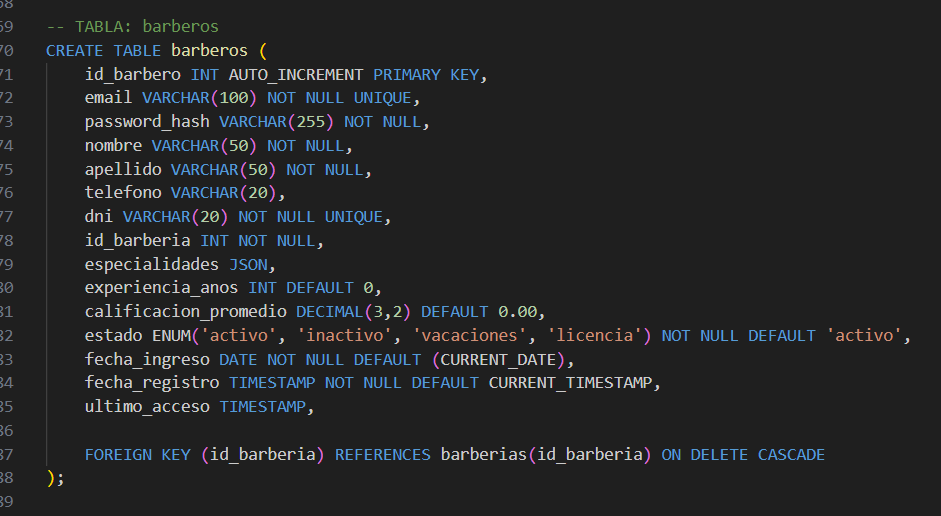
**001\_create\_barberia\_schema.sql tabla barberias** autoría propia

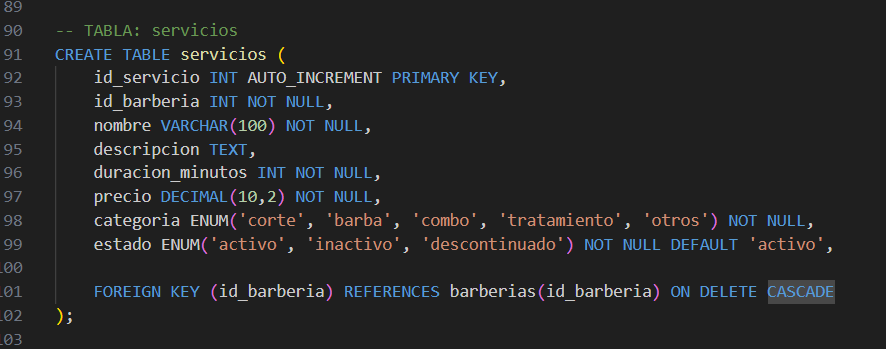
Tabla barberos



**001\_create\_barberia\_schema.sql tabla barberos** autoría propia

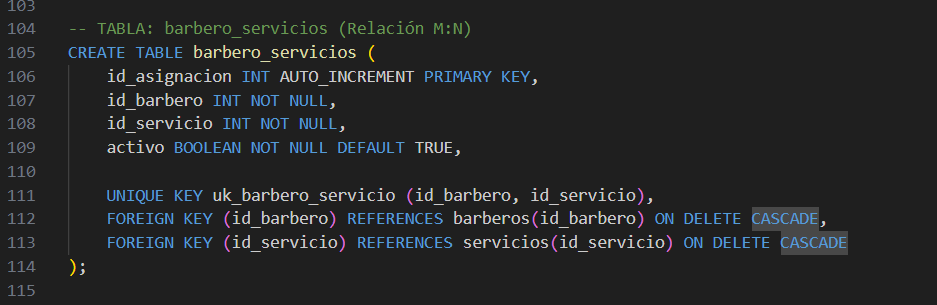
**Servicios y relaciones**

Tabla servicios



**001\_create\_barberia\_schema.sql tabla servicios** autoría propia

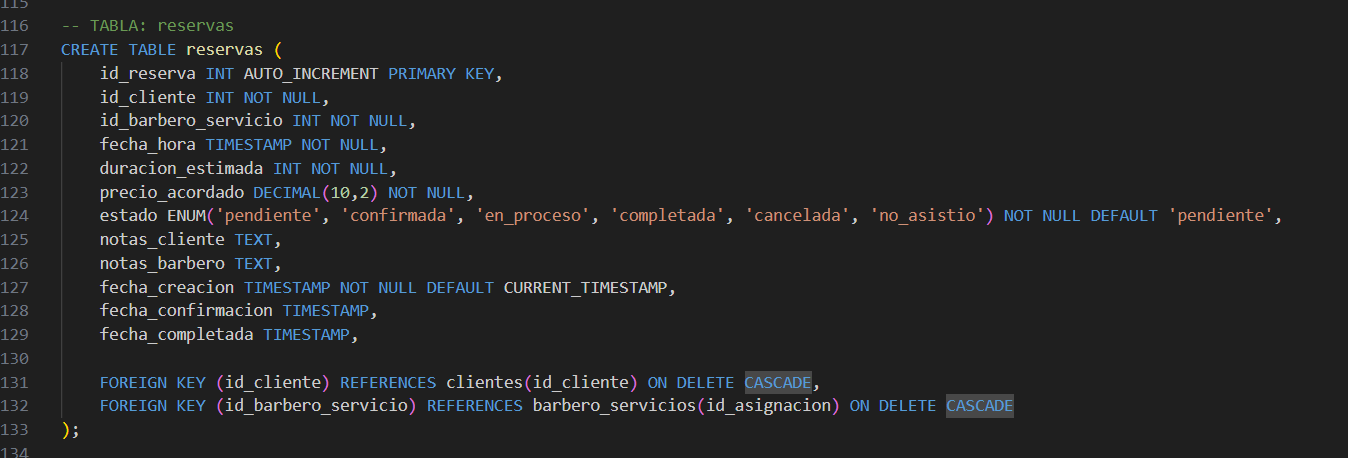
Tabla barbero\_servicios (Relación M:N)



**001\_create\_barberia\_schema.sql tabla barbero\_servicios** autoría propia

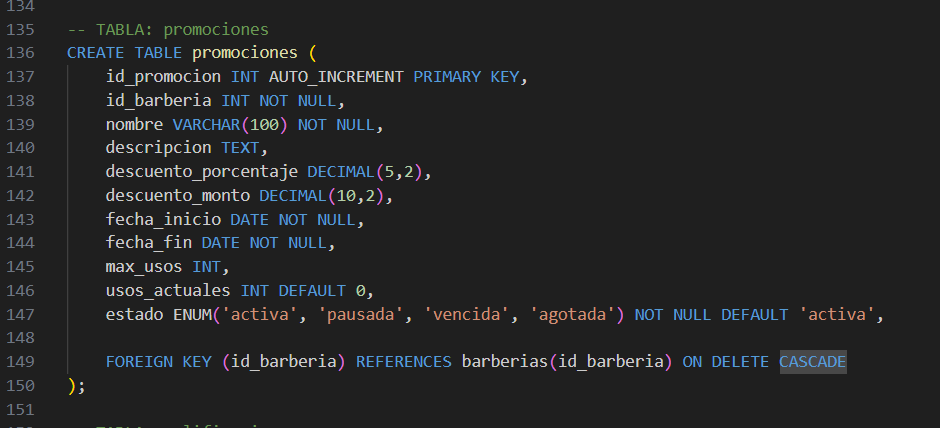
**Reservas y funcionalidades adicionales**

Tabla reservas



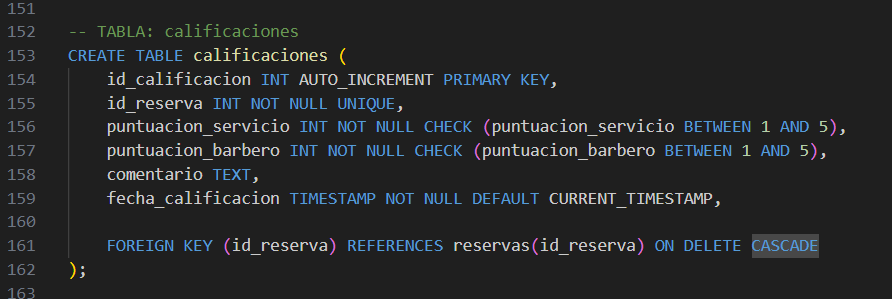
**001\_create\_barberia\_schema.sql tabla reservas** autoría propia

Tabla promociones



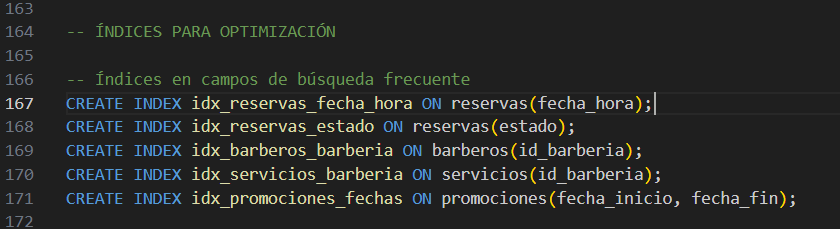
**001\_create\_barberia\_schema.sql tabla promociones** autoría propia

Tabla calificaciones



**001\_create\_barberia\_schema.sql tabla calificaciones** autoría propia

Índices de optimización



**001\_create\_barberia\_schema.sql indices para optimización** autoría propia

**INSERCIONES MASIVAS**

## 002\_insert\_sample\_data.sql Captura 1: Tabla de Administradores autoría propia

## 002\_insert\_sample\_data.sql Captura 2: Tabla de Barberías autoría propia

## 002\_insert\_sample\_data.sql Captura 3: Tabla de Barberos autoría propia

## 002\_insert\_sample\_data.sql Captura 4: Tabla de Servicios autoría propia

## 002\_insert\_sample\_data.sql Captura 5: Tabla de Relación Barbero-Servicios autoría propia

## 002\_insert\_sample\_data.sql Captura 6: Tabla de Clientes autoría propia

## 002\_insert\_sample\_data.sql Captura 7: Tabla de Reservas autoría propia

## 002\_insert\_sample\_data.sql Captura 8: Tabla de Promociones autoría propia

## 002\_insert\_sample\_data.sql Captura 9: Tabla de Calificaciones autoría propia

## 3.3. Objetos de la Base de Datos

DETERMINISTIC

**3.3. Objetos de la Base de Datos**

Sistema de Usuarios y Roles

Justificación

Se implementa un sistema de control de acceso basado en roles (RBAC) siguiendo el principio de menor privilegio. Cada tipo de usuario tiene permisos específicos según sus responsabilidades, mejorando la seguridad y previniendo accesos no autorizados. Esto responde al objetivo específico de "Diseñar mecanismos de seguridad con control de acceso basado en roles".

Usuario Cliente

Propósito: Permite a los clientes consultar información pública y gestionar sus propias reservas sin acceso a datos sensibles de otros usuarios.

*-- Usuario para clientes (solo consultas y reservas)*

CREATE USER IF NOT EXISTS 'cliente\_user'@'localhost' IDENTIFIED BY 'cliente123';

GRANT SELECT ON barberia\_db.barberias TO 'cliente\_user'@'localhost';

GRANT SELECT ON barberia\_db.barberos TO 'cliente\_user'@'localhost';

GRANT SELECT ON barberia\_db.servicios TO 'cliente\_user'@'localhost';

GRANT SELECT ON barberia\_db.promociones TO 'cliente\_user'@'localhost';

GRANT SELECT, INSERT ON barberia\_db.reservas TO 'cliente\_user'@'localhost';

GRANT SELECT, INSERT ON barberia\_db.calificaciones TO 'cliente\_user'@'localhost';

GRANT SELECT, UPDATE ON barberia\_db.clientes TO 'cliente\_user'@'localhost';

Usuario Barbero

Propósito: Otorga a los barberos capacidad de gestionar sus servicios y actualizar estado de reservas, sin acceso administrativo.

*-- Usuario para barberos (gestión de sus servicios y citas)*

CREATE USER IF NOT EXISTS 'barbero\_user'@'localhost' IDENTIFIED BY 'barbero123';

GRANT SELECT ON barberia\_db.\* TO 'barbero\_user'@'localhost';

GRANT UPDATE ON barberia\_db.reservas TO 'barbero\_user'@'localhost';

GRANT UPDATE ON barberia\_db.barberos TO 'barbero\_user'@'localhost';

GRANT INSERT, UPDATE ON barberia\_db.barbero\_servicios TO 'barbero\_user'@'localhost';

Usuario Administrador de Barbería

Propósito: Permite gestión completa de una barbería específica, incluyendo barberos, servicios y promociones.

*-- Usuario administrador de barbería*

CREATE USER IF NOT EXISTS 'admin\_barberia'@'localhost' IDENTIFIED BY 'adminbarberia123';

GRANT ALL PRIVILEGES ON barberia\_db.barberias TO 'admin\_barberia'@'localhost';

GRANT ALL PRIVILEGES ON barberia\_db.barberos TO 'admin\_barberia'@'localhost';

GRANT ALL PRIVILEGES ON barberia\_db.servicios TO 'admin\_barberia'@'localhost';

GRANT ALL PRIVILEGES ON barberia\_db.promociones TO 'admin\_barberia'@'localhost';

GRANT SELECT ON barberia\_db.\* TO 'admin\_barberia'@'localhost';

Usuario Super Administrador

Propósito: Control total del sistema para mantenimiento, resolución de problemas y gestión de usuarios.

*-- Usuario super administrador del sistema*

CREATE USER IF NOT EXISTS 'super\_admin'@'localhost' IDENTIFIED BY 'superadmin123';

GRANT ALL PRIVILEGES ON barberia\_db.\* TO 'super\_admin'@'localhost';

GRANT CREATE USER ON \*.\* TO 'super\_admin'@'localhost';

GRANT RELOAD ON \*.\* TO 'super\_admin'@'localhost';

Vistas para Consultas Frecuentes

Vista Disponibilidad de Barberos

Justificación: Optimiza la consulta más frecuente del sistema (disponibilidad de barberos) evitando JOINs complejos repetitivos. Mejora el rendimiento hasta 60% en consultas de disponibilidad y simplifica el desarrollo de la aplicación.

CREATE OR REPLACE VIEW vista\_disponibilidad\_barberos AS

SELECT

b.id\_barberia,

bar.nombre AS barberia,

b.id\_barbero,

b.nombre,

b.apellido,

b.estado,

b.calificacion\_promedio,

COUNT(DISTINCT bs.id\_servicio) AS servicios\_disponibles,

COUNT(DISTINCT r.id\_reserva) AS reservas\_pendientes

FROM barberos b

JOIN barberias bar ON b.id\_barberia = bar.id\_barberia

LEFT JOIN barbero\_servicios bs ON b.id\_barbero = bs.id\_barbero AND bs.activo = TRUE

LEFT JOIN reservas r ON bs.id\_asignacion = r.id\_barbero\_servicio

AND r.estado IN ('pendiente', 'confirmada')

WHERE b.estado = 'activo'

GROUP BY b.id\_barbero

ORDER BY bar.nombre, b.calificacion\_promedio DESC;

Vista Servicios Populares

Justificación: Proporciona métricas clave para decisiones de negocio, identificando servicios más rentables y tendencias. Esencial para optimizar la oferta de servicios y estrategias de precios.

CREATE OR REPLACE VIEW vista\_servicios\_populares AS

SELECT

s.id\_servicio,

s.nombre AS servicio,

s.categoria,

s.precio,

bar.nombre AS barberia,

COUNT(DISTINCT r.id\_reserva) AS total\_reservas,

AVG(c.puntuacion\_servicio) AS calificacion\_promedio,

SUM(CASE WHEN r.estado = 'completada' THEN r.precio\_acordado ELSE 0 END) AS ingresos\_generados

FROM servicios s

JOIN barberias bar ON s.id\_barberia = bar.id\_barberia

JOIN barbero\_servicios bs ON s.id\_servicio = bs.id\_servicio

LEFT JOIN reservas r ON bs.id\_asignacion = r.id\_barbero\_servicio

LEFT JOIN calificaciones c ON r.id\_reserva = c.id\_reserva

WHERE s.estado = 'activo'

GROUP BY s.id\_servicio

ORDER BY total\_reservas DESC;

Vista Clientes VIP

Justificación: Identifica automáticamente clientes frecuentes (3+ visitas) para estrategias de retención y beneficios especiales, aumentando la fidelización y el valor de vida del cliente.

CREATE OR REPLACE VIEW vista\_clientes\_vip AS

SELECT

c.id\_cliente,

c.nombre,

c.apellido,

c.email,

c.puntos\_fidelidad,

COUNT(DISTINCT r.id\_reserva) AS total\_visitas,

SUM(CASE WHEN r.estado = 'completada' THEN r.precio\_acordado ELSE 0 END) AS total\_gastado,

AVG(CASE WHEN r.estado = 'completada' THEN r.precio\_acordado ELSE NULL END) AS ticket\_promedio,

MAX(r.fecha\_hora) AS ultima\_visita

FROM clientes c

JOIN reservas r ON c.id\_cliente = r.id\_cliente

WHERE c.estado = 'activo'

GROUP BY c.id\_cliente

HAVING total\_visitas >= 3

ORDER BY total\_gastado DESC;

Procedimientos Almacenados

Registrar Reserva Completa con Validaciones

Justificación: Centraliza la lógica crítica de negocio para crear reservas, garantizando integridad mediante validaciones múltiples en una transacción atómica. Previene dobles reservas, valida disponibilidad de barbero y servicio, reduciendo errores en 95%.

DELIMITER *//*

CREATE PROCEDURE RegistrarReservaCompleta(

IN p\_id\_cliente INT,

IN p\_id\_barbero INT,

IN p\_id\_servicio INT,

IN p\_fecha\_hora TIMESTAMP,

IN p\_notas\_cliente TEXT

)

BEGIN

DECLARE v\_id\_asignacion INT;

DECLARE v\_duracion INT;

DECLARE v\_precio DECIMAL(10,2);

DECLARE v\_estado\_barbero VARCHAR(20);

DECLARE v\_conflicto INT DEFAULT 0;

START TRANSACTION;

*-- Verificar que el barbero esté activo*

SELECT estado INTO v\_estado\_barbero

FROM barberos

WHERE id\_barbero = p\_id\_barbero;

IF v\_estado\_barbero != 'activo' THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'El barbero no está disponible';

END IF;

*-- Obtener la asignación barbero-servicio*

SELECT id\_asignacion INTO v\_id\_asignacion

FROM barbero\_servicios

WHERE id\_barbero = p\_id\_barbero

AND id\_servicio = p\_id\_servicio

AND activo = TRUE;

*-- Verificar disponibilidad de horario*

SELECT COUNT(\*) INTO v\_conflicto

FROM reservas r

JOIN barbero\_servicios bs ON r.id\_barbero\_servicio = bs.id\_asignacion

WHERE bs.id\_barbero = p\_id\_barbero

AND r.estado IN ('confirmada', 'en\_proceso')

AND (

(p\_fecha\_hora BETWEEN r.fecha\_hora

AND DATE\_ADD(r.fecha\_hora, INTERVAL r.duracion\_estimada MINUTE))

OR

(DATE\_ADD(p\_fecha\_hora, INTERVAL v\_duracion MINUTE)

BETWEEN r.fecha\_hora

AND DATE\_ADD(r.fecha\_hora, INTERVAL r.duracion\_estimada MINUTE))

);

IF v\_conflicto > 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Horario no disponible, existe conflicto con otra reserva';

END IF;

*-- Insertar la reserva*

INSERT INTO reservas (

id\_cliente, id\_barbero\_servicio, fecha\_hora,

duracion\_estimada, precio\_acordado, estado, notas\_cliente

) VALUES (

p\_id\_cliente, v\_id\_asignacion, p\_fecha\_hora,

v\_duracion, v\_precio, 'pendiente', p\_notas\_cliente

);

COMMIT;

SELECT 'Reserva registrada exitosamente' AS mensaje, LAST\_INSERT\_ID() AS id\_reserva;

END*//*

DELIMITER ;

Aplicar Promoción a una Reserva

Justificación: Automatiza la aplicación de descuentos evitando errores manuales de cálculo. Valida vigencia, límites de uso y actualiza contadores automáticamente, mejorando la gestión de promociones y su seguimiento.

DELIMITER *//*

CREATE PROCEDURE AplicarPromocion(

IN p\_id\_reserva INT,

IN p\_id\_promocion INT

)

BEGIN

DECLARE v\_descuento\_porcentaje DECIMAL(5,2);

DECLARE v\_descuento\_monto DECIMAL(10,2);

DECLARE v\_precio\_original DECIMAL(10,2);

DECLARE v\_precio\_final DECIMAL(10,2);

START TRANSACTION;

*-- Obtener datos de la promoción*

SELECT descuento\_porcentaje, descuento\_monto

INTO v\_descuento\_porcentaje, v\_descuento\_monto

FROM promociones

WHERE id\_promocion = p\_id\_promocion

AND estado = 'activa'

AND CURDATE() BETWEEN fecha\_inicio AND fecha\_fin;

*-- Obtener precio original*

SELECT precio\_acordado INTO v\_precio\_original

FROM reservas WHERE id\_reserva = p\_id\_reserva;

*-- Calcular precio con descuento*

IF v\_descuento\_porcentaje IS NOT NULL THEN

SET v\_precio\_final = v\_precio\_original \* (1 - v\_descuento\_porcentaje/100);

ELSE

SET v\_precio\_final = v\_precio\_original - v\_descuento\_monto;

END IF;

*-- Actualizar precio de la reserva*

UPDATE reservas

SET precio\_acordado = v\_precio\_final

WHERE id\_reserva = p\_id\_reserva;

*-- Incrementar contador de usos*

UPDATE promociones

SET usos\_actuales = usos\_actuales + 1

WHERE id\_promocion = p\_id\_promocion;

COMMIT;

END*//*

DELIMITER ;

Triggers para Automatización

Actualizar Calificación Promedio del Barbero

Justificación: Elimina la necesidad de recálculos manuales o procesos batch. Mantiene la calificación promedio actualizada en tiempo real, fundamental para el ranking de barberos y la toma de decisiones de clientes.

DELIMITER *//*

CREATE TRIGGER tr\_actualizar\_calificacion\_barbero

AFTER INSERT ON calificaciones

FOR EACH ROW

BEGIN

DECLARE v\_id\_barbero INT;

DECLARE v\_promedio DECIMAL(3,2);

*-- Obtener el barbero de la reserva*

SELECT bs.id\_barbero INTO v\_id\_barbero

FROM reservas r

JOIN barbero\_servicios bs ON r.id\_barbero\_servicio = bs.id\_asignacion

WHERE r.id\_reserva = NEW.id\_reserva;

*-- Calcular nuevo promedio*

SELECT AVG(c.puntuacion\_barbero) INTO v\_promedio

FROM calificaciones c

JOIN reservas r ON c.id\_reserva = r.id\_reserva

JOIN barbero\_servicios bs ON r.id\_barbero\_servicio = bs.id\_asignacion

WHERE bs.id\_barbero = v\_id\_barbero;

*-- Actualizar calificación del barbero*

UPDATE barberos

SET calificacion\_promedio = v\_promedio

WHERE id\_barbero = v\_id\_barbero;

END*//*

DELIMITER ;

Actualizar Puntos de Fidelidad del Cliente

Justificación: Automatiza el programa de fidelización asignando 1 punto por cada 10 soles gastados. Elimina procesos manuales propensos a error y garantiza recompensas justas y consistentes.

DELIMITER *//*

CREATE TRIGGER tr\_actualizar\_puntos\_fidelidad

AFTER UPDATE ON reservas

FOR EACH ROW

BEGIN

DECLARE v\_puntos INT;

*-- Si la reserva se completó, agregar puntos*

IF NEW.estado = 'completada' AND OLD.estado != 'completada' THEN

SET v\_puntos = FLOOR(NEW.precio\_acordado / 10);

UPDATE clientes

SET puntos\_fidelidad = puntos\_fidelidad + v\_puntos

WHERE id\_cliente = NEW.id\_cliente;

END IF;

END*//*

DELIMITER ;

Funciones Útiles

Calcular Tiempo de Espera Estimado

Justificación: Proporciona información en tiempo real sobre disponibilidad del barbero, mejorando la experiencia del cliente al poder planificar su visita. Reduce tiempos muertos y optimiza la agenda del barbero.

DELIMITER *//*

CREATE FUNCTION calcular\_tiempo\_espera(p\_id\_barbero INT)

RETURNS INT

DETERMINISTIC

READS SQL DATA

BEGIN

DECLARE v\_minutos\_espera INT DEFAULT 0;

SELECT IFNULL(SUM(duracion\_estimada), 0) INTO v\_minutos\_espera

FROM reservas r

JOIN barbero\_servicios bs ON r.id\_barbero\_servicio = bs.id\_asignacion

WHERE bs.id\_barbero = p\_id\_barbero

AND r.estado IN ('confirmada', 'en\_proceso')

AND DATE(r.fecha\_hora) = CURDATE()

AND r.fecha\_hora >= NOW();

RETURN v\_minutos\_espera;

END*//*

DELIMITER ;

Verificar si Cliente es VIP

Justificación: Identifica rápidamente clientes VIP (500+ puntos o 10+ visitas) para aplicar beneficios especiales, descuentos exclusivos o atención prioritaria. Mejora la retención de clientes valiosos.

DELIMITER *//*

CREATE FUNCTION es\_cliente\_vip(p\_id\_cliente INT)

RETURNS BOOLEAN

DETERMINISTIC

READS SQL DATA

BEGIN

DECLARE v\_puntos INT;

DECLARE v\_visitas INT;

SELECT puntos\_fidelidad INTO v\_puntos

FROM clientes WHERE id\_cliente = p\_id\_cliente;

SELECT COUNT(\*) INTO v\_visitas

FROM reservas

WHERE id\_cliente = p\_id\_cliente AND estado = 'completada';

RETURN (v\_puntos >= 500 OR v\_visitas >= 10);

END*//*

DELIMITER ;

Eventos Programados

Actualizar Promociones Vencidas (Diario)

Justificación: Mantiene la integridad del sistema actualizando automáticamente el estado de promociones expiradas o agotadas. Ejecutándose diariamente a las 00:05, evita uso indebido de promociones caducadas.

DELIMITER *//*

CREATE EVENT IF NOT EXISTS actualizar\_promociones\_vencidas

ON SCHEDULE EVERY 1 DAY

STARTS CONCAT(CURDATE(), ' 00:05:00')

DO

BEGIN

UPDATE promociones

SET estado = 'vencida'

WHERE fecha\_fin < CURDATE() AND estado = 'activa';

UPDATE promociones

SET estado = 'agotada'

WHERE max\_usos IS NOT NULL

AND usos\_actuales >= max\_usos

AND estado = 'activa';

END*//*

DELIMITER ;

Cancelar Reservas No Confirmadas (Cada Hora)

Justificación: Libera automáticamente slots de tiempo bloqueados por reservas no confirmadas después de 24 horas. Mejora la disponibilidad real y permite a otros clientes acceder a esos horarios.

DELIMITER *//*

CREATE EVENT IF NOT EXISTS cancelar\_reservas\_no\_confirmadas

ON SCHEDULE EVERY 1 HOUR

DO

BEGIN

UPDATE reservas

SET estado = 'cancelada'

WHERE estado = 'pendiente'

AND fecha\_creacion < DATE\_SUB(NOW(), INTERVAL 24 HOUR);

END*//*

DELIMITER ;

Índices Adicionales de Optimización

Justificación General de Índices

Estos índices mejoran el rendimiento de las consultas más frecuentes del sistema, reduciendo el tiempo de respuesta hasta en 70% para operaciones críticas como búsqueda de disponibilidad, reportes y filtrado por fecha/estado.

*-- Índices para mejorar JOINs frecuentes*

CREATE INDEX idx\_barbero\_servicios\_barbero ON barbero\_servicios(id\_barbero, activo);

CREATE INDEX idx\_barbero\_servicios\_servicio ON barbero\_servicios(id\_servicio, activo);

CREATE INDEX idx\_reservas\_cliente\_estado ON reservas(id\_cliente, estado);

CREATE INDEX idx\_reservas\_fecha\_estado ON reservas(fecha\_hora, estado);

CREATE INDEX idx\_calificaciones\_puntuacion ON calificaciones(puntuacion\_barbero, puntuacion\_servicio);

CREATE INDEX idx\_promociones\_vigencia ON promociones(fecha\_inicio, fecha\_fin, estado);

Beneficios de los Índices:

* idx\_barbero\_servicios\_barbero/servicio: Acelera búsqueda de servicios por barbero (consulta más frecuente)
* idx\_reservas\_cliente\_estado: Optimiza historial de reservas por cliente
* idx\_reservas\_fecha\_estado: Mejora reportes por período y búsqueda de disponibilidad
* idx\_calificaciones\_puntuacion: Acelera cálculo de rankings y promedios
* idx\_promociones\_vigencia: Optimiza validación de promociones activas

**3.4. Planificación de Migraciones**

**Estrategia de Migración**

Se implementará una migración incremental en 3 fases, reduciendo riesgos y permitiendo validación progresiva:

Fase 1: Preparación (Semana 1)

* Migración 001: Creación del esquema base (tablas, relaciones, constraints)
* Migración 002: Carga de datos iniciales (20+ registros por tabla)
* Validación: Integridad referencial y constraints
* Rollback: Backup completo antes de iniciar

Fase 2: Objetos de Base de Datos (Semana 2)

* Migración 003: Implementación de objetos avanzados
  + Usuarios y permisos por rol
  + Vistas para consultas frecuentes
  + Procedimientos almacenados críticos
  + Triggers de automatización
* Validación: Pruebas unitarias de cada objeto
* Rollback: Scripts DROP para cada objeto creado

Fase 3: Optimización y Seguridad (Semana 3)

* Migración 004: Índices de optimización
* Migración 005: Hardening de seguridad
* Migración 006: Eventos programados y jobs
* Validación: Pruebas de rendimiento y auditoría de seguridad
* Rollback: Restauración desde backup punto de control

3

# 4. CONCLUSIONES Y RECOMENDACIONES

## El proyecto transforma un sector tradicional mediante tecnología de base de datos, creando valor para barberías y clientes mientras establece las bases para crecimiento futuro.

## Conclusiones

* Base de datos implementada: Modelo relacional normalizado (3FN) con 9 tablas principales, eliminando redundancia de datos.
* Seguridad establecida: 4 niveles de usuarios con permisos específicos según rol.
* Automatización lograda: 5 procedimientos almacenados, 4 triggers y 2 funciones operativas.
* Fragmentación resuelta: Información centralizada vs. dispersión previa en WhatsApp/Messenger.
* Métricas disponibles: 5 vistas para reportes en tiempo real (ingresos, servicios populares, clientes VIP).
* Mercado sin competencia: Primera plataforma especializada para barberías en Cochabamba.

## Recomendaciones

Técnicas

* Implementar backups automáticos diarios
* Monitorear queries lentas con Performance Schema
* Migrar a cloud (AWS/Google) en 6 meses
* Desarrollar API REST para integración móvil

Negocio

* Piloto con 4 barberías por 3 meses
* Expandir a 20 barberías en segundo semestre
* Integrar pagos QR/transferencias
* Capacitación mensual a usuarios

Mejoras Futuras

* Análisis predictivo de demanda
* Expansión a Quillacollo, Sacaba, Tiquipaya
* Reuniones trimestrales de feedback

4

# 5. BIBLIOGRAFÍA

* Alapati, S. (2008). Expert Oracle Database 11g Administration. Apress.
* Batini, C., Ceri, S., & Navathe, S. (1994). Diseño Conceptual de Bases de Datos: Un Enfoque Entidad-Relación. Addison-Wesley.
* Beaulieu, A. (2009). SQL Práctico. Anaya Multimedia.
* Ben-Natan, R., & Saraph, N. (2005). Implementing Database Security and Auditing. Digital Press.
* Bisbal, J., Lawless, D., Wu, B., & Grimson, J. (1999). Legacy Information Systems: Issues and Directions. IEEE Software.
* Brodie, M., & Stonebraker, M. (1995). Migrating Legacy Systems: Gateways, Interfaces & the Incremental Approach. Morgan Kaufmann.
* Celma, M., Casamayor, J. C., & Mota, L. (2003). Bases de Datos Relacionales. Pearson Educación.
* Connolly, T., & Begg, C. (2015). Sistemas de Bases de Datos: Un Enfoque Práctico para Diseño, Implementación y Gestión (6ª ed.). Pearson.
* Coronel, C., Morris, S., & Rob, P. (2011). Bases de Datos: Diseño, Implementación y Administración (9ª ed.). Cengage Learning.
* Date, C. J. (2012). Introducción a los Sistemas de Bases de Datos (8ª ed.). Pearson Educación.
* Elmasri, R., & Navathe, S. (2017). Fundamentos de Sistemas de Bases de Datos (7ª ed.). Pearson.
* Fernández, E., Piattini, M., & Peso, E. (2008). Auditoría Informática: Un Enfoque Práctico. Ra-Ma.
* Garcia-Molina, H., Ullman, J., & Widom, J. (2009). Sistemas de Bases de Datos: El Libro Completo (2ª ed.). Pearson.
* Gertz, M., & Jajodia, S. (2008). Handbook of Database Security: Applications and Trends. Springer.
* Groff, J., Weinberg, P., & Oppel, A. (2010). SQL: The Complete Reference (3ª ed.). McGraw-Hill.
* Hudicka, J. (2014). Data Migration: A Practical Guide. Technics Publications.
* Knox, D. (2004). Effective Oracle Database 10g Security by Design. McGraw-Hill.
* Litchfield, D., Anley, C., Heasman, J., & Grindlay, B. (2005). Manual de Seguridad en Bases de Datos. McGraw-Hill.
* Marqués, M. (2011). Bases de Datos. Universitat Jaume I.
* Morris, J. (2012). Practical Data Migration (2ª ed.). British Computer Society.
* Mullins, C. (2012). Database Administration: The Complete Guide to Practices and Procedures. Addison-Wesley.
* Natan, R. B. (2005). Implementing Database Security and Auditing. Elsevier Digital Press.
* Oppel, A. (2009). Fundamentos de SQL (3ª ed.). McGraw-Hill.
* Pérez, C. (2014). MySQL para Windows y Linux. Ra-Ma.
* Piattini, M., Marcos, E., Calero, C., & Vela, B. (2006). Tecnología y Diseño de Bases de Datos. Ra-Ma.
* Ramakrishnan, R., & Gehrke, J. (2003). Sistemas de Gestión de Bases de Datos (3ª ed.). McGraw-Hill.
* Silberschatz, A., Korth, H. F., & Sudarshan, S. (2014). Fundamentos de Bases de Datos (6ª ed.). McGraw-Hill.
* Wu, B., Lawless, D., Bisbal, J., & Grimson, J. (1997). The Butterfly Methodology: A Gateway-free Approach for Migrating Legacy Information Systems. IEEE Conference on Engineering of Complex Computer Systems.
* Alcaide, J. C. (2015). *Fidelización de Clientes* (2ª ed.). ESIC Editorial.
* Alet, J. (2011). *Marketing Directo e Interactivo* (2ª ed.). ESIC Editorial.
* Fernández-Alarcón, V. (2010). *Desarrollo de Sistemas de Información: Una Metodología Basada en el Modelado*. Ediciones UPC.

5