Uniremington

Facultad de ingenierías

PARCIAL: PROYECTO FINAL

Materia:

Programación Web I

(Distancia y Virtualidad)

Código:

2408B04G1

Estudiante:

Alejandro Vélez Gómez

Profesor:

Javier Ospina Moreno

Fecha de entrega:

10/05/2024

ÍNDICE:

Definición del problema

El sistema a desarrollar tiene como objetivo la gestión eficiente de una plataforma de ventas en línea. Las necesidades detectadas incluyen la falta de una herramienta para gestionar productos, categorías, clientes, y pedidos de manera organizada. El sistema debe permitir la creación, edición, eliminación y consulta de estos datos, brindando una interfaz amigable para los administradores.

Diseño de la base de datos

Nombre de la base de datos:

Ejemplo: ventas_online

Listado de tablas (referenciales y de movimiento):

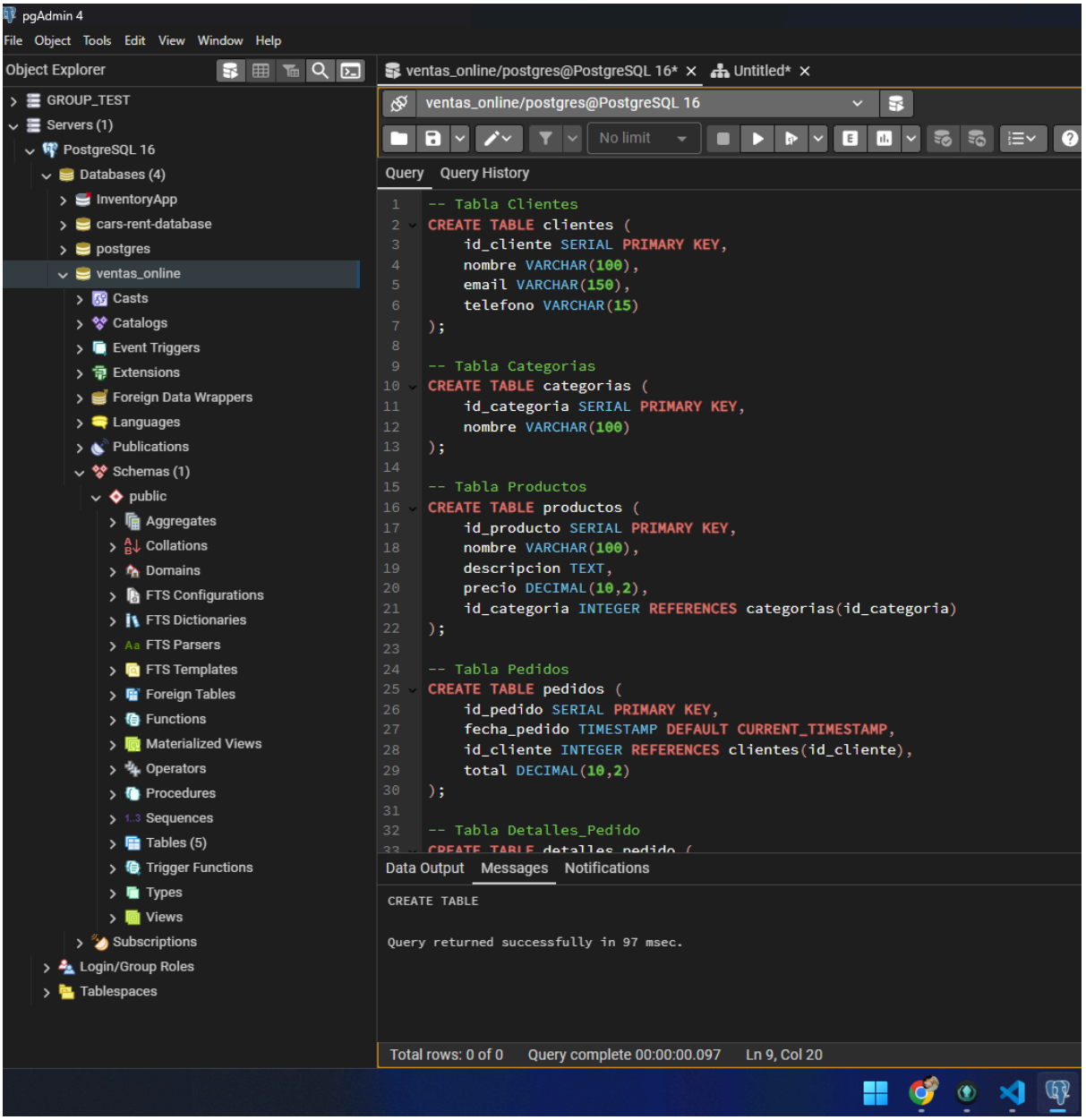
- **Tablas referenciales (maestras):**
    - clientes: para almacenar la información de los clientes.
    - productos: para almacenar la información de los productos.
    - categorías: para gestionar las categorías de los productos.
- **Tablas de movimiento (transaccionales):**
    - pedidos: para gestionar los pedidos realizados por los clientes.
    - detalles pedido: para almacenar los productos que forman parte de cada pedido.

Guia, Base de datos:

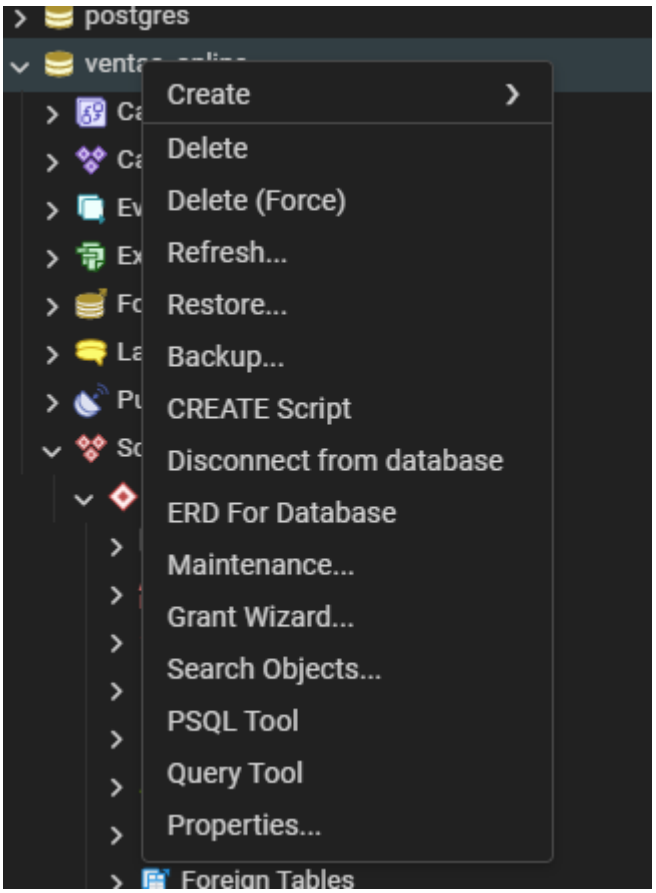
Para la creación de la base de datos, se usó postgres, para ello, una configurado pgAdmin 4, creamos un base de datos con nuestro nombre de preferencia, seguido a ello, usamos la herramienta de Querys SQL, y ejecutamos el siguiente código:

```sql
-- Tabla Clientes
CREATE TABLE clientes (
    id_cliente SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    email VARCHAR(150),
    telefono VARCHAR(15)
);

-- Tabla Categorias
CREATE TABLE categorias (
    id_categoria SERIAL PRIMARY KEY,
    nombre VARCHAR(100)
);

-- Tabla Productos
CREATE TABLE productos (
    id_producto SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    descripcion TEXT,
    precio DECIMAL(10,2),
    id_categoria INTEGER REFERENCES categorias(id_categoria)
);

-- Tabla Pedidos
CREATE TABLE pedidos (
    id_pedido SERIAL PRIMARY KEY,
    fecha_pedido TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    id_cliente INTEGER REFERENCES clientes(id_cliente),
    total DECIMAL(10,2)
);

-- Tabla Detalles_Pedido
CREATE TABLE detalles_pedido (
    id_detalle SERIAL PRIMARY KEY,
    id_pedido INTEGER REFERENCES pedidos(id_pedido),
    id_producto INTEGER REFERENCES productos(id_producto),
    cantidad INTEGER,
    precio_unitario DECIMAL(10,2)
);
```

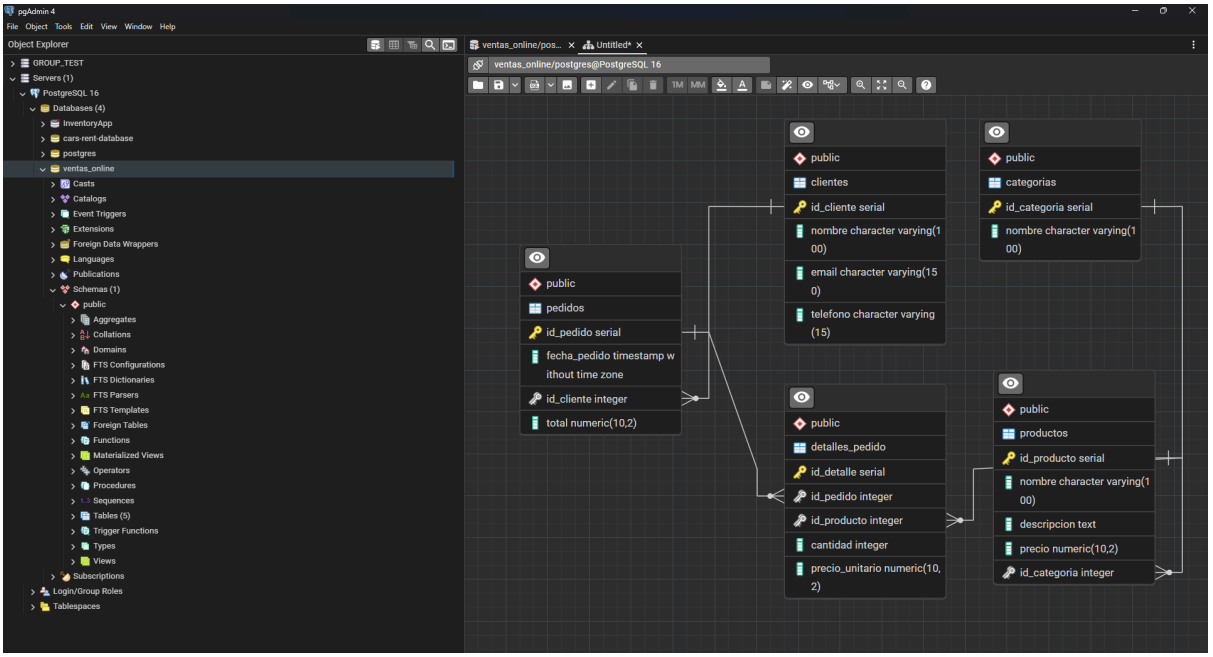
En nuestro pgAdmin 4 debería verse tal que así:

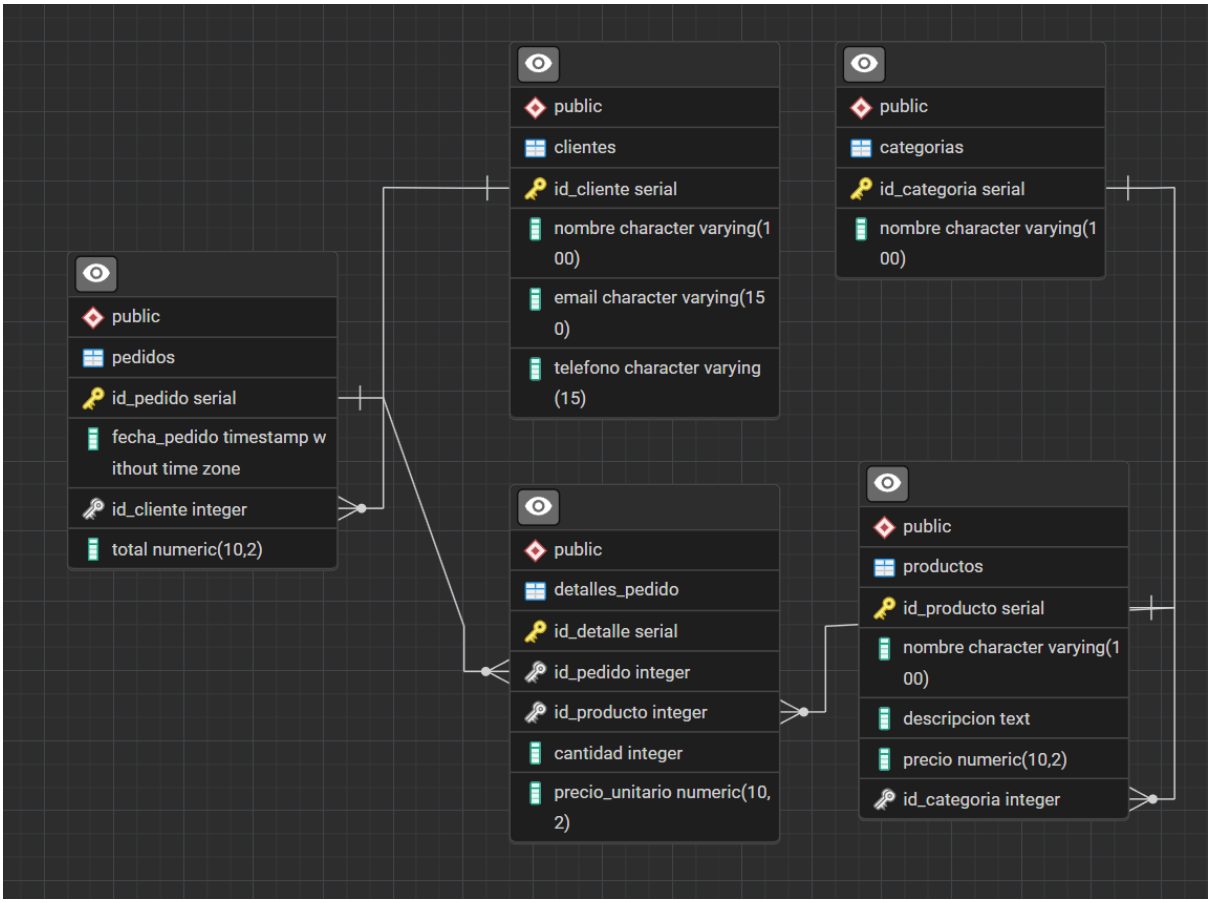


Una vez creadas las tablas podremos ver el modelo de ERD de nuestra base de datos. Haciendo uso de la Herramienta ERD for database, se nos genera automáticamente el modelo de nuestra base de datos:



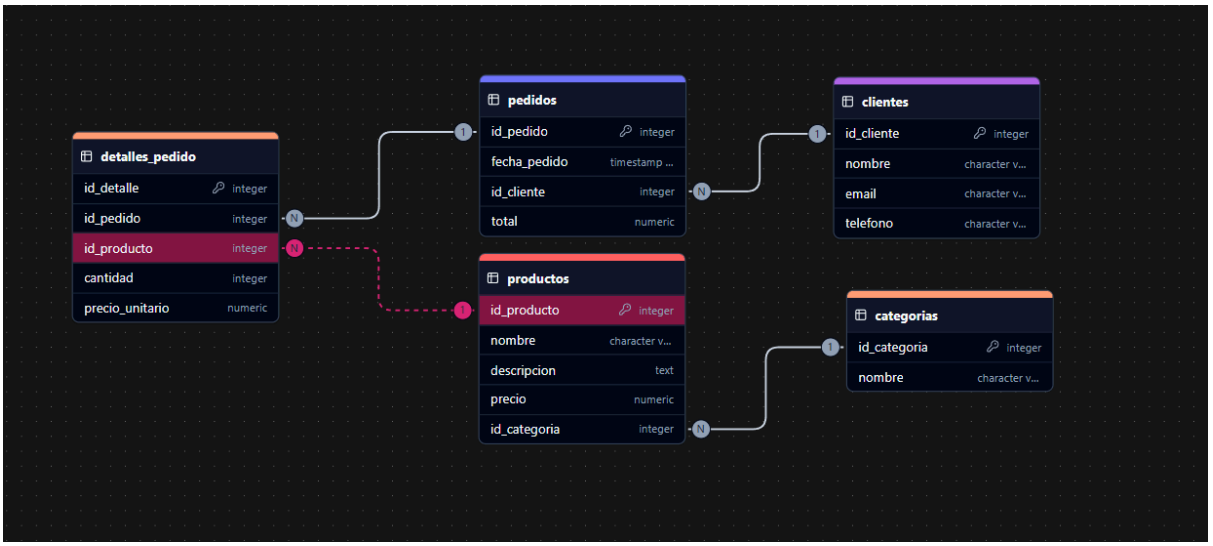
Luego de unos segundos tendremos lo siguiente:





Además, también podemos modelar nuestro propio ERD en páginas como:

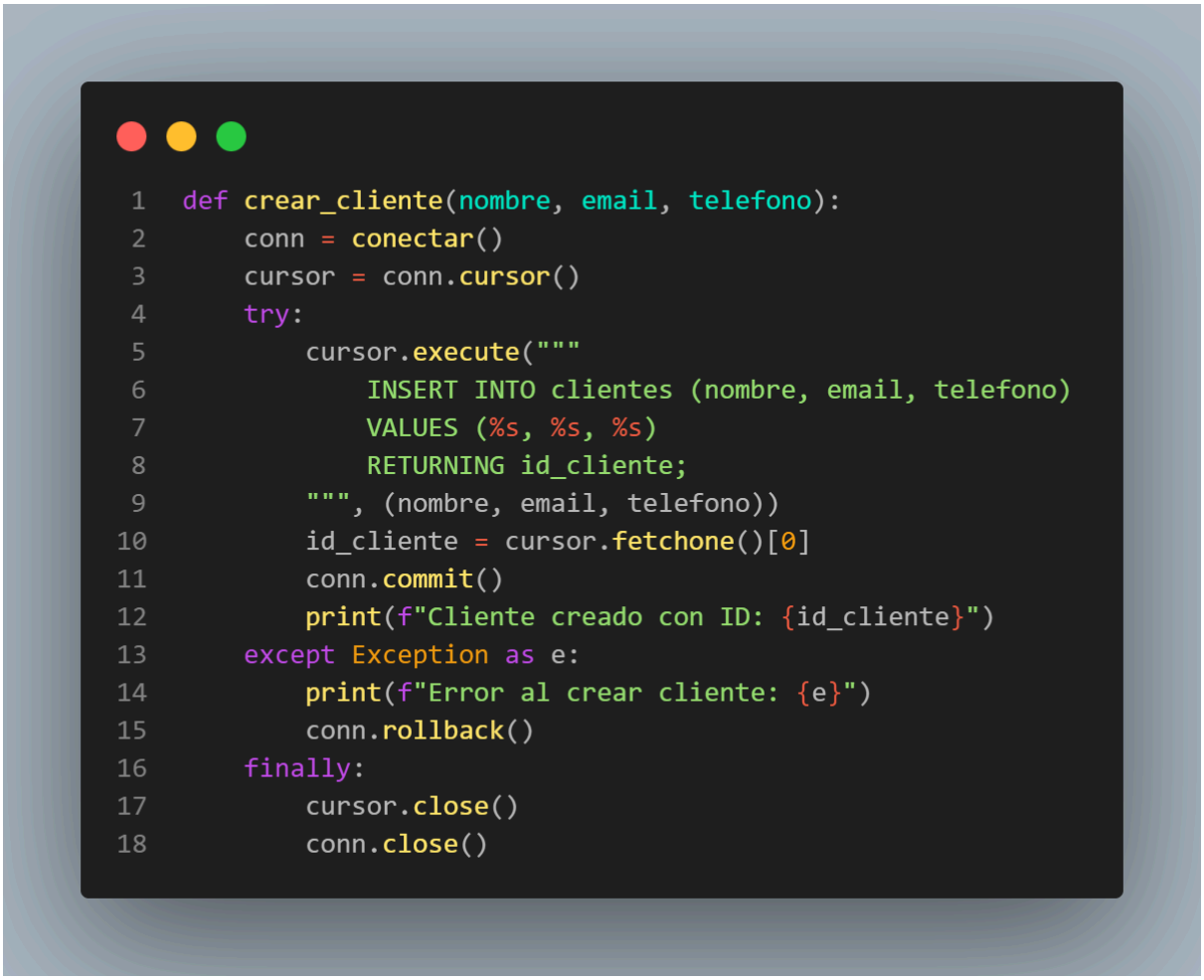
https://app.chartdb.io/



CRUD en python para la base de datos

Para temas de demostración, solo mostrare el crud de los clientes:

Creación:



```python
def crear_cliente(nombre, email, telefono):
    conn = conectar()
    cursor = conn.cursor()
    try:
        cursor.execute("""
            INSERT INTO clientes (nombre, email, telefono)
            VALUES (%s, %s, %s)
            RETURNING id_cliente;
        """, (nombre, email, telefono))
        id_cliente = cursor.fetchone()[0]
        conn.commit()
        print(f"Cliente creado con ID: {id_cliente}")
    except Exception as e:
        print(f"Error al crear cliente: {e}")
        conn.rollback()
    finally:
        cursor.close()
        conn.close()
```

Eliminación:

```python
def eliminar_cliente(id_cliente):
    conn = conectar()
    cursor = conn.cursor()
    try:
        cursor.execute("DELETE FROM clientes WHERE id_cliente = %s;", (id_cliente,))
        conn.commit()
        print(f"Cliente con ID {id_cliente} eliminado.")
    except Exception as e:
        print(f"Error al eliminar cliente: {e}")
        conn.rollback()
    finally:
        cursor.close()
        conn.close()
```

Actualización:

```python
def actualizar_cliente(id_cliente, nombre, email, telefono):
    conn = conectar()
    cursor = conn.cursor()
    try:
        cursor.execute("""
            UPDATE clientes
            SET nombre = %s, email = %s, telefono = %s
            WHERE id_cliente = %s;
        """, (nombre, email, telefono, id_cliente))
        conn.commit()
        print(f"Cliente con ID {id_cliente} actualizado.")
    except Exception as e:
        print(f"Error al actualizar cliente: {e}")
        conn.rollback()
    finally:
        cursor.close()
        conn.close()
```

Lectura:

```python
def leer_clientes():
    conn = conectar()
    cursor = conn.cursor()
    try:
        cursor.execute("SELECT * FROM clientes;")
        clientes = cursor.fetchall()
        for cliente in clientes:
            print(cliente)
    except Exception as e:
        print(f"Error al leer clientes: {e}")
    finally:
        cursor.close()
        conn.close()
```

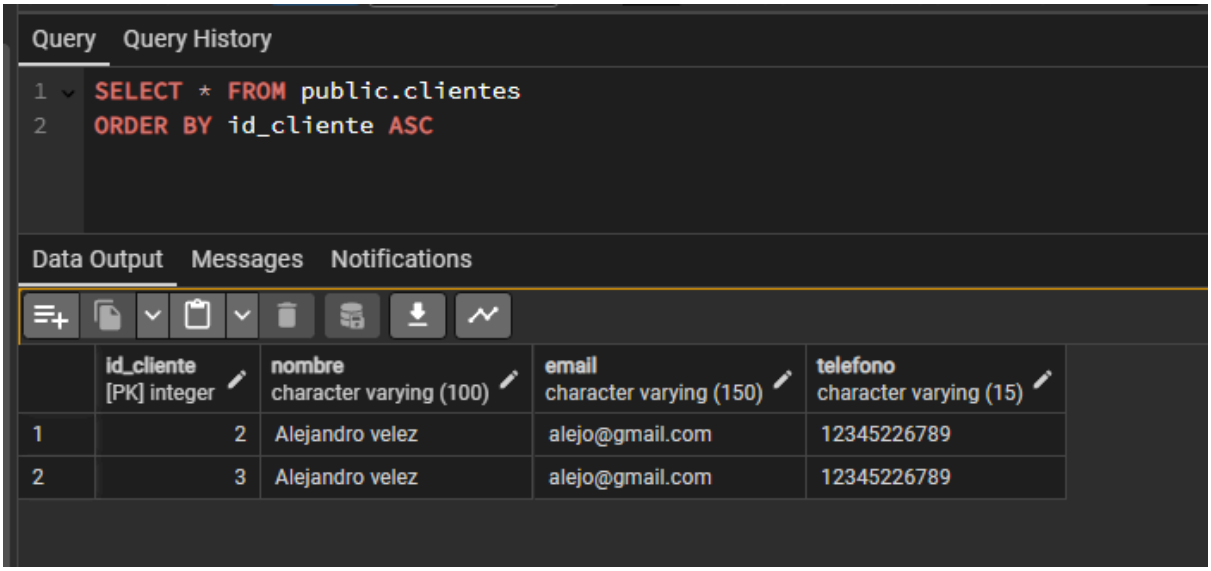
Ejemplo de uso:

```python
if __name__ == "__main__":
    # Crear cliente
    crear_cliente("Alejandro velez", "alejo@gmail.com", "12345226789")

    # Leer clientes
    leer_clientes()

    # Actualizar cliente
    actualizar_cliente(1, "Alejandro velez", "alejo@nuevoemail.com", "98761231254321")

    # Eliminar cliente
    eliminar_cliente(1)
```

Conexión:

```python
# Configuración de la conexión
def conectar():
    return psycopg2.connect(
        host="localhost",
        database="ventas_online",
        user="postgres",
        password=12345
    )
```

Verificación en la base de datos:



Query | Query History

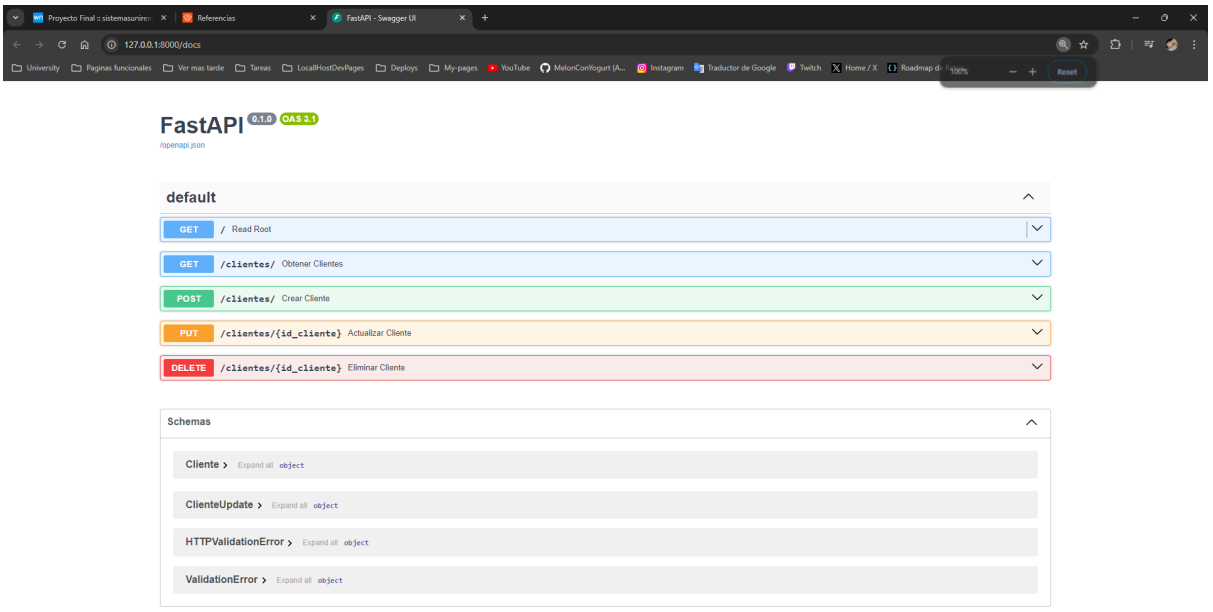
```
SELECT * FROM public.clientes
ORDER BY id_cliente ASC
```

Data Output | Messages | Notifications

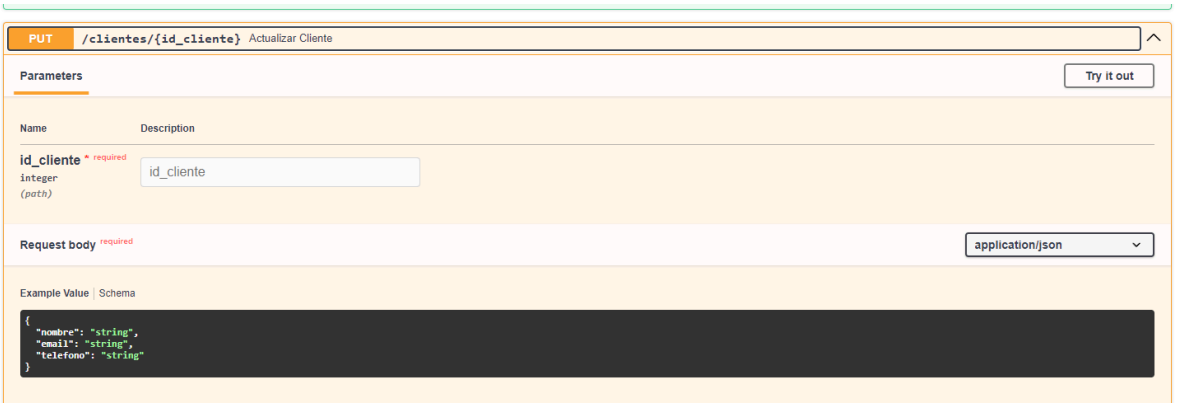
	id_cliente [PK] integer	nombre character varying (100)	email character varying (150)	telefono character varying (15)
1	2	Alejandro velez	alejo@gmail.com	12345226789
2	3	Alejandro velez	alejo@gmail.com	12345226789

Vista y manejo del crud desde la web

Haciendo uso de fast api, podremos tener un interfaz gráfica con la cual podremos hacer uso de nuestro crud fácilmente:

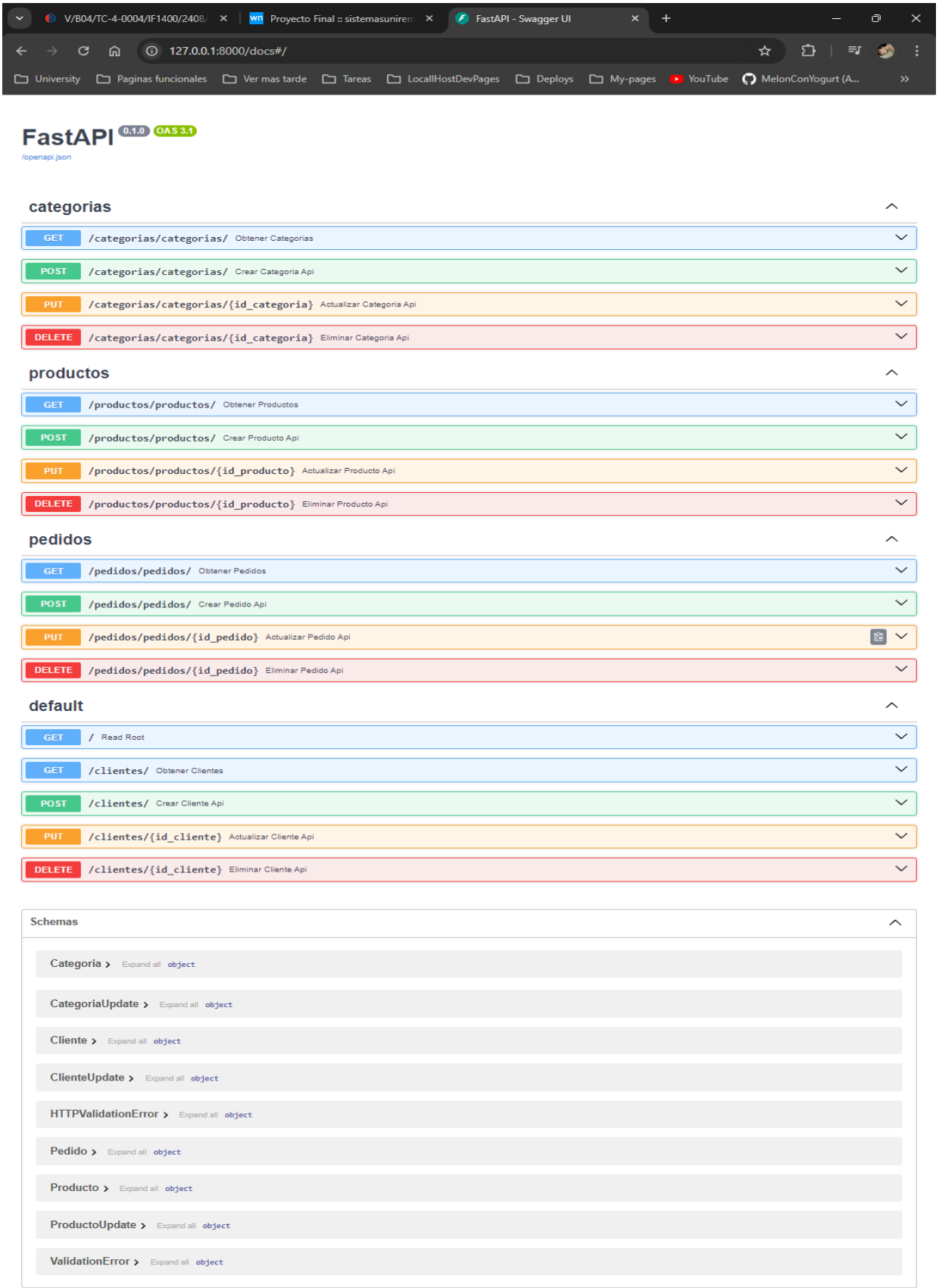


Cada método nos pedirá el dato que requiere, como en el método de PUT, que necesitamos el ID del cliente:



para acceder a al pagina, solo tenemos que ir a la ruta: http://127.0.0.1:8000/docs#/

Todas las funciones disponibles:



Referencias:

ChartDB. (s. f.). https://app.chartdb.io/

FastAPI. (s. f.). https://fastapi.tiangolo.com/

PostgreSQL Tutorial. (2024, 19 mayo). *PostgreSQL Python: Connecting to PostgreSQL Server*. https://www.postgresqltutorial.com/postgresql-python/connect/