
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE INGENIERÍA ELÉCTRICA

INGENIERÍA EN COMPUTACIÓN

# PROYECTO FINAL

**NOMBRES:**

ALVAREZ BADILLO RODRIGO

ANCIRA MOYA JESUS DOMINGO

CASTELAN RAMOS CARLOS

RAMOS QUIROZ ALEXIS

TERAN HERNANDEZ ALDO

**Nº de Cuenta:**

317282755

317242818

317042711

317329409

317226281

**GRUPO DE TEORÍA:** M.A. GABRIELA BETZABE LIZARRAGA RAMIREZ

**SEMESTRE 2024-1**

**Propuesta:**

Esta propuesta de base de datos ofrece una estructura completa para gestionar eficientemente su negocio. Con un enfoque centrado en la relación entre clientes, vendedores, productos, almacenes, ventas y carritos de compras, esta base de datos permite registrar y rastrear cada paso del proceso de ventas. Con atributos detallados para cada entidad, como ID, nombre, dirección, teléfono, correo electrónico, entre otros, y relaciones establecidas entre ellos, como la capacidad de un cliente para realizar múltiples compras o un vendedor para concretar varias ventas, aseguramos un seguimiento preciso y un control integral de inventario, ventas y transacciones. La relación entre productos y almacenes permite monitorear el stock y su ubicación, mientras que la conexión entre ventas y carritos de compras garantiza un seguimiento detallado de cada transacción. Esta estructura flexible y relacionada brinda una sólida base para administrar y optimizar sus operaciones comerciales de manera eficiente y efectiva.

**CRUD**

- Clientes
    - Create (Crear): Agregar nuevos clientes con sus atributos como nombre, dirección, teléfono, correo electrónico, etc.
    - Read (Leer): Ver la información de clientes existentes.
    - Update (Actualizar): Modificar la información de los clientes, como cambiar la dirección o actualizar el correo electrónico.
    - Delete (Eliminar): Eliminar registros de clientes que ya no están activos o necesarios en la base de datos.
- Vendedores
    - Create: Registrar nuevos vendedores con sus detalles como nombre, teléfono, correo electrónico, etc.
    - Read: Acceder a la información de vendedores existentes.
    - Update: Actualizar la información de los vendedores, por ejemplo, cambiar su número de teléfono.
    - Delete: Eliminar vendedores que ya no están asociados con la empresa.
- Productos
    - Create: Agregar nuevos productos con su nombre, descripción, precio, stock, etc.
    - Read: Ver la información detallada de los productos disponibles.
    - Update: Modificar detalles de productos como su descripción, precio o cantidad en stock.
    - Delete: Eliminar productos que ya no se comercializan o están obsoletos.
- Almacenes
    - Create: Registrar nuevos almacenes con detalles como nombre, dirección, teléfono, etc.
    - Read: Acceder a la información de los almacenes existentes.
    - Update: Actualizar la información de los almacenes, como cambiar la dirección o el número de teléfono.
    - Delete: Eliminar almacenes que ya no están en uso o son redundantes.
- Ventas

    - Create: Registrar nuevas ventas con detalles como fecha, monto total, método de pago, etc.
    - Read: Ver el historial de ventas con detalles específicos de cada transacción.
    - Update: Actualizar información asociada a una venta, como corregir el método de pago.
    - Delete: Eliminar registros de ventas incorrectas o duplicadas.
- Carritos de Compras
    - Create: Crear nuevos carritos de compras para clientes con detalles como fecha de creación, estado, etc.
    - Read: Acceder a la información de los carritos de compras existentes.
    - Update: Modificar el estado de un carrito de compras (por ejemplo, de abierto a cerrado) o actualizar detalles relevantes.
    - Delete: Eliminar carritos de compras que ya no son necesarios o están vacíos.

**BLOB**

Para la implantación de una BD con BLOB encontramos datos binarios como imágenes y documentos, es entonces que planteamos el BLOB de la siguiente manera:

Diseño de la Base de Datos Relacional:

- Tablas Estructuradas: Crea tablas separadas para cada entidad, como "Clientes," "Productos," "Ventas," "Empleados," "Proveedores," etc. Estas tablas deben contener columnas para datos estructurados como nombres, direcciones, fechas, precios, etc.
- Relaciones: Establece relaciones entre las tablas para representar cómo se relacionan las entidades. Por ejemplo, podrías tener una tabla de "Ventas" relacionada con "Clientes" y "Productos" para registrar qué productos compró un cliente en una venta específica.

Uso de BLOBs:

- Almacenar Imágenes de Productos: En la tabla de "Productos," agrega un campo BLOB para almacenar las imágenes de los productos. Esto te permitirá asociar imágenes con cada producto.
- Documentos de Ventas en Línea: Si necesitas almacenar documentos asociados con las ventas en línea, como facturas o recibos electrónicos, puedes utilizar BLOBs en la tabla de "Ventas" para guardar estos documentos en formato binario.
- Fotos de Empleados y Cajeros: Si deseas mantener fotos de empleados y cajeros, agrega campos BLOB a las tablas de "Empleados" y "Cajeros" para almacenar las imágenes de sus fotos de perfil.
- Documentos de Proveedores: Si necesitas almacenar documentos proporcionados por proveedores, como contratos o catálogos en formato PDF, puedes utilizar BLOBs en la tabla de "Proveedores" para mantener estos documentos.

Gestión de BLOBs:

- Utiliza las funciones y características proporcionadas por tu sistema de gestión de bases de datos específico para trabajar con BLOBs, como INSERTAR, ACTUALIZAR y RECUPERAR datos binarios.

Seguridad y Rendimiento:

- Asegurar implementar medidas de seguridad adecuadas para proteger los datos binarios almacenados, ya que estos pueden ser sensibles.

Copia de Seguridad:

- Realiza copias de seguridad regulares de la base de datos, incluyendo los datos binarios, para garantizar la disponibilidad y la integridad de estos archivos.

Indexación:

- Considera indexar adecuadamente las tablas y campos que contienen BLOBs para mejorar el rendimiento de las consultas.

**ACID**

La implementación del ACID en nuestra propuesta de proyecto contempla los siguientes puntos.

Atomicidad (Atomicity):

- Ventas y Actualizaciones de Inventario: Cada transacción de venta, ya sea en línea o física, se considera una unidad atómica. Si un cliente realiza una compra que involucra varios productos, todas las operaciones de actualización de inventario y registro de la venta deben completarse con éxito o revertirse en su totalidad en caso de fallo. Esto evita que se realicen ventas parciales o se dejen productos en un estado inconsistente.

Consistencia (Consistency):

- Mantenimiento de Reglas de Integridad: La base de datos debe mantener reglas de integridad para garantizar que los datos sean consistentes. Por ejemplo, la cantidad disponible de un producto nunca debe ser negativa, y los empleados no deben tener más roles de los permitidos. Cualquier operación que viole estas reglas debe revertirse para mantener la consistencia.

Aislamiento (Isolation):

- Transacciones Concurrentes: Dado que la tienda maneja ventas en línea y físicas, así como registros de empleados, es probable que varias transacciones se ejecuten simultáneamente. El aislamiento garantiza que cada transacción se ejecute en un entorno aislado y que los cambios realizados por una transacción no sean visibles para otras transacciones hasta que se confirmen. Esto evita problemas como la lectura sucia o la escritura conflictiva entre transacciones concurrentes.

Durabilidad (Durability):

- Registro de Transacciones y Copias de Seguridad: La base de datos debe mantener un registro de transacciones y realizar copias de seguridad regularmente. Esto garantiza que, una vez que una transacción se haya confirmado, los cambios sean permanentes y resistentes a fallos del sistema o pérdida de energía. Los datos deben poder recuperarse incluso después de un reinicio del sistema.

**ESQUEMA Y DIAGRAMA RELACIONAL**

Cliente
Atributos = {ID (pk), Nombre, Apellido, Dirección, Teléfono, Correo Electrónico}

Vendedor
Atributos = {ID (pk), Nombre, Apellido, Teléfono, Correo Electrónico}

Producto
Atributos = {ID (pk), Nombre, Descripción, Precio, Stock}

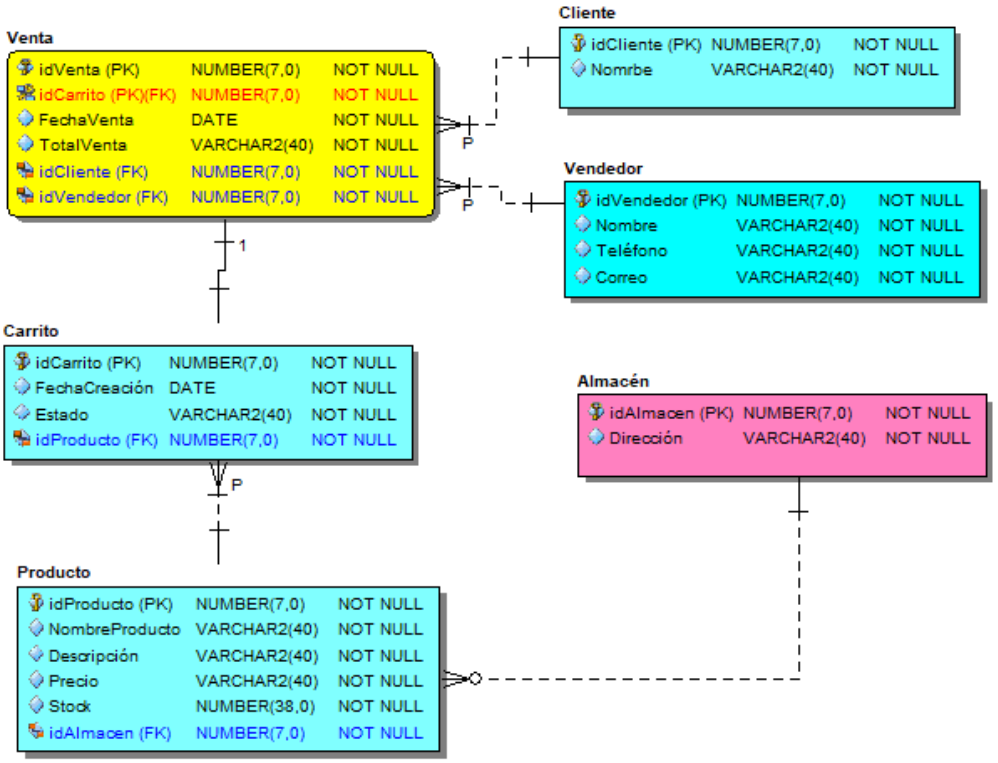
Almacén
Atributos = ID (pk), Nombre, Dirección, Teléfono}

Venta
Atributos = {ID (pk), Fecha de Venta, Total de Venta, Método de Pago}

Carrito de Compras
Atributos = {ID (Clave primaria), Fecha de Creación, Estado (por ejemplo, abierto o cerrado)}

Relaciones:

- Un cliente puede realizar muchas compras (uno a muchos) -> Relación entre Cliente y Venta.
- Un vendedor puede realizar muchas ventas (uno a muchos) -> Relación entre Vendedor y Venta.
- Un producto puede estar en muchos carritos de compras (uno a muchos) -> Relación entre Producto y Carrito de Compras.
- Un producto puede estar en muchos almacenes (uno a muchos) -> Relación entre Producto y Almacén.
- Un carrito de compras puede contener muchos productos (muchos a muchos) -> Relación entre Carrito de Compras y Producto.
- Una venta puede estar asociada a un solo carrito de compras (uno a uno) -> Relación entre Venta y Carrito de Compras.
- Un almacén puede tener muchos productos (uno a muchos) -> Relación entre Almacén y Producto.




**CÁLCULOS STORAGE**

Tabla VENTA

| Columna | Tipo | Longitud máxima con promedios nulos | Longitud + 1 byte de separación |
|---|---|---|---|
| idVenta | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| idCarrito | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| FechaVenta | DATE | 7 | 7+1=8 |
| TotalVenta | NUMERIC(2) | Ceil(2/2+1)=2 | 2+1=3 |
| idCliente | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| idVendedor | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| | | Total=7 | Total= 35+5=40 |

Tamaño de bloques = 2KB = 2048 BYTES -90 BYTES (VALOR CTE DEL HEADER) = 1958 BYTES

Si (100 registros * 40) / 1958 bytes = 2.04 = 3 bloques * 2 KB = 6 KB = 0.006 MB ß INITIAL

PCTFREE

Como 40 es el 100% entonces 22 es ¿?

Solución:

X=(7*100)/40 = 17.5%

X=18% de PCTFREE

PCTUSED

Si 1958 es el 100% 40 es ¿?

Solución:

X=(40*100)/1958=2.04=2.5%

X=2.5%(% que ocupa cada registro)

MAXTRANS

1958/40=48.95 = 458 registros.

Tabla Carrito

| Columna | Tipo | Longitud máxima con promedios nulos | Longitud + 1 byte de separación |
|---|---|---|---|
| idCarrito | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| FechaCreación | DATE | 7 | 7+1=8 |
| Estado | VARCHAR(40) | 34+6=40 | 40+1=41 |
| idProducto | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| | | Total=6 | Total= 61+5=66 |

Tamaño de bloques = 2KB = 2048 BYTES -90 BYTES (VALOR CTE DEL HEADER) = 1958 BYTES

Si (100 registros * 66) / 1958 bytes = 3.37 = 4 bloques * 2 KB = 8 KB = 0.008 MB ß INITIAL

PCTFREE

Como 66 es el 100% entonces 5 es ¿?

Solución:

X=(6*100)/66 = 9.09%

X=10% de PCTFREE

PCTUSED

Si 1958 es el 100% 66 es ¿?

Solución:

X=(66*100)/1958=3.37=4%

X=4% (% que ocupa cada registro)

MAXTRANS

1958/66=29.66 = 30 registros.

Tabla Producto

| Columna | Tipo | Longitud máxima con promedios nulos | Longitud + 1 byte de separación |
|---|---|---|---|
| idProducot | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| NombreProducto | VARCHAR(40) | 34+6=40 | 40+1=41 |
| Descripción | VARCHAR(40) | 34+6=40 | 40+1=41 |
| Precio | NUMERIC(2) | Ceil(2/2+1)=2 | 2+1=3 |
| Stock | VARCHAR(40) | 34+6=40 | 40+1=41 |
| idAlmacen | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| | | Total=18 | Total= 138 |

Tamaño de bloques  = 2KB = 2048 BYTES -90 BYTES (VALOR CTE DEL HEADER) = 1958 BYTES

Si (100 registros * 138) / 1958 bytes = 7.04 = 8 bloques * 2 KB = 16 KB = 0.016 MB ß INITIAL

PCTFREE

Como 138 es el 100% entonces 18 es ¿?

Solución:

X=(18*100)/138 = 13.04%

X=13.5% de PCTFREE

PCTUSED

Si 1958 es el 100% 138 es ¿?

Solución:

X=(138*100)/1958=7.04=7.5%

X=7.5% (% que ocupa cada registro)

MAXTRANS

1958/138=14.18 = 15 registros.

Tabla Cliente

| Columna | Tipo | Longitud máxima con promedios nulos | Longitud + 1 byte de separación |
|---|---|---|---|
| idCliente | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |

| | | | |
|---|---|---|---|
| Nombre | VARCHAR(40) | 34+6=40 | 40+1=41 |
| | | Total=6 | Total= 47 |

Tamaño de bloques  = 2KB = 2048 BYTES -90 BYTES (VALOR CTE DEL HEADER) = 1958 BYTES

Si (100 registros * 47) / 1958 bytes = 2.4 = 3 bloques * 2 KB = 6 KB = 0.006 MB ß INITIAL

PCTFREE

Como 47 es el 100% entonces 18 es ¿?

Solución:

X=(6*100)/47 = 12.76%

X=13% de PCTFREE

PCTUSED

Si 1958 es el 100% 47 es ¿?

Solución:

X=(47*100)/1958=2.4=3%

X=3% (% que ocupa cada registro)

MAXTRANS

1958/47=41.65 = 42 registros.

Tabla Vendedor

| Columna | Tipo | Longitud máxima con promedios nulos | Longitud + 1 byte de separación |
|---|---|---|---|
| idVendedor | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| Nombre | VARCHAR(40) | 34+6=40 | 40+1=41 |
| Teléfono | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| correo | VARCHAR(40) | 34+6=40 | 40+1=41 |
| | | Total=12 | Total= 94 |

Tamaño de bloques = 2KB = 2048 BYTES -90 BYTES (VALOR CTE DEL HEADER) = 1958 BYTES

Si (100 registros * 94) / 1958 bytes = 4.80 = 5 bloques * 2 KB = 10 KB = 0.010 MB ß INITIAL

PCTFREE

Como 94 es el 100% entonces 18 es ¿?

Solución:

X=(18*100)/94 = 19.14%

X=2% de PCTFREE

PCTUSED

Si 1958 es el 100% 94 es ¿?

Solución:

X=(94*100)/1958=4.8=5%

X=5% (% que ocupa cada registro)

MAXTRANS

1958/84=23.3 = 24 registros.

| Columna | Tipo | Longitud máxima con promedios nulos | Longitud + 1 byte de separación |
|---|---|---|---|
| idAlmacen | NUMERIC(7) | Ceil(7/2+1)=5 | 5+1=6 |
| Dirección | VARCHAR(40) | 34+6=40 | 40+1=41 |
| | | Total=6 | Total= 47 |

Tamaño de bloques = 2KB = 2048 BYTES -90 BYTES (VALOR CTE DEL HEADER) = 1958 BYTES

Si (100 registros * 47) / 1958 bytes = 2.4 = 3 bloques * 2 KB = 6 KB = 0.006 MB ß INITIAL

PCTFREE

Como 47 es el 100% entonces 6 es ¿?

Solución:

X=(6*100)/47 = 12.76%

X=13% de PCTFREE

PCTUSED

Si 1958 es el 100% 47 es ¿?

Solución:

X=(47*100)/1958=2.4=3%

X=3% (% que ocupa cada registro)

MAXTRANS

1958/47=41.65 = 52 registros.

**SCRIPT STORAGE**

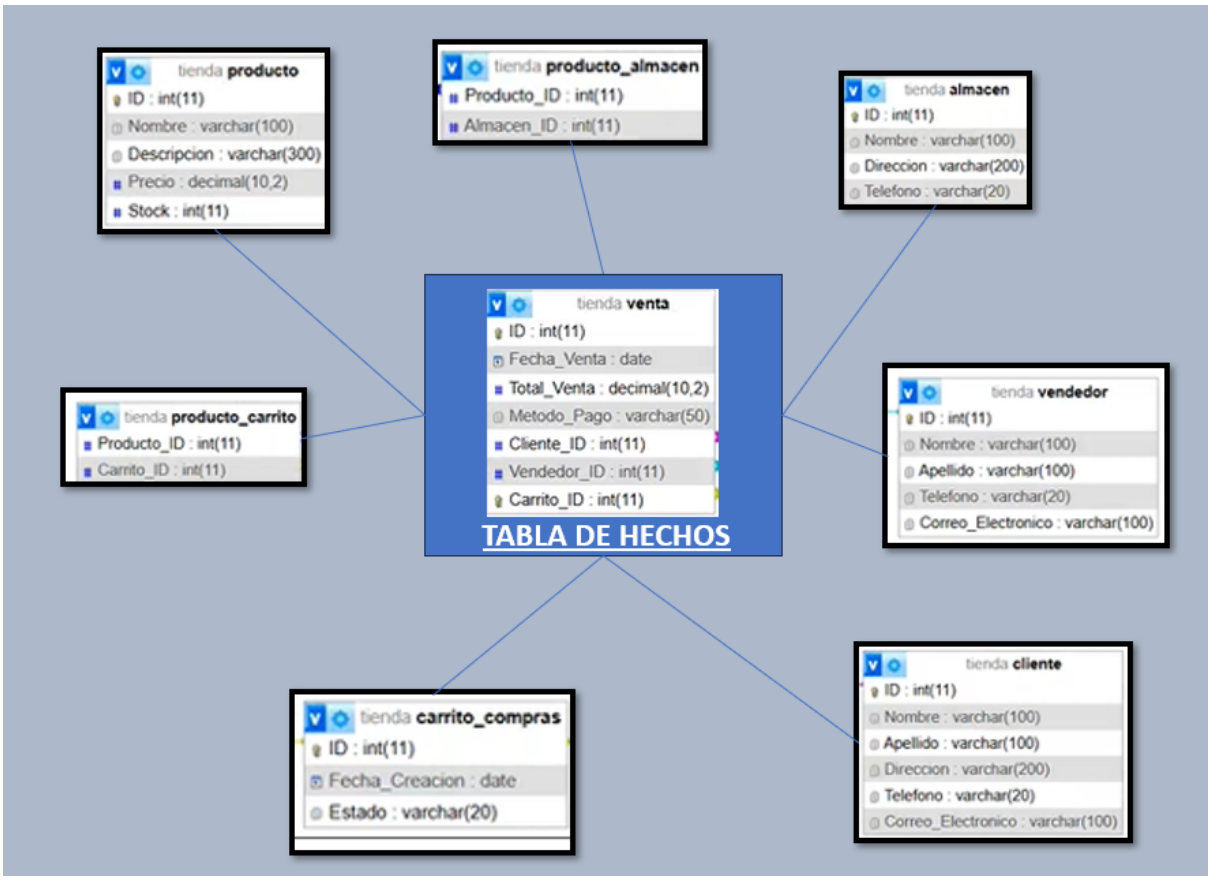
```
CREATE TABLE Venta(
    idVenta       NUMBER(7, 0)    NOT NULL,
    idCarrito     NUMBER(7, 0)    NOT NULL,
    FechaVenta    DATE            NOT NULL,
    TotalVenta    VARCHAR2(40)    NOT NULL,
    idCliente     NUMBER(7, 0)    NOT NULL,
    idVendedor    NUMBER(7, 0)    NOT NULL,
    CONSTRAINT PK6 PRIMARY KEY (idVenta, idCarrito)
    USING INDEX
        LOGGING,
    CONSTRAINT RefCliente8 FOREIGN KEY (idCliente)
    REFERENCES Cliente(idCliente),
    CONSTRAINT RefVendedor10 FOREIGN KEY (idVendedor)
    REFERENCES Vendedor(idVendedor),
    CONSTRAINT RefCarrito18 FOREIGN KEY (idCarrito)
    REFERENCES Carrito(idCarrito)
)
PCTFREE 18
PCTUSED 3
INITRANS 1
MAXTRANS 458
LOGGING
STORAGE(INITIAL 6K
        NEXT 6K
        MINEXTENTS 1
        PCTINCREASE 50
        )
;

CREATE TABLE Carrito(
    idCarrito        NUMBER(7, 0)    NOT NULL,
    FechaCreación    DATE            NOT NULL,
    Estado           VARCHAR2(40)    NOT NULL,
    idProducto       NUMBER(7, 0)    NOT NULL,
    CONSTRAINT pkCarrito PRIMARY KEY (idCarrito)
    USING INDEX
        LOGGING,
```

```
    CONSTRAINT RefProducto17 FOREIGN KEY (idProducto)
    REFERENCES Producto(idProducto)
)
PCTFREE 10
PCTUSED 4
INITRANS 1
MAXTRANS 30
LOGGING
STORAGE(INITIAL 8K
        NEXT 8K
        MINEXTENTS 1
        PCTINCREASE 50
        )
;

CREATE TABLE Producto(
    idProducto        NUMBER(7, 0)     NOT NULL,
    NombreProducto    VARCHAR2(40)     NOT NULL,
    Descripción       VARCHAR2(40)     NOT NULL,
    Precio            VARCHAR2(40)     NOT NULL,
    Stock             NUMBER(38, 0)    NOT NULL,
    idAlmacen         NUMBER(7, 0)     NOT NULL,
    CONSTRAINT pkProducto PRIMARY KEY (idProducto)
    USING INDEX
        LOGGING,
    CONSTRAINT RefAlmacén1 FOREIGN KEY (idAlmacen)
    REFERENCES Almacén(idAlmacen)
)
PCTFREE 14
PCTUSED 8
INITRANS 1
MAXTRANS 15
LOGGING
STORAGE(INITIAL 16K
        NEXT 16K
        MINEXTENTS 1
        PCTINCREASE 50
        )
;

CREATE TABLE Cliente(
    idCliente    NUMBER(7, 0)    NOT NULL,
    Nomrbe       VARCHAR2(40)    NOT NULL,
    CONSTRAINT pkCliente PRIMARY KEY (idCliente)
    USING INDEX
        LOGGING
)
PCTFREE 13
```

```
PCTUSED 3
INITRANS 1
MAXTRANS 42
LOGGING
STORAGE(INITIAL 6K
        NEXT 6K
        MINEXTENTS 1
        PCTINCREASE 50
        )
;

CREATE TABLE Vendedor(
    idVendedor    NUMBER(7, 0)    NOT NULL,
    Nombre        VARCHAR2(40)    NOT NULL,
    Teléfono      VARCHAR2(40)    NOT NULL,
    Correo        VARCHAR2(40)    NOT NULL,
    CONSTRAINT pkVendedor PRIMARY KEY (idVendedor)
    USING INDEX
        LOGGING
)
PCTFREE 2
PCTUSED 5
INITRANS 1
MAXTRANS 24
LOGGING
STORAGE(INITIAL 10K
        NEXT 10K
        MINEXTENTS 1
        PCTINCREASE 50
        )
;


CREATE TABLE Almacén(
    idAlmacen    NUMBER(7, 0)    NOT NULL,
    Dirección    VARCHAR2(40)    NOT NULL,
    CONSTRAINT PK4 PRIMARY KEY (idAlmacen)
    USING INDEX
        LOGGING
)
PCTFREE 13
PCTUSED 3
INITRANS 1
MAXTRANS 52
LOGGING
STORAGE(INITIAL 6K
        NEXT 6K
        MINEXTENTS 1
```

```
        PCTINCREASE 50
        )
;
```

**DATAMART**




Un datamart es una versión simplificada de un almacén de datos (data warehouse) que se enfoca en un solo tema o área de negocio. En este caso, el tema es el proceso de ventas de una tienda, que incluye productos, carritos de compras, ventas, almacenes, vendedores y clientes. Aquí hay un resumen de cada componente y cómo podrían interactuar en este datamart:

**Tabla de Hechos: tienda_venta**

ID: Un identificador único para cada venta.

Fecha_Venta: La fecha en que se realizó la venta.

Total_Venta: El monto total de la venta.

Metodo_Pago: El método de pago utilizado en la venta.

Cliente_ID: La referencia al cliente que realizó la compra.

Vendedor_ID: La referencia al vendedor que manejó la venta.

Carrito_ID: La referencia al carrito de compras que se convirtió en la venta.

Esta tabla es el centro del esquema y es donde se registran las transacciones de ventas. Las demás tablas están relacionadas con esta tabla principal y proporcionan detalles adicionales sobre cada aspecto de la venta.

**Tablas Dimensionales**

Estas tablas contienen atributos contextuales relacionados con las dimensiones de la tabla de hechos.

**tienda_producto**

ID: El identificador único del producto.

Nombre: El nombre del producto.

Descripcion: Una descripción del producto.

Precio: El precio del producto.

Stock: La cantidad de producto disponible.

**tienda_almacen**

ID: Identificador único del almacén.

Nombre: Nombre del almacén.

Direccion: Dirección física del almacén.

Telefono: Número de teléfono del almacén.

**tienda_vendedor**

ID: Identificador único del vendedor.

Nombre: Nombre del vendedor.

Apellido: Apellido del vendedor.

Telefono: Número de teléfono del vendedor.

Correo_Electronico: Dirección de correo electrónico del vendedor.

**tienda_cliente**

ID: Identificador único del cliente.

Nombre: Nombre del cliente.

Apellido: Apellido del cliente.

Direccion: Dirección física del cliente.

Telefono: Número de teléfono del cliente.

Correo_Electronico: Dirección de correo electrónico del cliente.

**tienda_producto_carrito**

Producto_ID: Referencia al producto añadido al carrito.

Carrito_ID: Referencia al carrito de compras donde se añade el producto.

**tienda_carrito_compras**

ID: Identificador único del carrito de compras.

Fecha_Creacion: Fecha en que se creó el carrito.

Estado: Estado del carrito (por ejemplo, activo, pendiente, completado).

**tienda_producto_almacen**

Producto_ID: Referencia al producto almacenado.

Almacen_ID: Referencia al almacén donde se guarda el producto.

Cada tabla dimensional está relacionada con la tabla de hechos a través de claves foráneas que coinciden con las claves primarias de las tablas dimensionales. Por ejemplo, Cliente_ID en la tabla de hechos tienda_venta se vincula con ID en la tabla tienda_cliente.

Este datamart permite realizar análisis enfocados en el área de ventas, cómo rastrear las ventas a lo largo del tiempo, analizar el rendimiento del vendedor, entender las preferencias del cliente, y administrar el inventario y la logística del almacén. Con esta estructura, se pueden generar informes y visualizaciones para soportar la toma de decisiones empresariales.

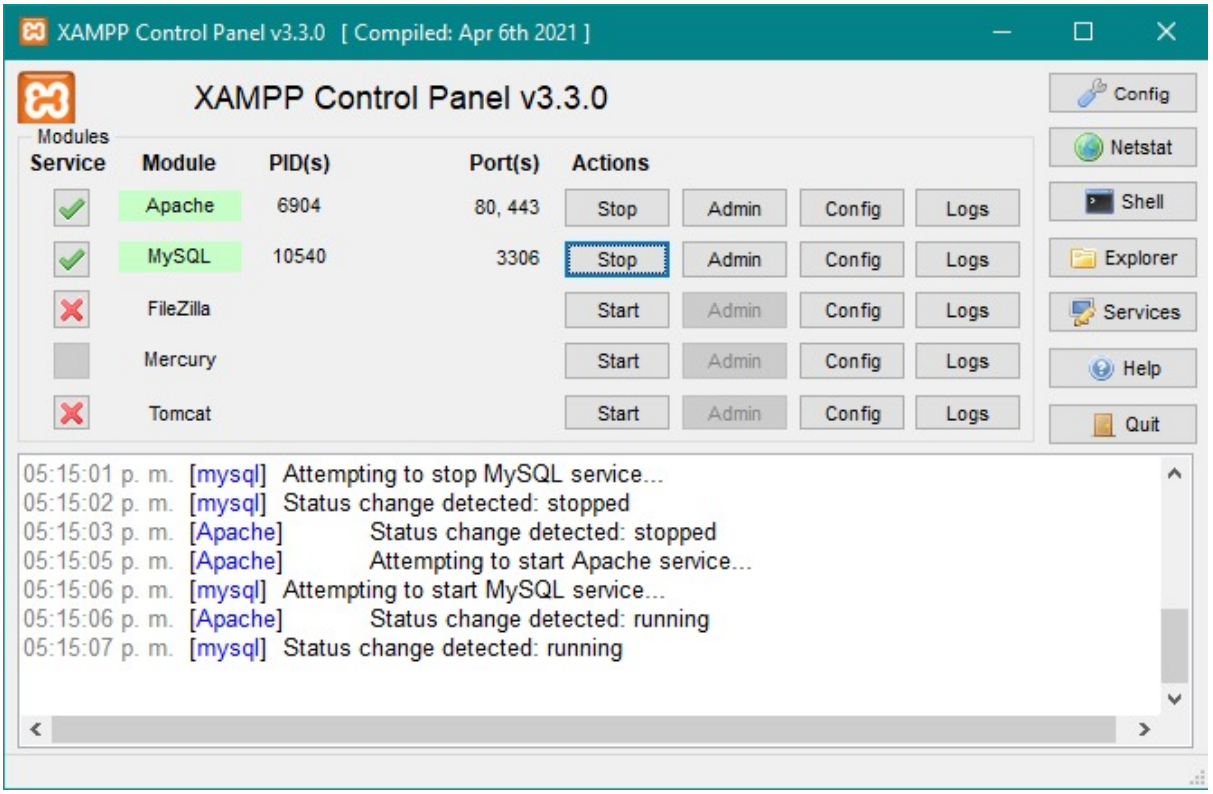
## IMPLEMENTACIÓN MYSQL-APACHE EN PHPMYADMIN

Para realizar la implementación de nuestra base de datos en una página web se hizo uso del framework Flask, siendo un ligero y flexible para Python que facilita la creación de aplicaciones web. Es conocido por su simplicidad y su enfoque minimalista, lo que significa que proporciona las herramientas necesarias para construir aplicaciones web de manera rápida y sencilla, sin imponer una estructura rígida.
Por otro lado, el sistema manejador de base de datos que elegimos es MySQL, ya que Flask proporciona extensiones como Flask-MySQL que permiten conectarse a una base de datos MySQL de manera sencilla. Estas extensiones facilitan la interacción con la base de

datos mediante consultas SQL desde la aplicación Flask. Además puede trabajar con ORM como SQLAlchemy, que proporciona una capa de abstracción sobre la base de datos.
Otra herramienta de la que hicimos uso fue phpMyAdmin que nos sirvió para la administración de bases de datos MySQL que se ejecuta a través de un navegador web. Esta aplicación está escrita en PHP y ofrece una interfaz gráfica para interactuar con tus bases de datos MySQL de una manera amigable y visual.

Finalmente para tener todas las instalaciones de forma correcta se hizo uso de XAMPP, el cual es un paquete de software gratuito y de código abierto que facilita la creación de entornos de desarrollo web local. Este proporciona una manera fácil de instalar y configurar un entorno de servidor web local que incluye estos componentes esenciales para el desarrollo web. Esto permite a los desarrolladores trabajar en aplicaciones web sin necesidad de una conexión a Internet, ya que pueden simular un entorno de servidor en sus propias computadoras.

Teniendo todo esto listo, se habilitó el uso Apache y MySQL en XAMPP:



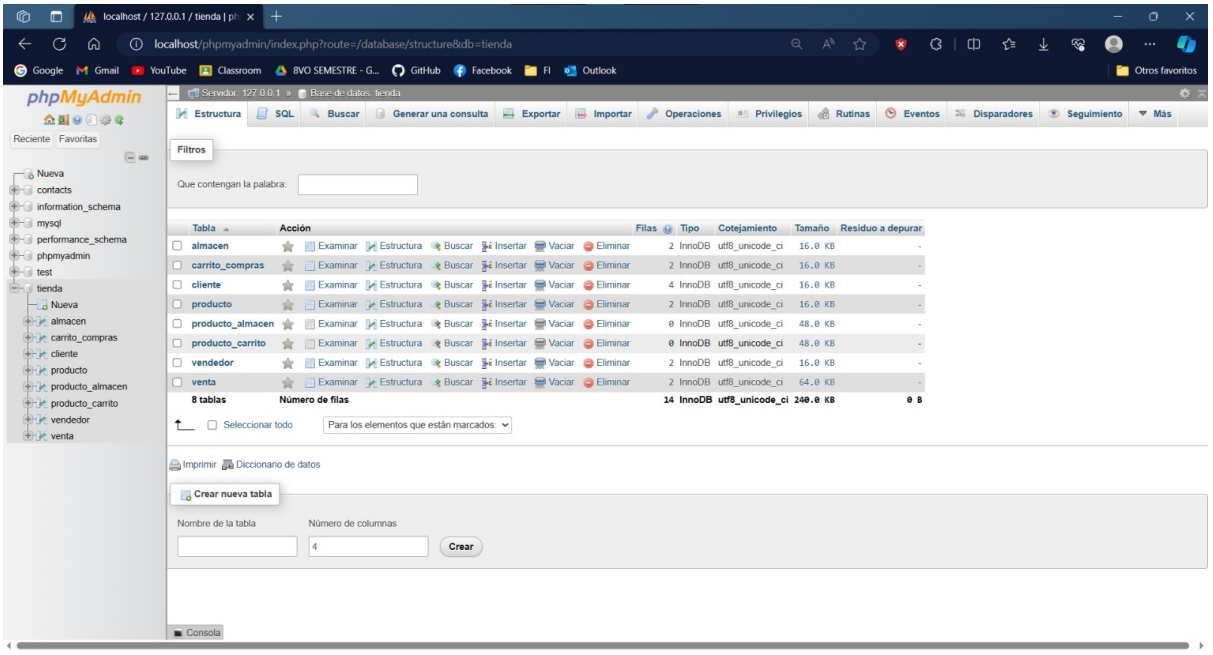

*Configuración XAMPP*

Una vez levantados los servicios se inicia phpMyAdmin y se crea la base de datos tienda. Para crear la base de datos, fue necesario hacerlo con la sintaxis de MySQL, una vez teniendo listo Script se importó desde phpMyAdmin:

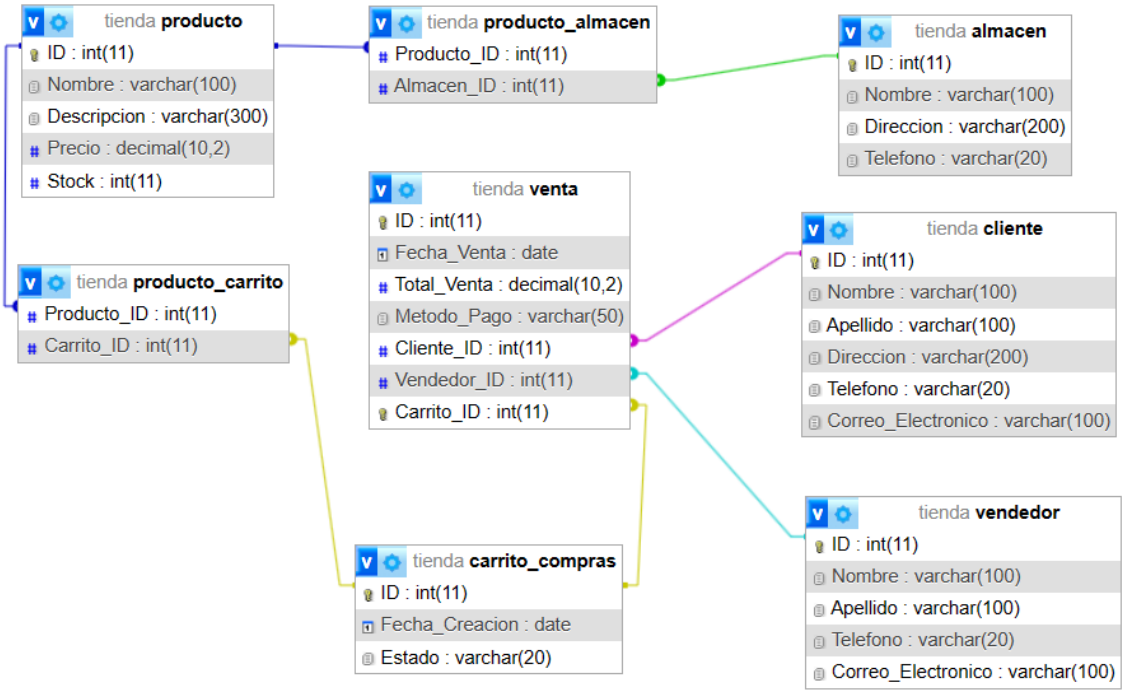
```
-- Tabla Cliente
CREATE TABLE Cliente (
    ID INT PRIMARY KEY,
    Nombre VARCHAR(100),
    Apellido VARCHAR(100),
    Direccion VARCHAR(200),
```

```
    Telefono VARCHAR(20),
    Correo_Electronico VARCHAR(100)
);

-- Tabla Vendedor
CREATE TABLE Vendedor (
    ID INT PRIMARY KEY,
    Nombre VARCHAR(100),
    Apellido VARCHAR(100),
    Telefono VARCHAR(20),
    Correo_Electronico VARCHAR(100)
);

-- Tabla Producto
CREATE TABLE Producto (
    ID INT PRIMARY KEY,
    Nombre VARCHAR(100),
    Descripcion VARCHAR(300),
    Precio DECIMAL(10,2),
    Stock INT
);

-- Tabla Almacén
CREATE TABLE Almacen (
    ID INT PRIMARY KEY,
    Nombre VARCHAR(100),
    Direccion VARCHAR(200),
    Telefono VARCHAR(20)
);

-- Tabla Venta
CREATE TABLE Venta (
    ID INT PRIMARY KEY,
    Fecha_Venta DATE,
    Total_Venta DECIMAL(10,2),
    Metodo_Pago VARCHAR(50),
    Cliente_ID INT,
    Vendedor_ID INT,
       CONSTRAINT fk_cliente FOREIGN KEY (Cliente_ID) REFERENCES
Cliente(ID),
      CONSTRAINT fk_vendedor FOREIGN KEY (Vendedor_ID) REFERENCES
Vendedor(ID)
);

-- Tabla Carrito de Compras
CREATE TABLE Carrito_Compras (
    ID INT PRIMARY KEY,
    Fecha_Creacion DATE,
```

```
    Estado VARCHAR(20)
);

-- Relaciones Many-to-Many (Producto - Carrito de Compras) y
(Producto - Almacén)
-- Creación de tablas intermedias

-- Producto en Carrito de Compras
CREATE TABLE Producto_Carrito (
    Producto_ID INT,
    Carrito_ID INT,
          CONSTRAINT fk_producto_carrito_producto FOREIGN KEY
(Producto_ID) REFERENCES Producto(ID),
           CONSTRAINT fk_producto_carrito_carrito FOREIGN KEY
(Carrito_ID) REFERENCES Carrito_Compras(ID)
);

-- Producto en Almacén
CREATE TABLE Producto_Almacen (
    Producto_ID INT,
    Almacen_ID INT,
          CONSTRAINT fk_producto_almacen_producto FOREIGN KEY
(Producto_ID) REFERENCES Producto(ID),
           CONSTRAINT fk_producto_almacen_almacen FOREIGN KEY
(Almacen_ID) REFERENCES Almacen(ID)
);

-- Relación Uno a Uno (Venta - Carrito de Compras)
ALTER TABLE Venta ADD (
    Carrito_ID INT UNIQUE,
         CONSTRAINT fk_carrito_venta FOREIGN KEY (Carrito_ID)
REFERENCES Carrito_Compras(ID)
);
```




*Base de datos tienda en phpMyAdmin*




*Modelo relacional de la Base de Datos Tienda*

## CRUD EN PÁGINA WEB - FLASK

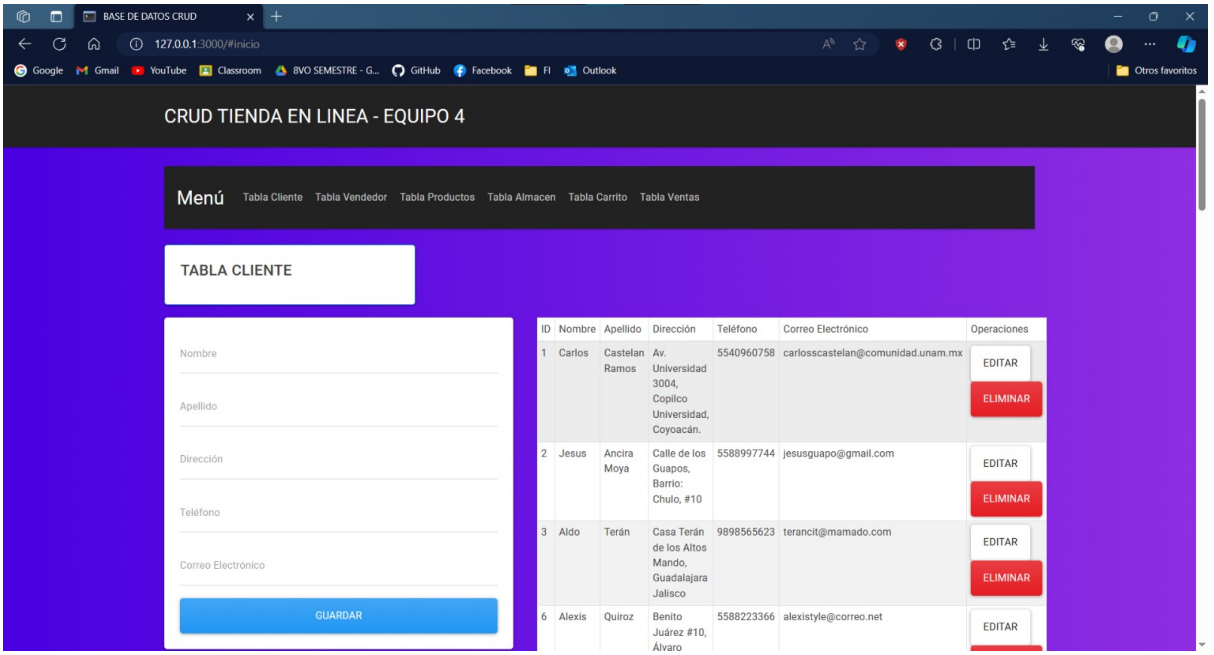
Para la implementación de CRUD en cada una de las tablas de la base de datos se realizó su código de python el cuál cada uno implementa 5 funciones principales:

- index: Es una función que manda a llamar una consulta de todos los elementos que se encuentran en la tabla dentro del index principal de HTML.Esta función es la implementación de READ en la página web.

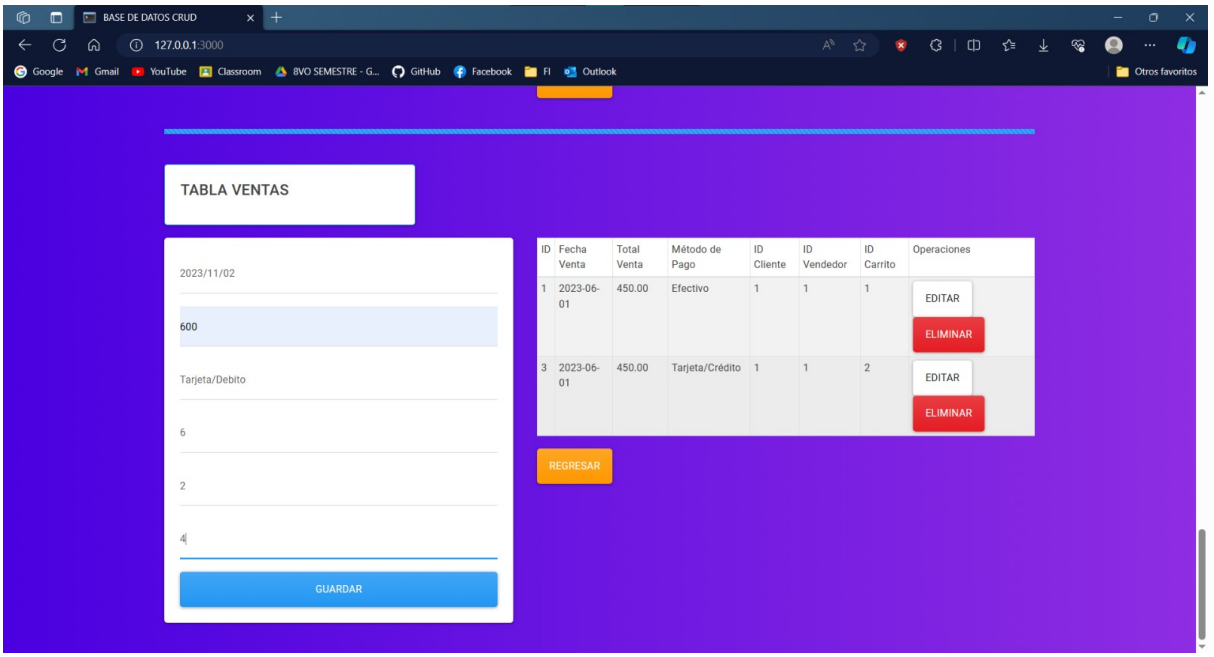
- add: Es una función que añade registros a la tabla que se haya seleccionado. Esta función es la implementación de CREATE en la página web.
- update: Es una función que al seleccionar algún registro de la tabla desplegada en index permite actualizar los valores del registro según el atributo elegido. Esta función es la implementación de UPDATE en la página web.
- get: Esta es una función auxiliar para Update, que al igual que index, obtiene una vista de la consulta de la tabla elegida a actualizar.
- delete: Esta función accede a la tabla elegida y elimina el elemento seleccionado. Esta función es la implementación de DELETE en la página web.

Finalmente estas funciones son llamadas cada que se solicite su uso en el HTML principal, por otro lado existen otros HTML auxiliares, por ejemplo para actualizar las tablas. Además existen códigos en python las cuales inician el servidor, importan las funciones anteriores y la que carga la base de datos la cual es la siguiente sintáxis:

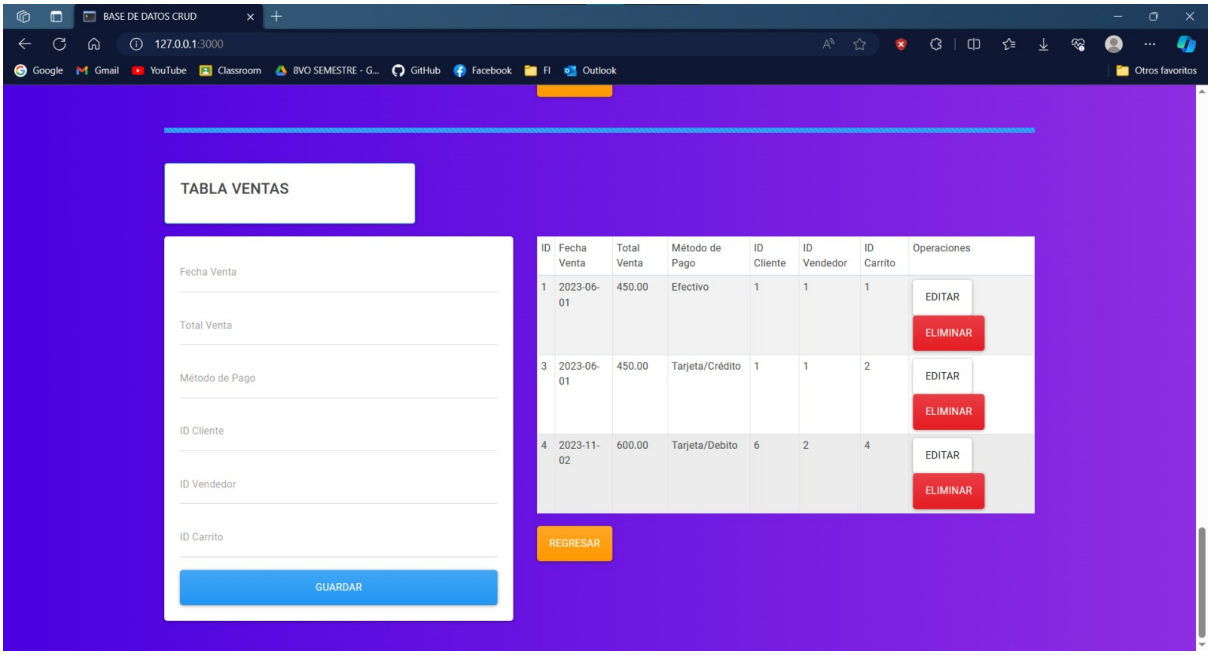
```
from app import app
from flask_mysqldb import MySQL
from dotenv import load_dotenv
import os

load_dotenv()  # take environment variables from .env.

# Mysql Settings
app.config['MYSQL_USER'] = os.getenv('MYSQL_USER') or 'root'
app.config['MYSQL_PASSWORD'] = os.getenv('MYSQL_PASSWORD') or
''
app.config['MYSQL_HOST'] = os.getenv('MYSQL_HOST') or
'127.0.0.1' # localhost
app.config['MYSQL_DB'] = os.getenv('MYSQL_DB') or 'tienda'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'

# MySQL Connection
mysql = MySQL(app)
```



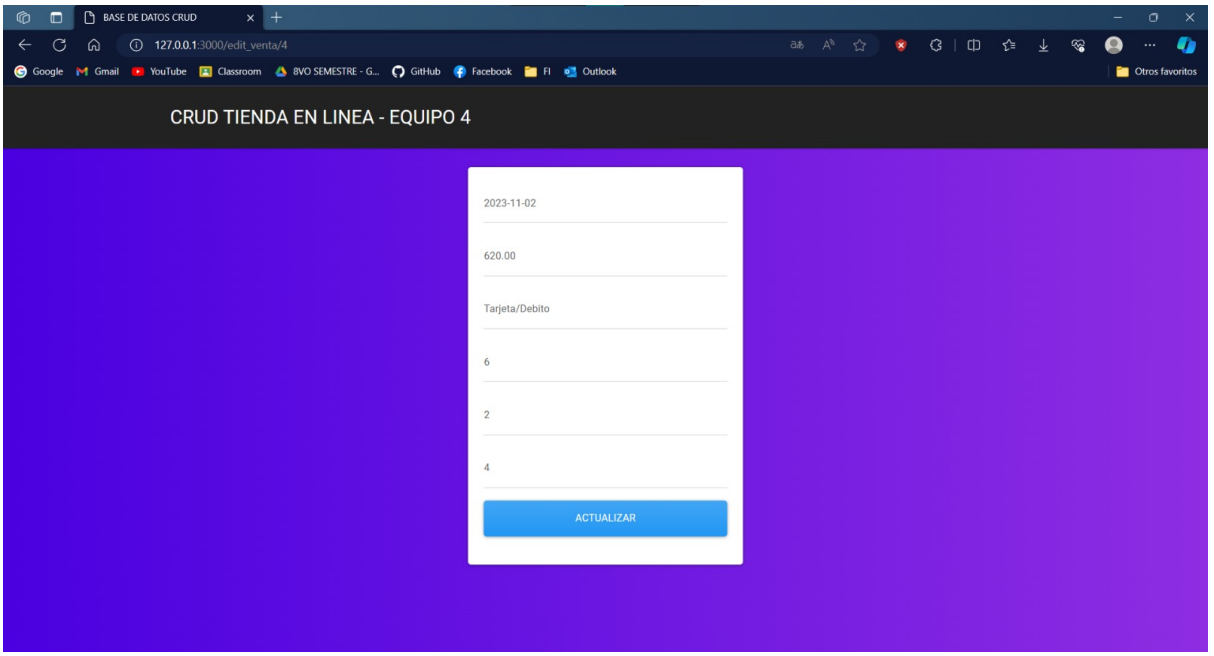

*Página principal.*



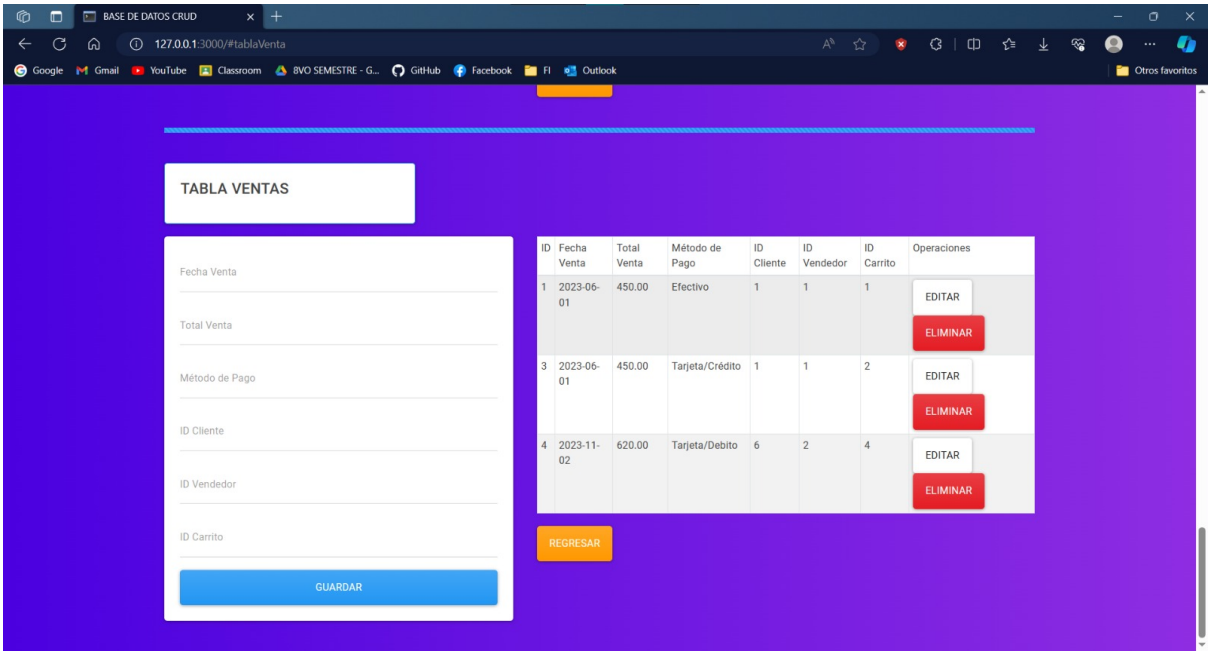

*Crear registro 1*



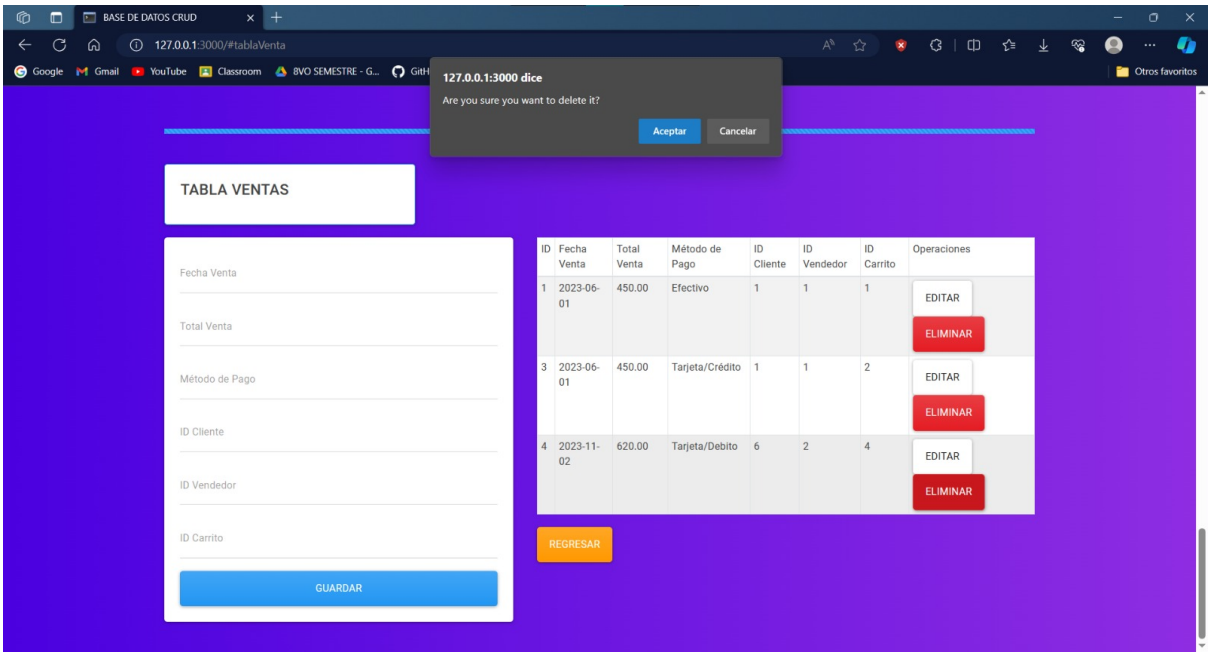

*Crear registro 2*



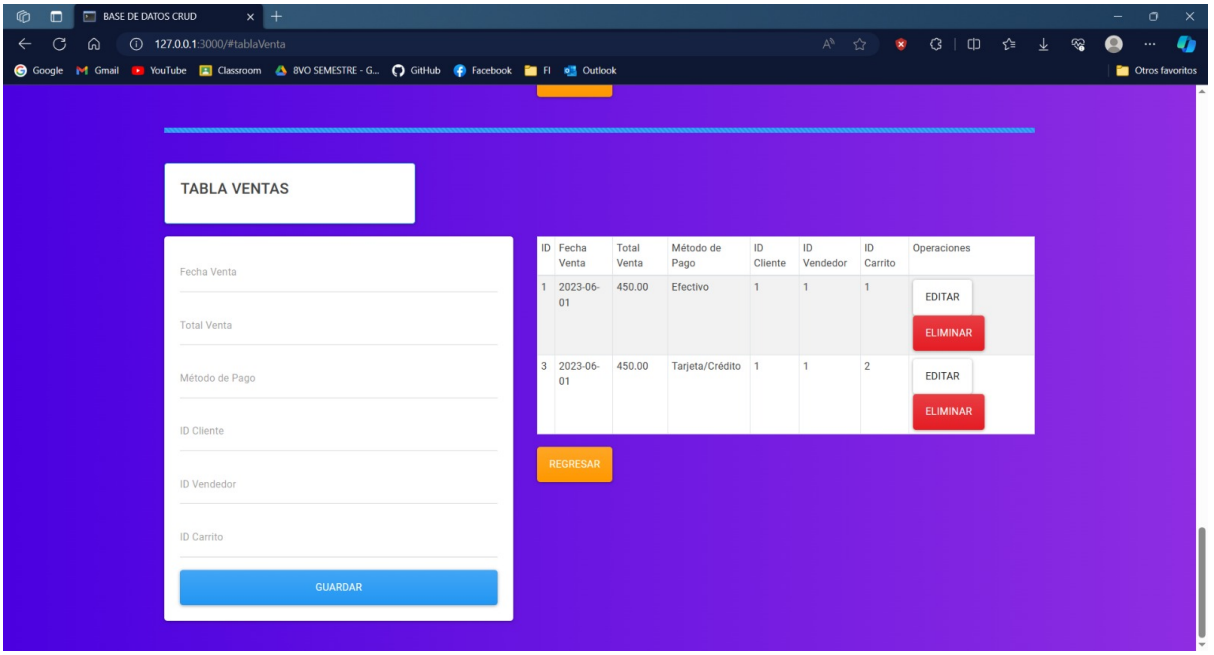

*Actualizar registro.*




*Registro actualizado.*




*Eliminar registro.*




*Registro eliminado.*

**Repositorio del proyecto:**
https://github.com/CarlosCR07/BD-Avanzadas-Proyecto-Final/tree/main

**Conclusión**

El proyecto final, centrado en la aplicación práctica de bases de datos avanzadas con operaciones CRUD en una página web, ha integrado con éxito Python, Flask y PHPMyAdmin para crear una aplicación web funcional y eficiente. Las tecnologías se combinaron para facilitar la creación de la interfaz de usuario y la gestión de la base de datos MySQL, destacando la importancia de un buen diseño de base de datos.

El proyecto proporcionó aprendizajes valiosos en la interacción entre tecnologías de backend y frontend, y el desarrollo de habilidades en programación de bases de datos y diseño web. El proyecto sugiere futuras mejoras en seguridad y experiencia del usuario, y considera la incorporación de nuevas tecnologías. En resumen, el proyecto demostró la importancia de la integración tecnológica y sentó una base sólida para futuros desarrollos en el campo de las bases de datos y el desarrollo web.

**Referencias:**


- Combaudon, S. (2018). *MySQL 5.7: administración y optimización*. Ediciones Eni.
- Kofler, M. (2001). What Is MySQL?. In *The Definitive Guide to MySQL* (pp. 3-19). Berkeley, CA: Apress.
- McHugh, J., Abiteboul, S., Goldman, R., Quass, D., & Widom, J. (1997). Lore: A database management system for semistructured data. *ACM Sigmod Record*, *26*(3), 54-66.
- Mullins, C. (2002). *Database administration: the complete guide to practices and procedures*. Addison-Wesley Professional.