MANUAL TECNICO

Proyecto 1:

Índice

Introducción

Este manual técnico proporciona una guía detallada para el diseño e implementación de una base de datos y una API REST para gestionar un sistema de ventas de un pequeño emprendimiento. La migración de un sistema de almacenamiento de datos basado en archivos Excel a un entorno digitalizado y eficiente es fundamental para mejorar la gestión de la información empresarial.

En respuesta a las necesidades específicas del cliente, se ha desarrollado un proceso completo que abarca desde el diseño conceptual hasta la implementación funcional, cubriendo aspectos clave como el modelado de la base de datos, la carga de datos, la generación de informes y la interacción a través de una API REST.

Este manual proporciona a los desarrolladores y administradores de sistemas una referencia completa para implementar y mantener el sistema de gestión de ventas, garantizando una transición exitosa hacia un entorno digitalizado y eficiente.

Objetivos

Proporcionar instrucciones detalladas sobre cómo instalar y configurar la API en el entorno de desarrollo o producción, incluyendo los requisitos del sistema, la instalación de dependencias y la configuración de la base de datos.

Describir exhaustivamente cada uno de los endpoints de la API, incluyendo su propósito, los parámetros que acepta (si los hay), los métodos HTTP admitidos y los posibles códigos de estado de respuesta. Esto ayudará a los desarrolladores a comprender cómo interactuar con la API y cómo utilizarla correctamente.

Detallar cómo realizar consultas a través de la API para obtener los diferentes reportes requeridos, incluyendo ejemplos de solicitud y respuesta. Además, proporcionar información sobre cómo manejar y diagnosticar errores que puedan surgir durante el uso de la API, como errores de autenticación, errores de validación de datos o errores del servidor.

Información destacada

El manual técnico hace referencia a la información necesaria con el fin de orientar al personal en la concepción, planteamiento análisis programación e instalación del sistema. Es de notar que la redacción propia del manual técnico está orientada a personal con conocimientos en sistemas y tecnologías de información, conocimientos de programación avanzada sobre Python, Flask y bases de datos (MySQL), responsables del mantenimiento e instalación del programa computacional en el computador.

Conocimientos Previos

Los conocimientos mínimos que deben tener las personas que operarán las páginas y deberán utilizar este manual son:

- Conocimientos sobre los conceptos de modelado de bases de datos, incluyendo modelos conceptuales, lógicos y relacionales.
- Conocimientos sobre SQL para comprender las consultas efectivas en la base de datos.
- Tener conocimientos de desarrollo web y APIs REST

Requerimientos

El sistema puede ser instalado en cualquier sistema operativo que cumpla con los siguientes requerimientos:

- Sistema Operativo: Windows 7 o superior
- Procesador mínimo Intel Pentium (800MHz Intel Pentium)
- Mínimo 1GB en RAM
- MySQL Workbench
- IDE Visual Studio Code, o compatible con Python
- Postman
- Exploradores: Internet Explorer 9 y superior

Instalación y configuración

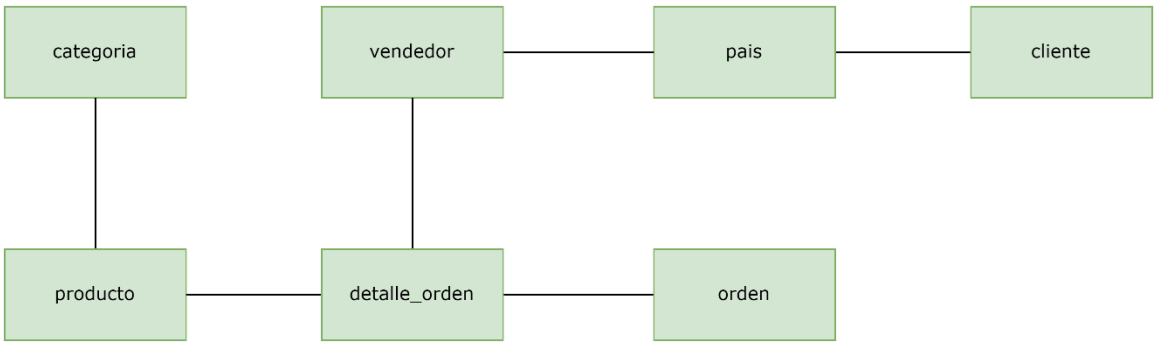
Para el proceso de instalación de esta aplicación únicamente es necesario tener MySQL, Postman y un IDE que sea compatible con el lenguaje Python para ejecutar la aplicación desde la terminal de este.

No es necesario tener alguna configuración ya que la configuración que trae por determinado el IDE es la necesaria para que el funcionamiento del programa sea posible.

Diseño de la Base de Datos:

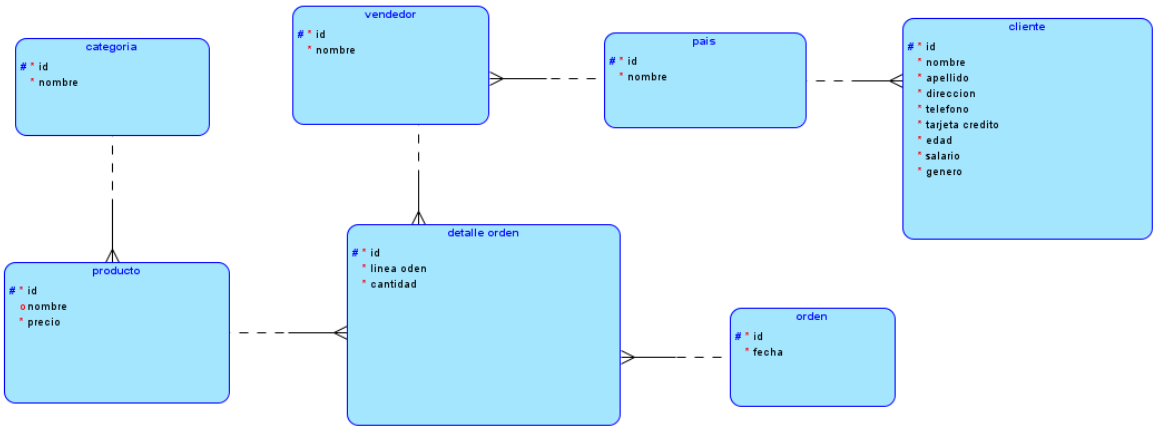
Modelo Conceptual:

El modelo conceptual define las entidades principales y las relaciones entre ellas. El modelo para este sistema es el siguiente:



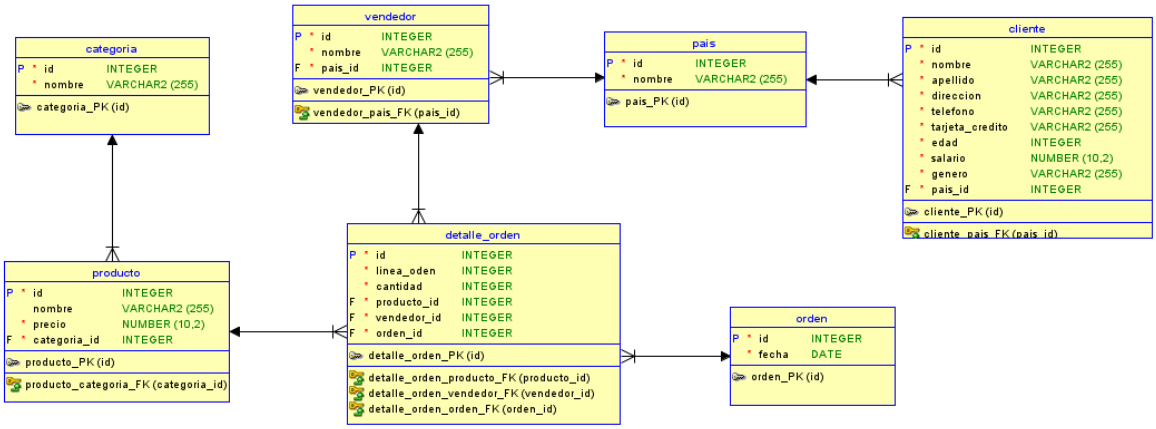
Modelo Lógico:

El modelo lógico traduce el modelo conceptual a un formato más técnico, especificando atributos y relaciones. El modelo para este sistema es el siguiente:



Modelo Relacional:

El modelo relacional representa las tablas y sus relaciones mediante un diagrama ER (Entidad-Relación). El modelo para este sistema es el siguiente:



Relación que tiene cada tabla:

- Categoria: No tiene una relación directa con otras tablas.
- producto: Tiene una relación de muchos a uno con la tabla categoría, ya que cada producto pertenece a una categoría específica.
- Pais: No tiene una relación directa con otras tablas
- Cliente: Tiene una relación de uno a muchos con la tabla orden, ya que un cliente puede realizar múltiples órdenes de venta. También tiene una relación de muchos a uno con la tabla país, ya que cada cliente pertenece a un país específico.
- Vendedor: Tiene una relación de uno a muchos con la tabla orden, ya que un vendedor puede realizar múltiples órdenes de venta. También tiene una relación de muchos a uno con la tabla pais, ya que cada vendedor pertenece a un país específico.
- Orden: Tiene una relación de uno a muchos con la tabla detalle_orden, ya que una orden puede tener múltiples detalles de orden. También tiene una relación de muchos a uno con la tabla cliente, ya que cada orden está asociada a un cliente específico.
- Detalle_orden: Tiene relaciones de muchos a uno con las tablas producto, vendedor y orden, ya que cada detalle de orden está asociado a un producto específico, un vendedor específico y una orden específica.

Estructura de la Base de Datos:

Categoria:

Esta tabla almacena las categorías de los productos. Se utiliza para categorizar los productos en la tabla producto.

```sql
CREATE TABLE categoria (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR (255) NOT NULL
);
```

- id: Identificador único de la categoría (clave primaria).
- nombre: Nombre de la categoría.

Producto:

Esta tabla almacena la información de los productos disponibles para la venta.

```sql
CREATE TABLE producto (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR (255) NOT NULL,
    precio DECIMAL (10, 2) NOT NULL,
    categoria_id INT NOT NULL,
    FOREIGN KEY (categoria_id) REFERENCES categoria(id)
);
```

- id: Identificador único del producto (clave primaria).
- nombre: Nombre del producto.
- precio: Precio del producto.
- categoria_id: Identificador de la categoría a la que pertenece el producto (clave foránea hacia la tabla categoría).

Pais:

Esta tabla almacena los países. Se utiliza para registrar el país de origen de los clientes y vendedores.

```
CREATE TABLE pais (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR (255) NOT NULL
);
```

- id: Identificador único del país (clave primaria).
- nombre: Nombre del país.

Cliente:

Esta tabla almacena la información de los clientes.

```
CREATE TABLE cliente (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR (255) NOT NULL,
    apellido VARCHAR (255) NOT NULL,
    direccion VARCHAR (255) NOT NULL,
    telefono VARCHAR (255) NOT NULL,
    tarjeta_credito VARCHAR (255) NOT NULL,
    edad INT NOT NULL,
    salario DECIMAL (10,2) NOT NULL,
    genero VARCHAR (255) NOT NULL,
    pais_id INT NOT NULL,
    FOREIGN KEY (pais_id) REFERENCES pais(id)
);
```

- id: Identificador único del cliente (clave primaria).
- nombre: Nombre del cliente.
- apellido: Apellido del cliente.
- direccion: Dirección del cliente.
- telefono: Número de teléfono del cliente.
- tarjeta_credito: Número de tarjeta de crédito del cliente.
- edad: Edad del cliente.
- salario: Salario del cliente.
- genero: Género del cliente.

- pais_id: Identificador del país de origen del cliente (clave foránea hacia la tabla pais).

Vendedor:

Esta tabla almacena la información de los vendedores.

```
CREATE TABLE vendedor (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR (255) NOT NULL,
    pais_id INT NOT NULL,
    FOREIGN KEY (pais_id) REFERENCES pais(id)
);
```

- id: Identificador único del vendedor (clave primaria).
- nombre: Nombre del vendedor.
- pais_id: Identificador del país de origen del vendedor (clave foránea hacia la tabla pais).

Orden:

Esta tabla almacena la información de las órdenes de venta.

```
CREATE TABLE orden (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    fecha DATE NOT NULL,
    cliente_id INT NOT NULL,
    FOREIGN KEY (cliente_id) REFERENCES cliente(id)
);
```

- id: Identificador único de la orden (clave primaria).
- fecha: Fecha en que se realizó la orden.
- cliente_id: Identificador del cliente que realizó la orden (clave foránea hacia la tabla cliente).

Detalle_Orden:

Esta tabla almacena los detalles de las órdenes de venta, es decir, los productos que se incluyeron en cada orden.

```sql
CREATE TABLE detalle_orden (
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    linea_orden INT NOT NULL,
    cantidad INT NOT NULL,
    producto_id INT NOT NULL,
    vendedor_id INT NOT NULL,
    orden_id INT NOT NULL,
    FOREIGN KEY (producto_id) REFERENCES producto(id),
    FOREIGN KEY (vendedor_id) REFERENCES vendedor(id),
    FOREIGN KEY (orden_id) REFERENCES orden(id)
);
```

- id: Identificador único del detalle de orden (clave primaria).
- linea_orden: Número de línea de la orden.
- cantidad: Cantidad de productos incluidos en la orden.
- producto_id: Identificador del producto incluido en la orden (clave foránea hacia la tabla producto).
- vendedor_id: Identificador del vendedor asociado a la orden (clave foránea hacia la tabla vendedor).
- orden_id: Identificador de la orden a la que pertenece el detalle (clave foránea hacia la tabla orden).

API

A continuación se muestran fragmentos de códigos con su respectiva descripción.

Conexión a la base de datos:

```python
connection = pymysql.connect(
    host='localhost',
    user='root',
    password='admin123',
    database='empresa',
    cursorclass=pymysql.cursors.DictCursor
)
```

Este código establece una conexión en un servidor local con la base de datos en MySQL la cual tiene como nombre “empresa”, también configura el cursor para devolver los resultados de las consultas como diccionarios

Consulta 1:

```python
@app.route('/consulta1', methods=['GET'])
def get_query1():
    try:
        with connection.cursor() as cursor:
            query = """
            SELECT cliente.id AS id_cliente,
                cliente.nombre AS nombre_cliente,
                cliente.apellido AS apellido_cliente,
                pais.nombre AS pais_cliente,
                SUM(detalle_orden.cantidad * producto.precio) AS monto_total
            FROM cliente
            JOIN orden ON cliente.id = orden.cliente_id
            JOIN detalle_orden ON orden.id = detalle_orden.orden_id
            JOIN producto ON detalle_orden.producto_id = producto.id
            JOIN pais ON cliente.pais_id = pais.id
            GROUP BY cliente.id
            ORDER BY monto_total DESC
            LIMIT 1;
            """
            cursor.execute(query)
            customer = cursor.fetchone()
    except Exception as e:
        return f"Error: {str(e)}"
    finally:
        cursor.close()
    return jsonify(customer)
```

Este código maneja las solicitudes GET en la ruta “/consulta1”, ejecuta una consulta SQL para recuperar la información del cliente que más ha comprado y devuelve los resultados como una respuesta JSON.

Eliminar modelo:

```python
@app.route('/eliminarmodelo', methods=['GET'])
def get_delete_model():
    try:
        with connection.cursor() as cursor:
            queries = [
                "DROP TABLE detalle_orden;",
                "DROP TABLE orden;",
                "DROP TABLE cliente;",
                "DROP TABLE vendedor;",
                "DROP TABLE pais;",
                "DROP TABLE producto;",
                "DROP TABLE categoria;"
            ]
            for query in queries:
                cursor.execute(query)
            connection.commit()
    except Exception as e:
        return f"Error: {str(e)}"
    finally:
        cursor.close()
    return jsonify({'message': 'Modelo eliminado correctamente'})
```

Este código maneja las solicitudes GET en la ruta “/eliminarmodelo”, ejecuta una serie de consultas SQL para eliminar las tablas del modelo de la base de datos y devuelve un mensaje JSON indicando el éxito de la operación.