



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TSDS)

ASIGNATURA:

Bases de Datos

PROFESOR:

Ing. Lorena Chulde

FECHA DE ENTREGA:

2025 - 02-05

PERÍODO ACADÉMICO:

2024-B

PROYECTO FINAL

TÍTULO

SISTEMA DE GESTIÓN PARA FERRETERÍA



Estudiantes

Erick Nuñez
Jhonny Villanueva
Adrián Caiza
Paul Sandoval

ÍNDICE

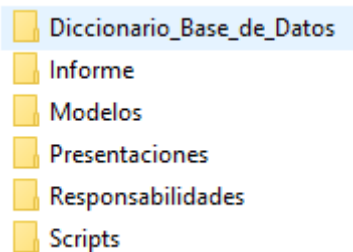
1. RESUMEN EJECUTIVO.....	
2. INTRODUCCIÓN.....	
2.1.CONTEXTO Y JUSTIFICACIÓN DEL TEMA ELEGIDO.....	
3. OBJETIVOS DEL PROYECTO Y METODOLOGÍA.....	
3.1.METODOLOGÍA.....	
4. ALCANCE DEL PROYECTO.....	
5. DESCRIPCIÓN DEL NEGOCIO.....	
6. MODELO ENTIDAD-RELACIÓN (MER).....	
7. DESCRIPCIÓN DE CADA FASE DEL PROYECTO.....	
7.1.MODELADO DE LA BASE DE DATOS.....	
7.2.IMPLEMENTACIÓN DE FUNCIONALIDADES Y VISTAS.....	
7.3.SEGURIDAD, AUDITORÍA Y CONTROL DE ACCESO.....	
7.4.RESPALDOS Y RECUPERACIÓN DE DATOS.....	
7.5.OPTIMIZACIÓN Y RENDIMIENTO.....	
8. RESULTADOS OBTENIDOS.....	
9. CONCLUSIONES.....	
10.RECOMENDACIONES ADICIONALES.....	

RESUMEN EJECUTIVO

Este proyecto tiene como objetivo desarrollar un sistema de gestión para una ferretería utilizando una base de datos relacional en MySQL. Se diseñó un modelo de datos eficiente que permite administrar clientes, empleados, productos, ventas y devoluciones de manera estructurada y segura. Además, se implementaron vistas y procedimientos almacenados para optimizar el acceso a la información y mejorar el rendimiento del sistema. Se aseguró la integridad de los datos mediante claves primarias y foráneas, y se aplicaron medidas de seguridad con roles y permisos para restringir accesos no autorizados.



A su vez para garantizar una mejor organización de datos decidimos crear una carpeta llamada Proyecto_Ferreteria.zip en la cual se encuentran varias subcarpetas organizadas donde se presentan toda la información requerida para el informe:



INTRODUCCIÓN

Contexto y Justificación del tema elegido

Como sabemos una ferretería requiere de un sistema eficiente para administrar su inventario, gestionar clientes, empleados y registrar ventas y devoluciones. Esto se logra con un sistema de base de datos bien diseñado el cual mejora la organización de la información, facilita la toma de decisiones y reduce errores administrativos.



(Imágenes referenciales de stock de productos en una ferretería)

OBJETIVOS DEL PROYECTO

Diseñar e implementar un modelo de base de datos que represente la estructura y operación de la ferretería.

- ✓ Optimizar consultas mediante vistas y procedimientos almacenados.
- ✓ Implementar roles y permisos de acceso para mejorar la seguridad de los datos.
- ✓ Realizar respaldos y estrategias de recuperación de datos para prevenir pérdidas de información.

Metodología

El desarrollo del proyecto se realizó en varias fases:

- ✓ Modelado de la Base de Datos: Creación del modelo entidad-relación y normalización.
- ✓ Implementación: Escritura del código SQL para la creación de tablas, restricciones e índices.
- ✓ Optimización: Desarrollo de vistas y procedimientos almacenados.
- ✓ Seguridad: Creación de roles y asignación de permisos.
- ✓ Pruebas: Evaluación de la integridad, rendimiento y seguridad de la base de datos.

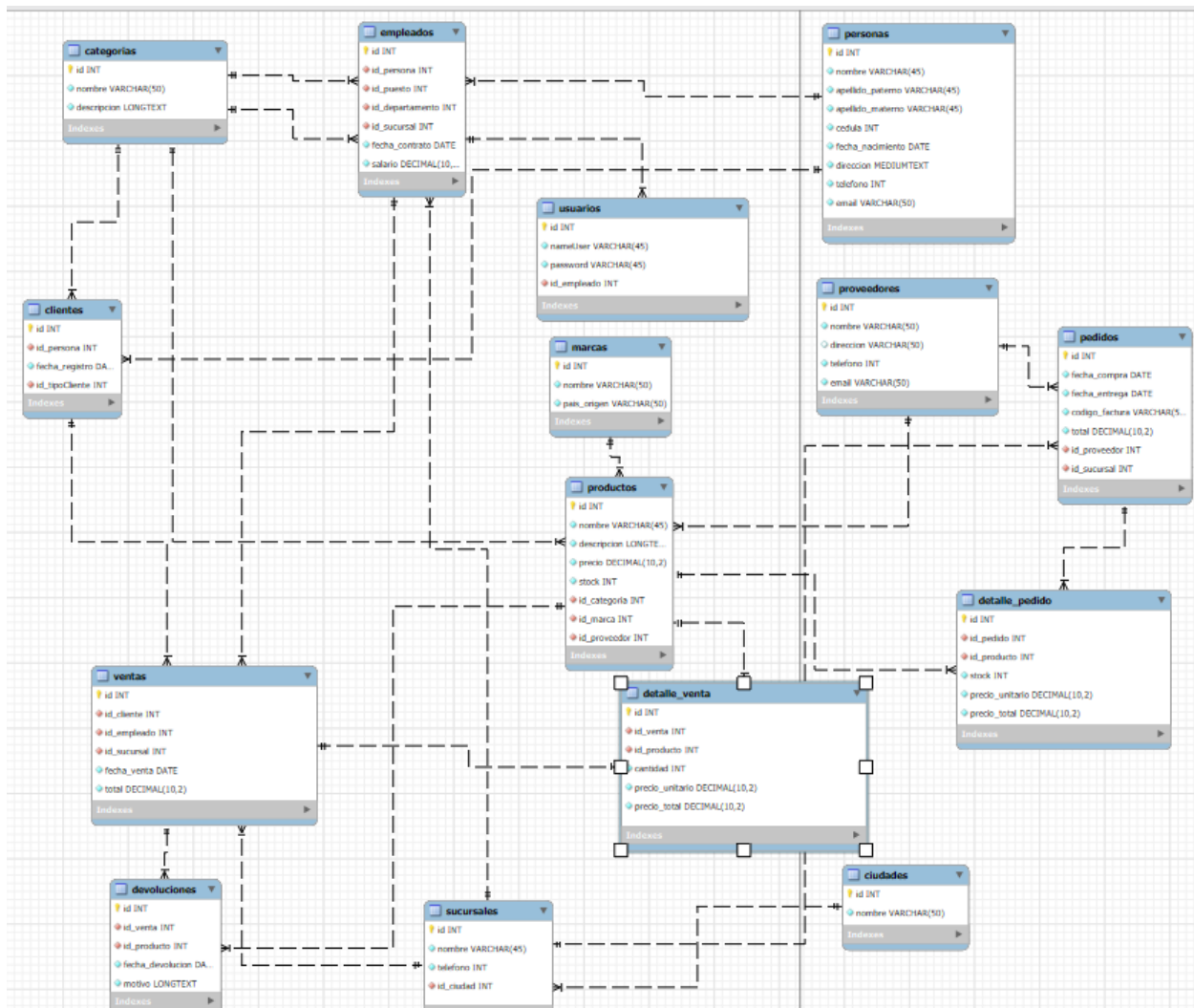
ALCANCE DEL PROYECTO

El sistema que se va a presentar permite la gestión de clientes, empleados, proveedores y productos. También facilita el registro de ventas y facturación, asegurando la integridad y disponibilidad de la información.

DESCRIPCIÓN DEL NEGOCIO

La ferretería ofrece una amplia variedad de productos para la construcción y el hogar. Requiere de un sistema que optimice la administración del stock, facilite el control de ventas y permita la generación de reportes eficientes.

MODELO ENTIDAD-RELACIÓN (MER)



DESCRIPCIÓN DE CADA FASE DEL PROYECTO

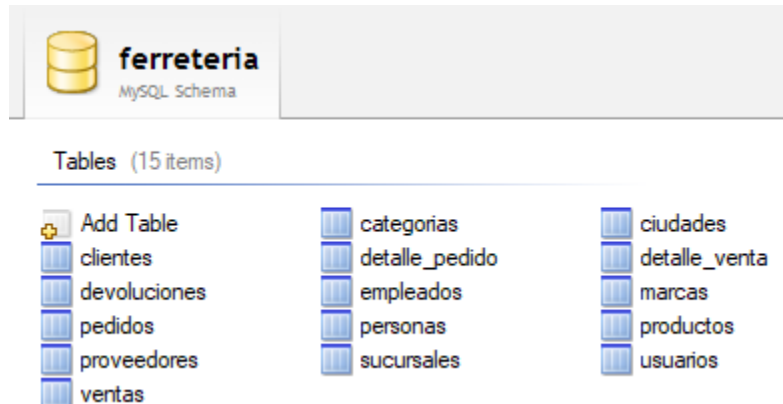
A. Modelado de la Base de Datos

Diseño Conceptual y Lógico

El modelo entidad-relación consta de las siguientes entidades principales:

- **Cientes** (almacena información de los clientes).
- **Empleados** (gestiona los datos de los trabajadores de la ferretería).
- **Productos** (registro del inventario disponible).
- **Ventas y Devoluciones** (registra transacciones de compra y devoluciones).

En total se dispone de 15 tablas las cuales son las siguientes:



Diseño Físico

Se implementaron las tablas en MySQL con claves primarias y foráneas para asegurar la integridad referencial. Se usaron restricciones como **NOT NULL**, y **CHECK** para mejorar la calidad de los datos.

```
-- Table `ferreteria`.`categorias`  
-----  
CREATE TABLE IF NOT EXISTS `ferreteria`.`categorias` (  
  `id` INT NOT NULL COMMENT 'Identificador único de categorías',  
  `nombre` VARCHAR(50) NOT NULL COMMENT 'Nombre de la categoría',  
  `descripcion` LONGTEXT NOT NULL COMMENT 'Descripción de la categoría',  
  PRIMARY KEY (`id`),  
  UNIQUE INDEX `nombre_UNIQUE` (`nombre` ASC) VISIBLE)  
ENGINE = InnoDB;
```

EXPLICACIÓN DE CODIGOS IMPLEMENTADOS

```
SET SQL_MODE=@OLD_SQL_MODE;
```

Este comando restaura la configuración previa del modo SQL (@OLD_SQL_MODE) que estaba activa antes de realizar cambios.

Propósito:


El SQL_MODE afecta el comportamiento de MySQL en aspectos como la validación de datos, manejo de restricciones y el cumplimiento de ciertas reglas SQL.

```
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
```

Este comando restaura la configuración previa de la verificación de claves foráneas (@OLD_FOREIGN_KEY_CHECKS).

Propósito:

Cuando se desactiva FOREIGN_KEY_CHECKS (por ejemplo, SET FOREIGN_KEY_CHECKS=0;), MySQL permite realizar cambios en tablas relacionadas (como eliminar tablas con claves foráneas) sin verificar las restricciones de integridad referencial. Con esto podemos volver a a habilitar la verificación de claves foráneas al final del script, garantizando la integridad de los datos nuevamente.



```
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Este comando restaura la configuración previa de las verificaciones de unicidad (@OLD_UNIQUE_CHECKS).

Propósito:

Al desactivar UNIQUE_CHECKS (por ejemplo, SET UNIQUE_CHECKS=0;), MySQL permite insertar o modificar datos sin verificar restricciones de unicidad (como claves únicas). Esto nos resulta más útil al importar grandes cantidades de datos.

B. Implementación de Funcionalidades y Vistas

Vistas

Se crearon vistas para facilitar consultas comunes, como **v_info_clientes**, que muestra información consolidada de los clientes,

```
CREATE VIEW v_info_clientes AS
SELECT c.id, p.nombre, p.apellido_paterno, p.apellido_materno, p.cedula, p.telefono, p.email, c.fecha_registro, cat.nombre AS tipo_cliente
FROM clientes c
JOIN personas p ON c.id_persona = p.id
JOIN categorias cat ON c.id_tipoCliente = cat.id;
```

y **v_info_productos**, que permite acceder rápidamente al inventario.

```
CREATE VIEW v_info_productos AS
SELECT p.id, p.nombre, p.descripcion, p.precio, p.stock, c.nombre AS categoria, m.nombre AS marca, prov.nombre AS proveedor
FROM productos p
JOIN categorias c ON p.id_categoria = c.id
JOIN marcas m ON p.id_marca = m.id
JOIN proveedores prov ON p.id_proveedor = prov.id
ORDER BY p.id asc;
```

Procedimientos Almacenados

Se desarrollaron procedimientos como:

- **ContarClientes():** Devuelve la cantidad de clientes registrados.

```

DELIMITER $$
CREATE PROCEDURE ContarClientes()
» BEGIN
    SELECT COUNT(*) AS total_clientes FROM clientes;
- END$$
DELIMITER ;

CALL ContarClientes();

```

- **RegistrarCliente():** Inserta un nuevo cliente en la base de datos.

```

-- Función para registrar un nuevo cliente
DELIMITER $$
CREATE PROCEDURE RegistrarCliente(
    IN p_nombre VARCHAR(45),
    IN p_apellido_paterno VARCHAR(45),
    IN p_apellido_materno VARCHAR(45),
    IN p_cedula INT,
    IN p_fecha_nacimiento DATE,
    IN p_direccion MEDIUMTEXT,
    IN p_telefono INT,
    IN p_email VARCHAR(50),
    IN p_id_tipoCliente INT
)
» BEGIN
    DECLARE v_id_persona INT;

    INSERT INTO personas (nombre, apellido_paterno, apellido_materno, cedula, fecha_nacimiento, direccion, telefono, email)
    VALUES (p_nombre, p_apellido_paterno, p_apellido_materno, p_cedula, p_fecha_nacimiento, p_direccion, p_telefono, p_email);

    SET v_id_persona = LAST_INSERT_ID();

    INSERT INTO clientes (id_persona, fecha_registro, id_tipoCliente)
    VALUES (v_id_persona, CURDATE(), p_id_tipoCliente);
- END$$
DELIMITER ;

CALL RegistrarCliente(
    'Juan', 'Pérez', 'Gómez',
    12345678, '1990-05-15',
    'Calle 123, Ciudad',
    987654321, 'juan.perez@example.com',
    1
);

```

- **ObtenerProductosPorCategoria():** Devuelve productos filtrados por categoría.


```

DELIMITER $$
CREATE PROCEDURE ObtenerProductosPorCategoria(
    IN p_categoria_id INT
)
BEGIN
    SELECT * FROM productos WHERE id_categoria = p_categoria_id;
END$$
DELIMITER ;

CALL ObtenerProductosPorCategoria(3);

```

C. Seguridad, Auditoría y Control de Acceso

Configuración de Roles y Permisos

Se definieron tres roles principales:

- **Admin:** Acceso total al sistema.
- **Usuario:** Permiso de consulta y operación en ventas.
- **Auditor:** Solo lectura para revisión de datos.

```


GRANT ALL PRIVILEGES ON ferreteria.* TO 'admin';
GRANT SELECT, INSERT, UPDATE ON ferreteria.* TO 'usuario';
GRANT SELECT ON ferreteria.* TO 'auditor';

```

Se aplicaron restricciones mediante **GRANT** para controlar el acceso a datos sensibles.


D. Respaldos y Recuperación de Datos

Se realizó un respaldo de la base de datos (**ferreteria.mwb.bak**) para garantizar la recuperación en caso de fallos.

 ferreteria.mwb.bak

E. Optimización y Rendimiento

Se implementaron índices en campos clave (**idx_cedula**, **idx_productos_nombres**) para acelerar las consultas. Se utilizaron herramientas como **EXPLAIN** para analizar y mejorar la eficiencia de las consultas SQL.



```
CREATE INDEX idx_cedula ON personas(cedula);  
SHOW INDEXES FROM personas;  
DROP INDEX idx_cedula ON personas;  
  
CREATE INDEX idx_productos_nombres ON productos(nombre);  
SHOW INDEXES FROM productos;
```

RESULTADOS OBTENIDOS

Implementación Exitosa

El sistema cumple con los objetivos planteados, ofreciendo una estructura de base de datos robusta y eficiente.

Beneficios

- Mayor organización y gestión de la información.
- Consultas optimizadas y rápidas.
- Seguridad mediante roles y permisos.
- Estrategia de respaldo implementada.

CONCLUSION

Con el proyecto de base de datos enfocado en la administración de datos de una ferretería, hemos logrado optimizar la gestión del negocio mediante una estructura eficiente y bien organizada. Esto asegura no solo la organización y seguridad de los datos, sino también un acceso más rápido y confiable a la información.

A su vez se implementaron con éxito vistas que simplifican las consultas, procedimientos almacenados que automatizan tareas comunes, y un control de acceso robusto para garantizar la seguridad de los datos. Además, las pruebas realizadas demuestran un rendimiento óptimo en la gestión de operaciones clave, como el registro de ventas, el control de inventarios y la generación de reportes.

RECOMENDACIONES ADICIONALES

A continuación, presentamos una serie de recomendaciones clave que deberían considerarse para la implementación de esta base de datos en una ferretería que requiera un sistema gestor eficiente:

- **Pruebas y Monitoreo:** Realizar pruebas de carga periódicas para garantizar el rendimiento a medida que aumente el volumen de datos y la concurrencia de usuarios.

- **Respaldos Frecuentes:** Implementar una estrategia de respaldos automáticos (incrementales y completos) para minimizar el riesgo de pérdida de información.
- **Escalabilidad:** Evaluar futuras ampliaciones del sistema, como el análisis avanzado de ventas o la integración con aplicaciones móviles.
- **Capacitación:** Capacitar al personal en el uso del sistema para maximizar su eficiencia y minimizar errores operativos.