

FACULTAD DE INGENIERÍA
INGENIERIA DE SISTEMAS

Desarrollo Basado en Plataformas

Arquitectura de Datos

Manual de pruebas

Proyecto Biblioteca

AUTOR/S

María Camila Bernal Calderón

Sebastian Guerra Garzón

DOCENTE

JUAN CARLOS MARTINEZ DIAZ

NRC:

60 – 3830

60 – 3825

BOGOTÁ D.C – COLOMBIA

2024

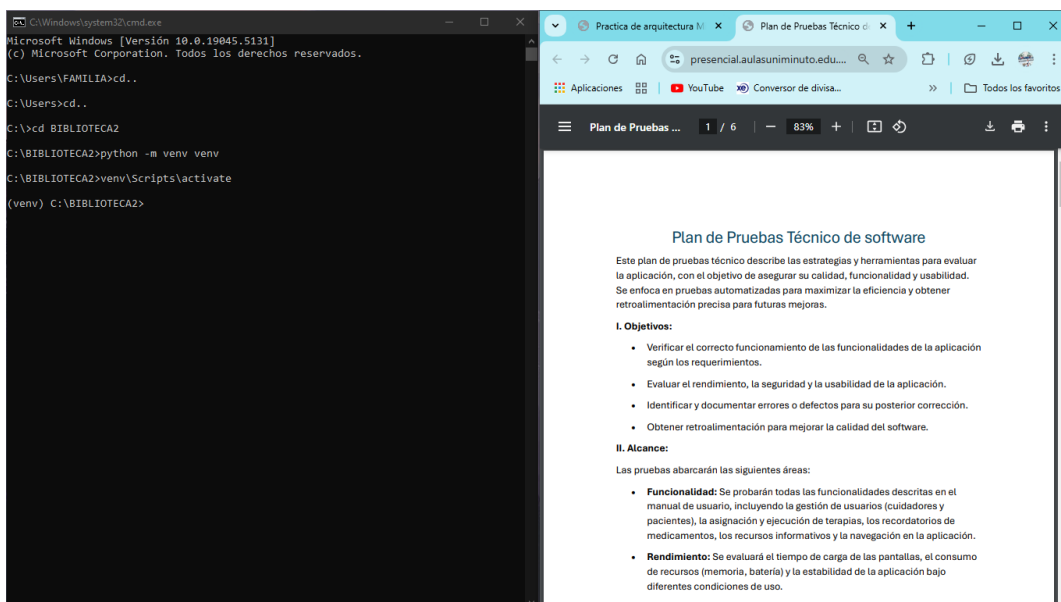
INTRODUCCIÓN

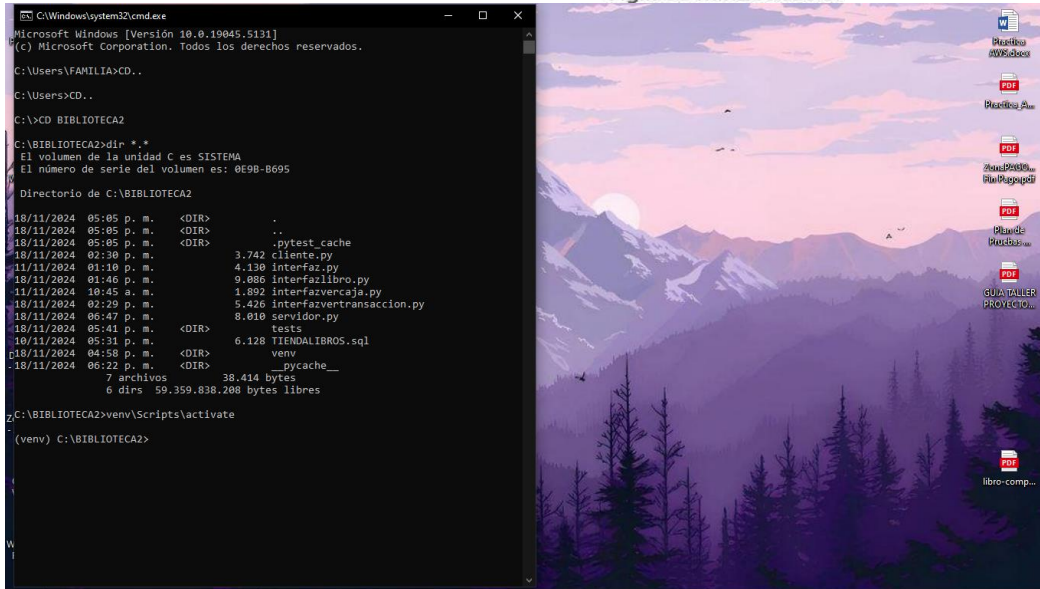
A continuación, veremos la documentación sobre las pruebas para el proyecto BIBLIOTECA utilizando el modelo cliente – servidor, el motivo es validar que el sistema funcione adecuadamente, para realizar esta practica vamos a utilizar el visual estudio code junto a un nueva librería llamada Pytest, esta es una librería que es netamente para probar código Python y se usa más para la creación de modelos cliente servidor para validar si cumplen con ciertos estándares que se les pide al servidor. Es como hacer una prueba de estrés al sistema si este sistema puede aguantar una ejecución como si miles de personas hicieran peticiones como si de un sistema de búsqueda se tratara. Dentro de la documentación veremos prueba automática y manual tanto para el Python con automática y manual y para la base de datos SQL haremos manual aun que se tratara de implementar una prueba automatizada con el fin de presentar una prueba adecuada para el sistema además de contar con pruebas de captura de pantalla y grabaciones de pantalla como se es solicitado en el documento proporcionado por el docente Juan Carlos Martínez Diaz. Con esto aclarado demos comienzo a nuestra documentación.

Bueno para iniciar con este modelo, primero iniciemos desde un principio ¿verdad?, nosotros como futuros ingenieros en sistemas que seremos si decidimos crear un software para una compañía o una empresa que requiera algún sistema que pueda debemos de realizarle a este un respectivo plan de pruebas esto con la idea de poder generar más confianza y seguridad a quien nos contacto para que creamos un sistema. En este caso nuestro sistema se trata de crear un sistema que pueda gestionar una biblioteca. Este debe cumplir con algunos requisitos para poder funcionar estos son los requerimientos funcionales. El objetivo de esta practica es verificar el correcto funcionamiento de la aplicación evaluar el rendimiento y la seguridad al usarla, para nuestro proyecto no podemos usar todos los protocolos de validación no por que no queramos, si no que nuestro sistema fue diseñado para pruebas es decir que cumpla con los objetivos establecidos y que este pueda generar las funciones que se le solicita.

En este orden de ideas empezemos el plan de estudio para un cliente servidor, como recordatorio estamos usando un servidor por medio de un flask y requests esto como base claro está, para iniciar esta primera parte vamos a estructura nuestra carpeta en donde se aloja nuestro proyecto, en el caso de nosotros manejamos el proyecto desde la raíz del dispositivo es decir “C:” esto por comodidad, dentro de la carpeta del proyecto tenemos que crear un ambiente virtual, un ambiente virtual sirve para ejecutar pruebas de códigos como un cliente servidor o procesos como microservicios. Esto por comodidad y es como si ejecutáramos un programa pesado como un KALI LINUX en una máquina virtual como virtual BOX se ejecuta un programa grande, pero usando un ambiente de amplia capacidad para ejecutar.

Usando el comando de tecla Windows + r abrimos un terminal CMD y nos dirigimos a la carpeta donde alojamos nuestro proyecto. En este caso como esta en la raíz con el comando “cd..” esto con la finalidad de ir a la raíz de la máquina, al abrir la carpeta de nuestro proyecto en este caso con un “cd BIBLIOTECA2” dentro de la carpeta vamos a crear un ambiente virtual con el comando “Python -m venv venv” esto es para crearlo y para llamar el ambiente virtual es con el comando “venv/Scripts/actíivate” como sabemos que estamos usando el ambiente virtual fácil en la línea de código vemos un (venv) eso indica el ambiente virtual.





Cuando ingresamos a este punto debemos de realizar un cambio, pero en nuestro código fuente y carpeta de alojamiento, dentro de la carpeta de alojamiento creamos una nueva carpeta que diga test y dentro de esa carpeta vamos a alojar un archivo test_biblioteca.py que será nuestro testeo igual a una práctica que hicimos anteriormente. De acá en adelante vamos a realizar algunos cambios en dos códigos y son los principales en nuestro servidor y cliente esto con el fin de que se pueda reconocer el archivo al momento de ejecutar la prueba.

CODIGO FUENTE SERVIDOR.PY

```

from flask import Flask, jsonify, request
from flask_httpauth import HTTPBasicAuth
import pyodbc

app = Flask(__name__)
auth = HTTPBasicAuth()

# Configuración de la base de datos
def get_db_connection():
    conn = pyodbc.connect(
        'DRIVER={ODBC Driver 17 for SQL Server};'
        'SERVER=DESKTOP-0E1UK80\\SQLNEW;'
        'DATABASE=TiendaLibros;'
        'UID=sa;'
        'PWD=camila2004;'
        'TrustServerCertificate=yes;'
    )
    return conn

# Usuarios y contraseñas
usuarios = {

```

```
"camila": "ca2004",
"sebastian": "se2003"
}

@auth.verify_password
def verify_password(usuario, contraseña):
    if usuario in usuarios and usuarios[usuario] == contraseña:
        return usuario
    return None

# Ruta para obtener todos los libros
@app.route('/Libros', methods=['GET'])
@auth.login_required
def obtener_libros():
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM Libros')
    libros = cursor.fetchall()
    conn.close()
    return jsonify([
        {
            'ISBN': libro[0],
            'titulo': libro[1],
            'precio_compra': float(libro[2]),
            'precio_venta': float(libro[3]),
            'cantidad_actual': libro[4]
        } for libro in libros])

# Ruta para agregar un libro
@app.route('/Libros', methods=['POST'])
@auth.login_required
def agregar_libro():
    nuevo_libro = request.get_json()
    conn = get_db_connection()
    cursor = conn.cursor()
    try:
        cursor.execute('INSERT INTO Libros (ISBN, titulo, precio_compra,
precio_venta, cantidad_actual) VALUES (?, ?, ?, ?, ?)',
                        (nuevo_libro['ISBN'], nuevo_libro['titulo'],
nuevo_libro['precio_compra'], nuevo_libro['precio_venta'],
nuevo_libro['cantidad_actual']))
        conn.commit()
        return jsonify({'mensaje': 'Libro agregado exitosamente'}), 201
    except Exception as e:
        conn.rollback()
```

```
        return jsonify({'error': str(e)}), 400
    finally:
        conn.close()

# Ruta para actualizar un libro
@app.route('/Libros/<string:isbn>', methods=['PUT'])
@auth.login_required
def actualizar_libro(isbn):
    datos_libro = request.get_json()
    conn = get_db_connection()
    cursor = conn.cursor()
    try:
        cursor.execute('UPDATE Libros SET titulo = ?, precio_compra = ?,
precio_venta = ?, cantidad_actual = ? WHERE ISBN = ?',
                        datos_libro['titulo'],
datos_libro['precio_compra'], datos_libro['precio_venta'],
datos_libro['cantidad_actual'], isbn)
        conn.commit()
        return jsonify({'mensaje': 'Libro actualizado exitosamente'})
    except Exception as e:
        conn.rollback()
        return jsonify({'error': str(e)}), 400
    finally:
        conn.close()

# Ruta para eliminar un libro
@app.route('/Libros/<string:isbn>', methods=['DELETE'])
@auth.login_required
def eliminar_libro(isbn):
    conn = get_db_connection()
    cursor = conn.cursor()
    try:
        cursor.execute('DELETE FROM Libros WHERE ISBN = ?', isbn)
        conn.commit()
        return jsonify({'mensaje': 'Libro eliminado exitosamente'})
    except Exception as e:
        conn.rollback()
        return jsonify({'error': str(e)}), 400
    finally:
        conn.close()

@app.route('/add_transaction', methods=['POST'])
@auth.login_required
def add_transaction():
```

```

data = request.get_json()
conn = get_db_connection()
cursor = conn.cursor()
try:
    cursor.execute("BEGIN TRANSACTION")

    # Obtener los datos del libro
    cursor.execute("SELECT cantidad_actual, precio_compra,
precio_venta FROM Libros WHERE ISBN = ?", data['ISBN'])
    libro = cursor.fetchone()

    if not libro:
        raise Exception("Libro no encontrado")

    cantidad_actual, precio_compra, precio_venta = libro

    # Comprobación para tipo de transacción
    if data['tipo_transaccion'] == 1: # VENTA
        if cantidad_actual < data['cantidad']:
            raise Exception("No hay suficiente stock")
        nueva_cantidad = cantidad_actual - data['cantidad'] #
Correcto: Restar la cantidad vendida
        monto = data['cantidad'] * precio_venta # Correcto: Calcular
el ingreso total
        tipo_movimiento = 'INGRESO'
    else: # ABASTECIMIENTO
        nueva_cantidad = cantidad_actual + data['cantidad'] #
Correcto: Sumar la cantidad abastecida
        monto = data['cantidad'] * precio_compra # Correcto: Calcular
el costo total
        tipo_movimiento = 'EGRESO'

    # Actualización del libro
    cursor.execute("UPDATE Libros SET cantidad_actual = ? WHERE ISBN =
?", nueva_cantidad, data['ISBN'])

    # Insertar transacción
    cursor.execute("INSERT INTO Transacciones (ISBN, tipo_transaccion,
cantidad) VALUES (?, ?, ?)",
                    data['ISBN'], data['tipo_transaccion'],
data['cantidad'])

    # Obtener el ID de la transacción insertada
    cursor.execute("SELECT SCOPE_IDENTITY()")

```

```
id_transaccion = cursor.fetchone()[0]

# Obtener el último saldo de caja
cursor.execute("SELECT TOP 1 saldo_actual FROM Caja ORDER BY
id_movimiento DESC")
ultimo_saldo = cursor.fetchone()

if ultimo_saldo:
    ultimo_saldo = ultimo_saldo[0]
else:
    ultimo_saldo = 0

# Calcular el nuevo saldo
nuevo_saldo = ultimo_saldo + monto if tipo_movimiento == 'INGRESO'
else ultimo_saldo - monto

# Insertar en la caja
cursor.execute("INSERT INTO Caja (tipo_movimiento, monto,
saldo_actual, id_transaccion) VALUES (?, ?, ?, ?)",
                tipo_movimiento, monto, nuevo_saldo,
id_transaccion)

# Finalizar la transacción
cursor.execute("COMMIT")

conn.commit()
return jsonify({'mensaje': 'Transacción registrada
exitosamente'}), 201

except Exception as e:
    print(f"Error en la transacción: {e}") # Agregar depuración
    conn.rollback()
    return jsonify({'error': str(e)}), 500
finally:
    conn.close()

@app.route('/Transacciones', methods=['GET'])
@auth.login_required
def obtener_transacciones():
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
        SELECT t.*, tt.nombre as nombre_tipo
        FROM Transacciones t
```



```

        JOIN TiposTransaccion tt ON t.tipo_transaccion = tt.id_tipo
        ORDER BY fecha_transaccion DESC
    """)
    transacciones = cursor.fetchall()
    conn.close()
    transacciones_json = [{'id_transaccion': t.id_transaccion,
                           'ISBN': t.ISBN,
                           'tipo_transaccion': t.nombre_tipo,
                           'fecha_transaccion': t.fecha_transaccion,
                           'cantidad': t.cantidad} for t in
transacciones]
    return jsonify(transacciones_json)

@app.route('/TiposTransaccion', methods=['GET'])
@auth.login_required
def obtener_tipos_transaccion():
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM TiposTransaccion")
    tipos = cursor.fetchall()
    conn.close()
    tipos_json = [{'id_tipo': tipo.id_tipo, 'nombre': tipo.nombre} for
tipo in tipos]
    return jsonify(tipos_json)

@app.route('/estado_caja', methods=['GET'])
@auth.login_required
def obtener_estado_caja():
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT TOP 1 saldo_actual FROM Caja ORDER BY
fecha_movimiento DESC")
    resultado = cursor.fetchone()
    conn.close()

    if resultado:
        estado_caja = resultado[0] # Extraer el saldo actual
        return jsonify({"estado_caja": estado_caja})
    else:
        return jsonify({"estado_caja": "No disponible"}), 404

if __name__ == '__main__':
    app.run(port=5000)

```

El código de nuestro servidor se mantiene igual puesto que este tiene los disparadores dentro de lo que cabe el cliente solo solicita una función y el servidor es que hace la petición a la base de datos hace petición la SQL la devuelve al servidor y el cliente la envía como respuesta a la solicita hecha por el usuario, eso es conocimiento básico que debemos de conocer verdad. Por esa razón se debe de mantener intacto el código del servidor, pero en nuestro caso mas adelante tuvimos que hacer un cambio, pero eso lo veremos mas adelante, por el momento sigamos con los códigos.

CODIGO FUENTE CLIENTE.PY

```
import requests

#Funcion para establecer la autenticación
def establecer_aut(usuario, contraseña):
    global auth
    auth = (usuario, contraseña)

#Funcion para obtener libros
def obtener_libros():
    try:
        response = requests.get('http://localhost:5000/Libros', auth=auth)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as e:
        print (f"Error al obtener Libros: {e}")
        return []

#Funcion para agregar Libros
def agregar_libro(ISBN, titulo, precio_compra, precio_venta,
cantidad_actual):
    data = {
        "ISBN": ISBN,
        "titulo": titulo,
        "precio_compra": precio_compra,
        "precio_venta": precio_venta,
        "cantidad_actual": cantidad_actual
    }
    try:
        response = requests.post('http://localhost:5000/Libros',
json=data, auth=auth)
        response.raise_for_status()
        print("Libro agregado exitosamente")
    except requests.exceptions.RequestException as e:
        print(f"Error al agregar libro: {e}")

#Funcion para actualizar un libro
```

```
def actualizar_libro(ISBN, titulo, precio_compra, precio_venta,
cantidad_actual):
    data = {
        "titulo": titulo,
        "precio_compra": precio_compra,
        "precio_venta": precio_venta,
        "cantidad_actual": cantidad_actual
    }
    try:
        response = requests.put(f'http://localhost:5000/Libros/{ISBN}',
json=data, auth=auth)
        response.raise_for_status()
        print("Libro actualizado exitosamente")
    except requests.exceptions.RequestException as e:
        print(f"Error al actualizar libro: {e}")

# Función para eliminar un libro
def eliminar_libro(ISBN):
    try:
        response = requests.delete(f'http://localhost:5000/Libros/{ISBN}',
auth=auth)
        response.raise_for_status()
        print("Libro eliminado exitosamente")
    except requests.exceptions.RequestException as e:
        print(f"Error al eliminar libro: {e}")

# Función para registrar una transacción
def registrar_transaccion(tipo_transaccion, ISBN, cantidad):
    data = {
        "tipo_transaccion": tipo_transaccion,
        "ISBN": ISBN,
        "cantidad": cantidad
    }
    try:
        response = requests.post('http://localhost:5000/add_transaction',
json=data, auth=auth)
        response.raise_for_status()
        print("Transacción registrada exitosamente")
    except requests.exceptions.RequestException as e:
        print(f"Error al registrar transacción: {e}")

# Función para obtener transacciones
def obtener_transacciones():
    try:
```

```

        response = requests.get('http://localhost:5000/Transacciones',
auth=auth)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as e:
        print(f"Error al obtener transacciones: {e}")
        return []

# Función para obtener tipos de transacción
def obtener_tipos_transaccion():
    try:
        response = requests.get('http://localhost:5000/TiposTransaccion',
auth=auth)
        response.raise_for_status()
        return response.json()
    except requests.exceptions.RequestException as e:
        print(f"Error al obtener tipos de transacción: {e}")
        return []

# Función para obtener el estado de la caja
def obtener_estado_caja():
    try:
        response = requests.get('http://localhost:5000/estado_caja',
auth=auth)
        response.raise_for_status()
        return response.json().get("estado_caja", "No disponible")
    except requests.exceptions.RequestException as e:
        print(f"Error al obtener el estado de la caja: {e}")
        return "Error al conectar con el servidor"

```

El código de cliente juega un papel importante puesto que este es donde pide las solicitudes y como no hay una interfaz en este caso, todo tiene que ser de manera interna, manteniendo una comunicación entre servidor y cliente junto con el test de biblioteca que tenemos.

CODIGO FUENTE TEST_BIBLIOTECA.PY

```

import pytest
import requests
from servidor import app

# Fixture para ejecutar el servidor durante las pruebas
@pytest.fixture(scope="module")
def app_cliente():
    # Configuramos las credenciales por defecto para todas las pruebas
    app.config['TESTING'] = True
    with app.test_client() as test_client:

```

```
yield test_client

# Prueba la respuesta del servidor
def test_servidor_responde(app_cliente):
    # Agregamos autenticación básica
    response = app_cliente.get('/Libros', headers={'Authorization': 'Basic
Y2FtaWxhOmNhMjAwNA=='}) # Base64: "camila:ca2004"
    assert response.status_code == 200

# Prueba que la respuesta sea en formato JSON
def test_formato_json(app_cliente):
    response = app_cliente.get('/Libros', headers={'Authorization': 'Basic
Y2FtaWxhOmNhMjAwNA=='})
    assert response.content_type == 'application/json'

# Prueba el contenido de la respuesta (obtenemos libros)
def test_obtener_libros(app_cliente):
    response = app_cliente.get('/Libros', headers={'Authorization': 'Basic
Y2FtaWxhOmNhMjAwNA=='})
    data = response.get_json()
    assert isinstance(data, list)
    # Modificamos para permitir lista vacía inicialmente
    assert isinstance(data, list) # Solo verificamos que sea una lista

# Prueba agregar un libro
def test_agregar_libro(app_cliente):
    nuevo_libro = {
        "ISBN": "1234567890",
        "titulo": "Nuevo libro de prueba",
        "precio_compra": 100,
        "precio_venta": 150,
        "cantidad_actual": 10
    }
    response = app_cliente.post('/Libros',
                                json=nuevo_libro,
                                headers={'Authorization': 'Basic
Y2FtaWxhOmNhMjAwNA==', 'Content-Type': 'application/json'})
    assert response.status_code == 201
    # Hacemos la aserción más flexible
    assert "exitosamente" in response.get_json()['mensaje'].lower()

# Prueba actualizar un libro
def test_actualizar_libro(app_cliente):
    datos_actualizados = {
```

```
"titulo": "Libro actualizado",
"precio_compra": 120,
"precio_venta": 180,
"cantidad_actual": 15
}
response = app_cliente.put('/Libros/1234567890',
                           json=datos_actualizados,
                           headers={'Authorization': 'Basic
Y2FtaWxhOmNhMjAwNA==', 'Content-Type': 'application/json'})
assert response.status_code == 200
# Hacemos la aserción más flexible
assert "exitosamente" in response.get_json()['mensaje'].lower()

# Prueba eliminar un libro
def test_eliminar_libro(app_cliente):
    response = app_cliente.delete('/Libros/1234567890',
headers={'Authorization': 'Basic Y2FtaWxhOmNhMjAwNA=='})
    assert response.status_code == 200
    # Hacemos la aserción más flexible
    assert "exitosamente" in response.get_json()['mensaje'].lower()

# Prueba registrar una transacción
def test_registrar_transaccion(app_cliente):
    transaccion_data = {
        "tipo_transaccion": 1, # Venta
        "ISBN": "9788420471839", # ISBN del libro existente
        "cantidad": 1
    }
    response = app_cliente.post('/add_transaction',
                                json=transaccion_data,
                                headers={'Authorization': 'Basic
Y2FtaWxhOmNhMjAwNA==', 'Content-Type': 'application/json'})
    assert response.status_code == 201
    # Hacemos la aserción más flexible
    assert "exitosamente" in response.get_json()['mensaje'].lower()

# Prueba obtener las transacciones
def test_obtener_transacciones(app_cliente):
    response = app_cliente.get('/Transacciones', headers={'Authorization':
'Basic Y2FtaWxhOmNhMjAwNA=='})
    assert response.status_code == 200
    data = response.get_json()
    assert isinstance(data, list)
```

```
# Removemos la verificación de longitud ya que puede estar vacía
inicialmente

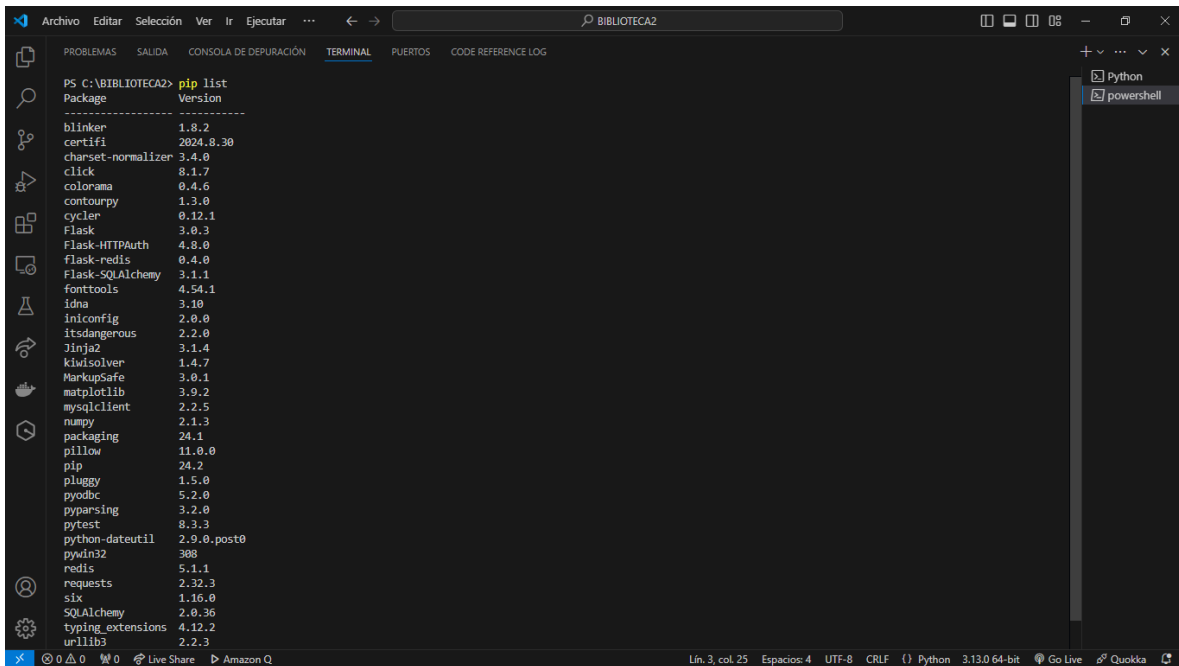
# Prueba obtener el estado de la caja
def test_obtener_estado_caja(app_cliente):
    response = app_cliente.get('/estado_caja', headers={'Authorization':
'Basic Y2FtaWxhOmNhMjAwNA=='})
    assert response.status_code == 200
    data = response.get_json()
    assert 'estado_caja' in data

# Prueba de error de servidor - modificada para probar un caso más
realista
def test_cliente_error_servidor(app_cliente):
    response = app_cliente.get('/ruta_inexistente')
    assert response.status_code == 404
```

El test de biblioteca actúa como si fuera un cliente de la vida real es decir hace peticiones simultaneas al cliente servidor para validar la comunicación como si fuera que el sistema esté en funcionamiento en un caso de la vida real. Esta acta como disparador y a su misma vez valida el funcionamiento del aplicativo. hicimos dos tipos de pruebas para el Python una manual que es ejecutar el código parte por parte y la otra automática, empezaremos con la automática y después con la manual.

PRUEBA AUTOMATICA ARCHIVO PYTHON

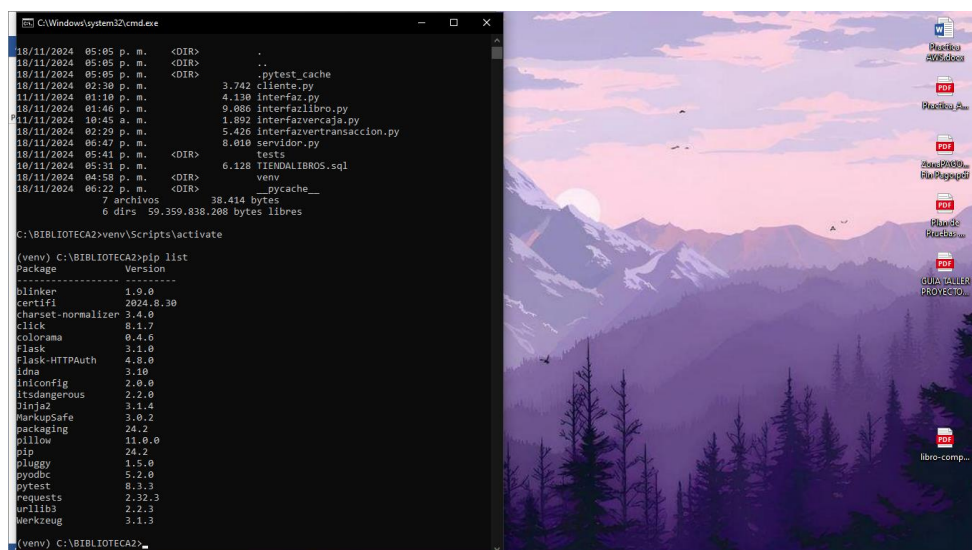
Dentro de nuestra prueba usaremos nuestro archivo test_biblioteca.py esta la usamos con ayuda de un pytest. Su función es hacer una prueba a tiempo real a nuestro servidor verdad. Pero al momento nos genero muchos errores tales como estos. Nuestro programa ejecutaba bien dentro de un ambiente contralo como en el visual estudio code ya que este contaba con todas sus librerías descargadas y todo como lo vemos en imagen.



```

PS C:\BIBLIOTECA2> pip list
Package Version
-----
blinker 1.8.2
certifi 2024.8.30
charset-normalizer 3.4.0
click 8.1.7
colorama 0.4.6
contourpy 1.3.0
cycler 0.12.1
Flask 3.0.3
Flask-HTTPAuth 4.8.0
flask-redis 0.4.0
Flask-SQLAlchemy 3.1.1
fonttools 4.54.1
idna 3.10
iniconfig 2.0.0
itsdangerous 2.2.0
Jinja2 3.1.4
kiwisolver 1.4.7
MarkupSafe 3.0.1
matplotlib 3.9.2
mysqlclient 2.2.5
numpy 2.1.3
packaging 24.1
pillow 11.0.0
pip 24.2
pluggy 1.5.0
pyodbc 5.2.0
pyparsing 3.2.0
pytest 8.3.3
python-dateutil 2.9.0.post0
pywin32 308
redis 5.1.1
requests 2.32.3
six 1.16.0
SQLAlchemy 2.0.36
typing_extensions 4.12.2
urllib3 2.2.3
  
```

Pero para recordar que cuando manejamos un ambiente virtual por medio de consola CDM estas importaciones o librerías no están hay como es un ambiente virtual es algo nuevo sin nada entonces tuvimos que instalar todas las librerías dentro del ambiente virtual para que ejecutara.



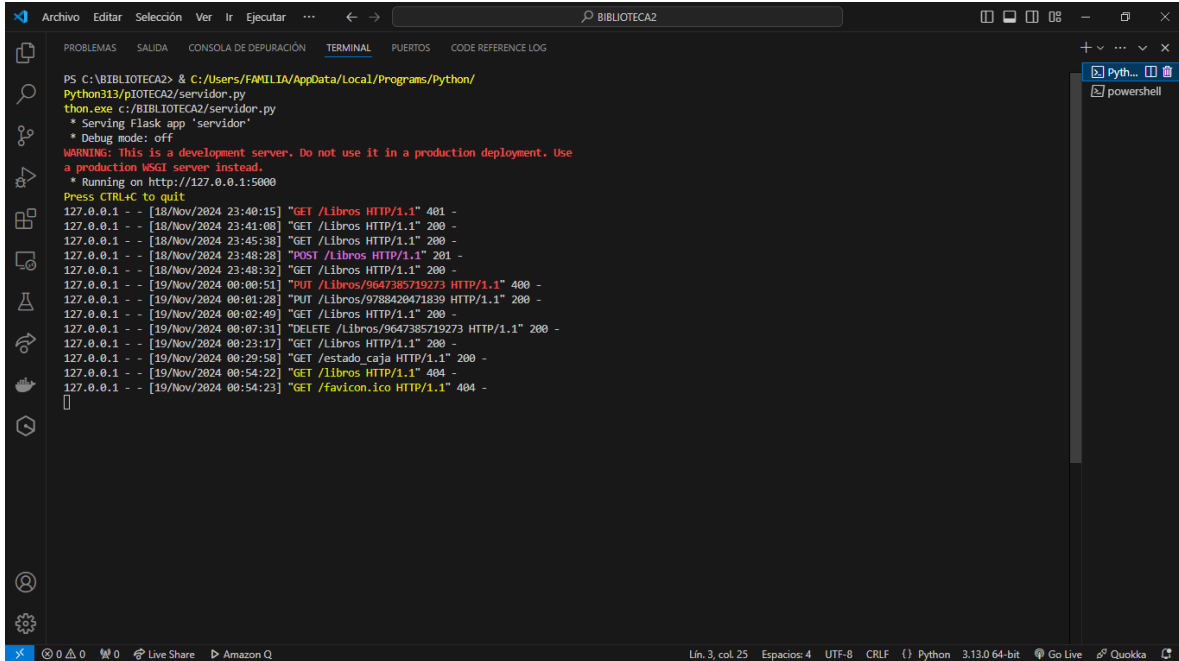
```

C:\Windows\system32\cmd.exe
18/11/2024 05:05 p. m. <DIR> .
18/11/2024 05:05 p. m. <DIR> .\pytest_cache
18/11/2024 05:05 p. m. <DIR> .\3.742 cliente.py
18/11/2024 02:30 p. m. 4.130 interfaz.py
18/11/2024 01:10 p. m. 9.006 interfazlibro.py
18/11/2024 01:46 p. m. 1.802 interfazvercaja.py
18/11/2024 10:45 a. m. 5.426 interfazvertransaccion.py
18/11/2024 02:29 p. m. 8.010 servidor.py
18/11/2024 06:47 p. m. <DIR> tests
18/11/2024 05:41 p. m. 6.128 TIENDALIBROS.sql
18/11/2024 04:58 p. m. <DIR> venv
18/11/2024 06:22 p. m. <DIR> .\pycache__
18/11/2024 06:22 p. m. 7 archivos 38.414 bytes
18/11/2024 06:22 p. m. 6 dirs 59.359.838.208 bytes libres

C:\BIBLIOTECA2>venv\Scripts\activate

(venv) C:\BIBLIOTECA2>pip list
Package Version
-----
blinker 1.9.0
certifi 2024.8.30
charset-normalizer 3.4.0
click 8.1.7
colorama 0.4.6
Flask 3.1.0
Flask-HTTPAuth 4.8.0
idna 3.10
iniconfig 2.0.0
itsdangerous 2.2.0
Jinja2 3.1.4
MarkupSafe 3.0.2
packaging 24.2
pillow 11.0.0
pip 24.2
pluggy 1.5.0
pyodbc 5.2.0
pytest 8.3.3
requests 2.32.3
urllib3 2.2.3
werkzeug 3.1.3
  
```

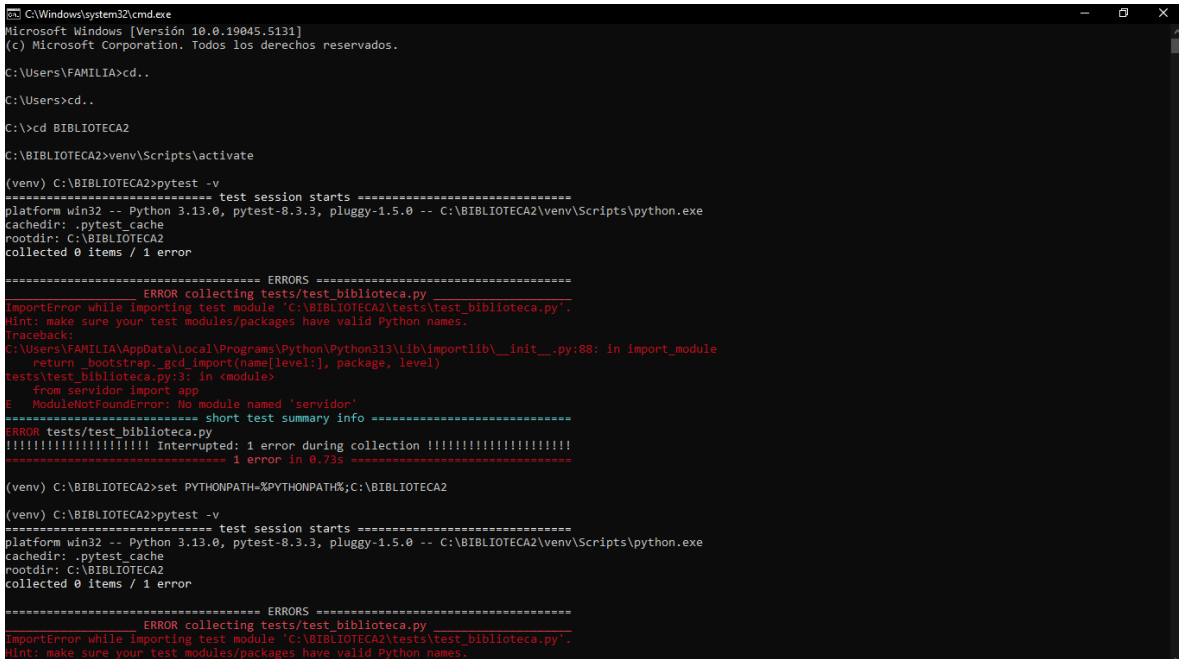

Y otro error que nos paso solo en este parte de inicio es que por el afán de poder concluir la prueba teníamos muchos errores al ejecutar y era por que como vamos a realizar una prueba aun servidor si este no se encuentra activo, no tiene lógica verdad, entonces recordar siempre que este tipo de pruebas deben de contar con el servidor activo, esto como recordatorio.



```

PS C:\BIBLIOTECA2> & C:/Users/FAMILIA/AppData/Local/Programs/Python/Python313/pip.exe install flask
Python313\pip.exe install flask
* Serving Flask app "servidor"
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - [18/Nov/2024 23:40:15] "GET /Libros HTTP/1.1" 401 -
127.0.0.1 - [18/Nov/2024 23:41:08] "GET /Libros HTTP/1.1" 200 -
127.0.0.1 - [18/Nov/2024 23:45:38] "GET /Libros HTTP/1.1" 200 -
127.0.0.1 - [18/Nov/2024 23:48:28] "POST /Libros HTTP/1.1" 201 -
127.0.0.1 - [18/Nov/2024 23:48:32] "GET /Libros HTTP/1.1" 200 -
127.0.0.1 - [19/Nov/2024 00:00:51] "PUT /Libros/9647385719273 HTTP/1.1" 400 -
127.0.0.1 - [19/Nov/2024 00:01:28] "PUT /Libros/9788428471839 HTTP/1.1" 200 -
127.0.0.1 - [19/Nov/2024 00:02:49] "GET /Libros HTTP/1.1" 200 -
127.0.0.1 - [19/Nov/2024 00:07:31] "DELETE /Libros/9647385719273 HTTP/1.1" 200 -
127.0.0.1 - [19/Nov/2024 00:23:17] "GET /Libros HTTP/1.1" 200 -
127.0.0.1 - [19/Nov/2024 00:29:58] "GET /estado_caja HTTP/1.1" 200 -
127.0.0.1 - [19/Nov/2024 00:54:22] "GET /Libros HTTP/1.1" 404 -
127.0.0.1 - [19/Nov/2024 00:54:23] "GET /favicon.ico HTTP/1.1" 404 -
  
```

El servidor ¿tiene que estar activo en el ambiente del CDM? No, no es necesario por que para eso este se ejecuta en el ambiente controlado de visual estudio code, el ambiente virtual que tenemos en la consola CMD es solo para la ejecución del testeio. Ya con estos problemas solucionados entro el dolor de cabeza con los disparadores del servidor



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\FAMILIA>cd..
C:\Users>cd..
C:\>cd BIBLIOTECA2
C:\BIBLIOTECA2>venv\Scripts\activate

(venv) C:\BIBLIOTECA2>pytest -v
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\BIBLIOTECA2\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\BIBLIOTECA2
collected 0 items / 1 error

===== ERRORS =====
ERROR collecting tests/test_biblioteca.py
ImportError while importing test module 'C:\BIBLIOTECA2\tests\test_biblioteca.py'.
Hint: make sure your test modules/packages have valid Python names.
Traceback:
C:\Users\FAMILIA\AppData\Local\Programs\Python\Python313\Lib\importlib\_init_.py:88: in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
tests\test_biblioteca.py:3: in <module>
    from servidor import app
E ModuleNotFoundError: No module named 'servidor'

===== short test summary info =====
ERROR tests/test_biblioteca.py
!!!!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!!
===== 1 error in 0.73s =====

(venv) C:\BIBLIOTECA2>set PYTHONPATH=%PYTHONPATH%;C:\BIBLIOTECA2
(venv) C:\BIBLIOTECA2>pytest -v
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\BIBLIOTECA2\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\BIBLIOTECA2
collected 0 items / 1 error

===== ERRORS =====
ERROR collecting tests/test_biblioteca.py
ImportError while importing test module 'C:\BIBLIOTECA2\tests\test_biblioteca.py'.
Hint: make sure your test modules/packages have valid Python names.
  
```

Esto se debía a que el sistema no reconocía si el servidor estaba en una sub carpeta o dentro de la carpeta donde se encontraba el test, eso fue un poco fácil de arreglar ya que solo fue acomodar el servidor en la posición correcta es decir colocar el servidor en la carpeta principal para que este no lo detecte como en una carpeta secundaria.

```
(venv) C:\BIBLIOTECA2>set PYTHONPATH=%PYTHONPATH%;C:\BIBLIOTECA2

(venv) C:\BIBLIOTECA2>pytest -v
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\BIBLIOTECA2\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\BIBLIOTECA2
collected 0 items / 1 error

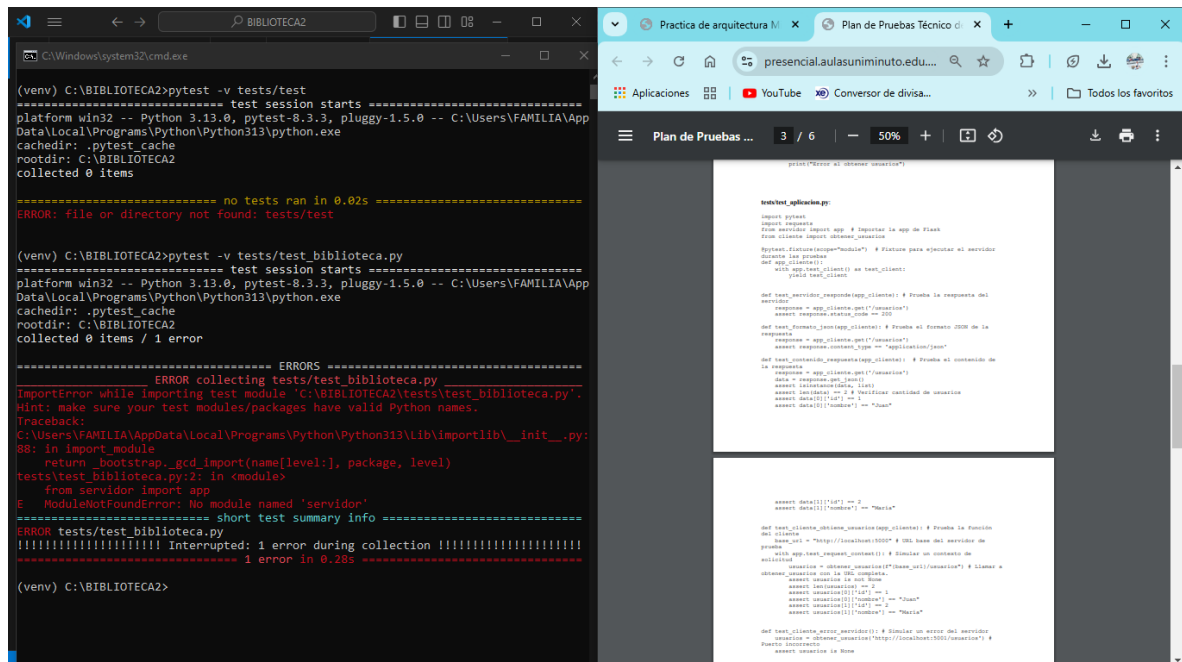
===== ERRORS =====
ERROR collecting tests/test_biblioteca.py
ImportError while importing test module 'C:\BIBLIOTECA2\tests\test_biblioteca.py'.
Hint: make sure your test modules/packages have valid Python names.
Traceback:
C:\Users\FAMILIA\AppData\Local\Programs\Python\Python313\Lib\importlib\__init__.py:88: in import_module
    return bootstrap._gcd_import(name[level:], package, level)
tests\test_biblioteca.py:3: in <module>
    from servidor import app
servidor.py:3: in <module>
    import pyodbc
E ModuleNotFoundError: No module named 'pyodbc'

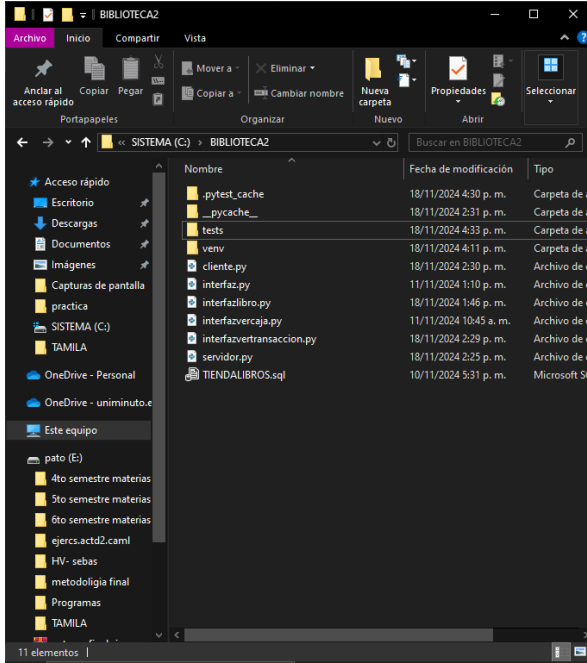
===== short test summary info =====
ERROR tests/test_biblioteca.py
!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!!!!
===== 1 error in 0.92s =====

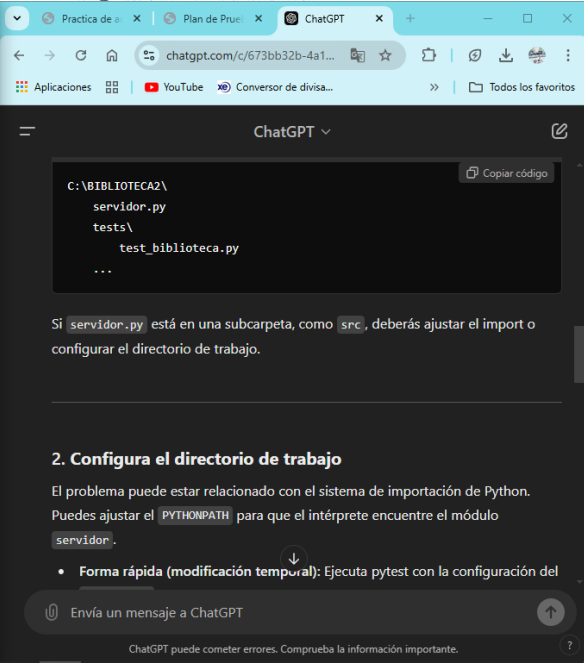
(venv) C:\BIBLIOTECA2>set PYTHONPATH=%PYTHONPATH%;C:\BIBLIOTECA2
```

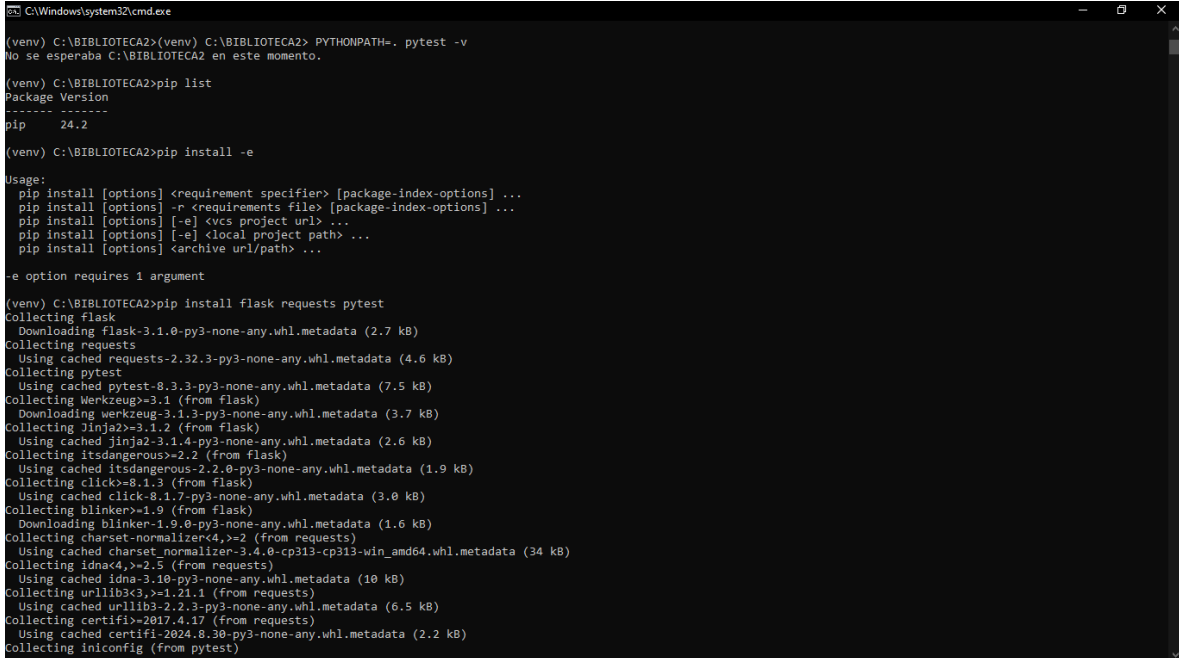
Pero entro otro error, nosotros descargamos todas las librerías si menos la que conectar el servidor con el Python y claro esto genero un error de reconocimiento y esto causo problemas ahorita lo explicamos fácil, pero en su momento fue casi una corta venas al no saber que pasaba, pero solo con un “pip install pyodbc” y asunto arreglado.

Pero a pesar de esto aún seguimos con los errores, pero no iniciaba el test y esto se debía a mala conexión se sintió como realizar una conexión de cables, si haces una mala conexión todo valió queso, como nos pasó a nosotros y no solo 1 si no muchas veces como se muestran en las siguientes imágenes al punto fue que pedimos ayuda de las IA pero casi que no podemos succionar la conexión.









```

C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\FAMILIA>cd..
C:\Users>cd..
C:\>cd BIBLIOTECAS
C:\BIBLIOTECAS>venv\Scripts\activate

(venv) C:\BIBLIOTECAS>pytest -v
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\BIBLIOTECAS\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\BIBLIOTECAS
collected 0 items / 1 error

===== ERRORS =====
ERROR collecting tests/test_biblioteca.py
ImportError while importing test module 'C:\BIBLIOTECAS\tests\test_biblioteca.py'.
Hint: make sure your test modules/packages have valid Python names.
Traceback:
C:\Users\FAMILIA\AppData\Local\Programs\Python\Python313\Lib\importlib\__init__.py:88: in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
tests/test_biblioteca.py:3: in <module>
    from servidor import app
E ModuleNotFoundError: No module named 'servidor'

===== Short test summary info =====
ERROR tests/test_biblioteca.py
!!!!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!!
===== 1 error in 0.73s =====

(venv) C:\BIBLIOTECAS>set PYTHONPATH=%PYTHONPATH%;C:\BIBLIOTECAS
(venv) C:\BIBLIOTECAS>pytest -v
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\BIBLIOTECAS\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\BIBLIOTECAS
collected 0 items / 1 error

===== ERRORS =====
ERROR collecting tests/test_biblioteca.py
ImportError while importing test module 'C:\BIBLIOTECAS\tests\test_biblioteca.py'.
Hint: make sure your test modules/packages have valid Python names.

```

Pero luego un momento que la prueba reacciona, si así es el test respondió, pero como la vida no es color de rosa, salieron mas errores, pero ya estábamos más aliviados porque reaccionaba así que fue muy esperanzador ver este resultado a pesar de que se vea mas rojo que verde

```

C:\Windows\system32\cmd.exe
===== 1 error in 1.24s =====

(venv) C:\BIBLIOTECAS>pip install pyodbc
Collecting pyodbc
  Using cached pyodbc-5.2.0-cp313-cp313-win_amd64.whl.metadata (2.8 kB)
Using cached pyodbc-5.2.0-cp313-cp313-win_amd64.whl (69 kB)
Installing collected packages: pyodbc
Successfully installed pyodbc-5.2.0

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(venv) C:\BIBLIOTECAS>
(venv) C:\BIBLIOTECAS>pytest -v
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\BIBLIOTECAS\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\BIBLIOTECAS
collected 10 items

tests/test_biblioteca.py::test_servidor_responde FAILED [ 10%]
tests/test_biblioteca.py::test_formato_json FAILED [ 20%]
tests/test_biblioteca.py::test_obtener_libros PASSED [ 30%]
tests/test_biblioteca.py::test_agregar_libro PASSED [ 40%]
tests/test_biblioteca.py::test_actualizar_libro PASSED [ 50%]
tests/test_biblioteca.py::test_eliminar_libro PASSED [ 60%]
tests/test_biblioteca.py::test_registrar_transacción FAILED [ 70%]
tests/test_biblioteca.py::test_obtener_transacciones PASSED [ 80%]
tests/test_biblioteca.py::test_obtener_estado_caja PASSED [ 90%]
tests/test_biblioteca.py::test_cliente_error_servidor FAILED [100%]

===== FAILURES =====
test_servidor_responde
app_cliente = <FlaskClient <Flask 'servidor'>>

def test_servidor_responde(app_cliente):
    response = app_cliente.get('/libros')
    assert response.status_code == 200
    assert 401 == 200
    + where 401 = <WrapperTestResponse streamed [401 UNAUTHORIZED]>.status_code

tests/test_biblioteca.py:14: AssertionError
test_formato_json
app_cliente = <FlaskClient <Flask 'servidor'>>

```

Como podemos ver en la imagen los errores que presenta son los siguientes:

- Respuesta del servidor == **FAILED**
- Formato JSON == **FAILED**
- Obtener libros == **PASSED**

- Agregar libros == PASSED
- Actualizar libros == PASSED
- Eliminar libros == PASSED
- Registrar transacción == FAILED
- Obtener transacciones == PASSED
- Estado de caja == PASSED
- Error servidor == FAILED

Cuando notamos esto teníamos un grave problema porque solo teníamos el 60% de funcionalidad, pero de que sirve esto si no es un 100% así que nos pusimos a ver el código paso a paso que pasaba y que mensaje arrojaba y fue un mensaje demasiado extenso que parecía 3 capítulos de un libro

```
C:\Windows\system32\cmd.exe
----- 1 error in 1.24s -----
(venv) C:\BIBLIOTECAS2>pip install pyodbc
Collecting pyodbc
  Using cached pyodbc-5.2.0-cp313-cp313-win_amd64.whl.metadata (2.8 kB)
Using cached pyodbc-5.2.0-cp313-cp313-win_amd64.whl (69 kB)
Installing collected packages: pyodbc
Successfully installed pyodbc-5.2.0

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(venv) C:\BIBLIOTECAS2>
(venv) C:\BIBLIOTECAS2>pytest -v
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\BIBLIOTECAS2\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\BIBLIOTECAS2
collected 10 items

tests/test_biblioteca.py::test_servidor_responde FAILED [ 10%]
tests/test_biblioteca.py::test_formato_json FAILED [ 20%]
tests/test_biblioteca.py::test_obtener_libros PASSED [ 30%]
tests/test_biblioteca.py::test_agregar_libro PASSED [ 40%]
tests/test_biblioteca.py::test_actualizar_libro PASSED [ 50%]
tests/test_biblioteca.py::test_eliminar_libro PASSED [ 60%]
tests/test_biblioteca.py::test_registrar_transaccion FAILED [ 70%]
tests/test_biblioteca.py::test_obtener_transacciones PASSED [ 80%]
tests/test_biblioteca.py::test_obtener_estado_caja PASSED [ 90%]
tests/test_biblioteca.py::test_cliente_error_servidor FAILED [100%]

===== FAILURES =====
test_servidor_responde
app_cliente = <FlaskClient <Flask 'servidor'>>

    def test_servidor_responde(app_cliente):
        response = app_cliente.get('/Libros')
        assert response.status_code == 200
E       assert 401 == 200
E       + where 401 = <WrapperTestResponse streamed [401 UNAUTHORIZED]>.status_code

tests/test_biblioteca.py:14: AssertionError
test_formato_json
app_cliente = <FlaskClient <Flask 'servidor'>>
```

```

C:\Windows\system32\cmd.exe
address = ('localhost', 5001), timeout = None, source_address = None
socket_options = [(6, 1, 1)]

def create_connection(
    address: tuple[str, int],
    timeout: _TYPE_TIMEOUT = _DEFAULT_TIMEOUT,
    source_address: tuple[str, int] | None = None,
    socket_options: _TYPE_SOCKET_OPTIONS | None = None,
) -> socket.socket:
    """Connect to *address* and return the socket object.

    Convenience function. Connect to *address* (a 2-tuple ``(host,
    port)``) and return the socket object. Passing the optional
    *timeout* parameter will set the timeout on the socket instance
    before attempting to connect. If no *timeout* is supplied, the
    global default timeout setting returned by :func:`socket.getdefaulttimeout`
    for the socket to bind as a source address before making the connection.
    An host of ``*`` or port 0 tells the OS to use the default.
    """

    host, port = address
    if host.startswith("["):
        host = host.strip("[")
    err = None

    # Using the value from allowed_gai_family() in the context of getaddrinfo lets
    # us select whether to work with IPv4 DNS records, IPv6 records, or both.
    # The original create_connection function always returns all records.
    family = allowed_gai_family()

    try:
        host.encode("idna")
    except UnicodeError:
        raise LocationParseError(f"[{host}]", label empty or too long") from None

    for res in socket.getaddrinfo(host, port, family, socket.SOCK_STREAM):
        af, socktype, proto, canonname, sa = res
        sock = None
        try:
            sock = socket.socket(af, socktype, proto)

            # If provided, set socket level options before connecting.
            _set_socket_options(sock, socket_options)

            if timeout is not _DEFAULT_TIMEOUT:
                sock.settimeout(timeout)
  
```

```

C:\Windows\system32\cmd.exe
tests\test_biblioteca.py:19: AssertionError
      test_registrar_transaccion
app_cliente = <FlaskClient <Flask 'servidor'>>

def test_registrar_transaccion(app_cliente):
    transaccion_data = {
        "tipo transaccion": 1, # Venta
        "ISBN": "1234567890",
        "cantidad": 1
    }
    response = app_cliente.post('/add_transaction', json=transaccion_data, auth=('camila', 'ca2004'))
    assert response.status_code == 201
    assert 500 == 201
    + where 500 = <WrapperTestResponse streamed [500 INTERNAL SERVER ERROR]>.status_code

tests\test_biblioteca.py:67: AssertionError
      test_cliente_error_servidor

self = <urllib3.connection.HTTPConnection object at 0x000001437DF46780>

def new_conn(self) -> socket.socket:
    """Establish a socket connection and set nodelay settings on it.

    :return: New socket connection.
    """
    try:
        sock = connection.create_connection(
            (self.dns_host, self.port),
            self.timeout,
            source_address=self.source_address,
            socket_options=self.socket_options,
        )
    except:
        raise err

venv\Lib\site-packages\urllib3\connection.py:199:
venv\Lib\site-packages\urllib3\util\connection.py:85: in create_connection
    raise err
-----
address = ('localhost', 5001), timeout = None, source_address = None
socket_options = [(6, 1, 1)]

def create_connection(
    address: tuple[str, int],
    timeout: _TYPE_TIMEOUT = _DEFAULT_TIMEOUT,
  
```

```
C:\Windows\system32\cmd.exe
> sock.bind(source_address)
> sock.connect(sa)
ConnectionRefusedError: [WinError 10061] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión

venv\Lib\site-packages\urllib3\util\connection.py:73: ConnectionRefusedError

The above exception was the direct cause of the following exception:

self = <urllib3.connectionpool.HTTPConnectionPool object at 0x000001437DF45D30>
method = 'GET', url = '/Libros', body = None
headers = {'User-Agent': 'python-requests/2.32.3', 'Accept-Encoding': 'gzip, deflate', 'Accept': '*//*', 'Connection': 'keep-alive'}
retries = Retry(total=0, connect=None, read=False, redirect=None, status=None)
redirect = False, assert_same_host = False
timeout = Timeout(connect=None, read=None, total=None), pool_timeout = None
release_conn = False, chunked = False, body_pos = None, preload_content = False
decode_content = False, response_kw = {}
parsed_url = Url(scheme=None, auth=None, host=None, port=None, path='/Libros', query=None, fragment=None)
destination_scheme = None, conn = None, release_this_conn = True
http_tunnel_required = False, err = None, clean_exit = False

def urlopen( # type: ignore[override]
    self,
    method: str,
    url: str,
    body: _TYPE_BODY | None = None,
    headers: typing.Mapping[str, str] | None = None,
    retries: Retry | bool | int | None = None,
    redirect: bool = True,
    assert_same_host: bool = True,
    timeout: _TYPE_TIMEOUT = _DEFAULT_TIMEOUT,
    pool_timeout: int | None = None,
    release_conn: bool | None = None,
    chunked: bool = False,
    body_pos: _TYPE_BODY_POSITION | None = None,
    preload_content: bool = True,
    decode_content: bool = True,
    **response_kw: typing.Any,
) -> BaseHTTPResponse:
    """
    Get a connection from the pool and perform an HTTP request. This is the
    lowest level call for making a request, so you'll need to specify all
    the raw details.

    .. note::

        More commonly, it's appropriate to use a convenience method
    """
```

```
C:\Windows\system32\cmd.exe

More commonly, it's appropriate to use a convenience method
such as :meth:`request`.

.. note::

    'release_conn' will only behave as expected if
    'preload_content=False' because we want to make
    'preload_content=False' the default behaviour someday soon without
    breaking backwards compatibility.

:param method:
    HTTP request method (such as GET, POST, PUT, etc.)

:param url:
    The URL to perform the request on.

:param body:
    Data to send in the request body, either :class:`str`, :class:`bytes`,
    an iterable of :class:`str`/:class:`bytes`, or a file-like object.

:param headers:
    Dictionary of custom headers to send, such as User-Agent,
    If-None-Match, etc. If None, pool headers are used. If provided,
    these headers completely replace any pool-specific headers.

:param retries:
    Configure the number of retries to allow before raising a
    :class:`~urllib3.exceptions.MaxRetryError` exception.

    If ``None`` (default) will retry 3 times, see ``Retry.DEFAULT``. Pass a
    over different types of retries.
    Pass an integer number to retry connection errors that many times,
    but no other types of errors. Pass zero to never retry.

    If ``False``, then retries are disabled and any exception is raised
    immediately. Also, instead of raising a MaxRetryError on redirects,
    the redirect response will be returned.

:type retries: :class:`~urllib3.util.retry.Retry`, False, or an int.

:param redirect:
    If True, automatically handle redirects (status codes 301, 302,
    303, 307, 308). Each redirect counts as a retry. Disabling retries
    will disable redirect, too.
```

```
C:\Windows\system32\cmd.exe
will disable redirect, too.

:param assert_same_host:
    If ``True``, will make sure that the host of the pool requests is
    consistent else will raise HostChangedError. When ``False``, you can
    use the pool on an HTTP proxy and request foreign hosts.

:param timeout:
    If specified, overrides the default timeout for this one
    request. It may be a float (in seconds) or an instance of
    :class:`urllib3.util.Timeout`.

:param pool_timeout:
    If set and the pool is set to block=True, then this method will
    block for ``pool_timeout`` seconds and raise EmptyPoolError if no
    connection is available within the time period.

:param bool preload_content:
    If True, the response's body will be preloaded into memory.

:param bool decode_content:
    If True, will attempt to decode the body based on the
    'content-encoding' header.

:param release_conn:
    If False, then the urlopen call will not release the connection
    back into the pool once a response is received (but will release if
    you read the entire contents of the response such as when
    'preload_content=True'). This is useful if you're not preloading
    the response's content immediately. You will need to call
    ``r.release_conn()`` on the response ``r`` to return the connection
    back into the pool. If None, it takes the value of ``preload_content``
    which defaults to ``True``.

:param bool chunked:
    If True, urllib3 will send the body using chunked transfer
    encoding. Otherwise, urllib3 will send the body using the standard
    content-length form. Defaults to False.

:param int body_pos:
    Position to seek to in file-like body in the event of a retry or
    redirect. Typically this won't need to be set because urllib3 will
    auto-populate the value when needed.
"""
parsed_url = parse_url(url)
destination_scheme = parsed_url.scheme
```

```
C:\Windows\system32\cmd.exe
if assert_same_host and not self.is_same_host(url):
    raise HostChangedError(self, url, retries)

# Ensure that the URL we're connecting to is properly encoded
if url.startswith("/"):
    url = to_str(_encode_target(url))
else:
    url = to_str(parsed_url.url)

conn = None

# Track whether 'conn' needs to be released before
# returning/raising/recursing. Update this variable if necessary, and
# leave 'release_conn' constant throughout the function. That way, if
# the function recurses, the original value of 'release_conn' will be
# passed down into the recursive call, and its value will be respected.
#
# See issue #651 [1] for details.
#
# [1] <https://github.com/urllib3/urllib3/issues/651>
release_this_conn = release_conn

http_tunnel_required = connection_requires_http_tunnel(
    self.proxy, self.proxy_config, destination_scheme
)

# Merge the proxy headers. Only done when not using HTTP CONNECT. We
# have to copy the headers dict so we can safely change it without those
# changes being reflected in anyone else's copy.
if not http_tunnel_required:
    headers = headers.copy() # type: ignore[attr-defined]
    headers.update(self.proxy_headers) # type: ignore[union-attr]

# Must keep the exception bound to a separate variable or else Python 3
# complains about UnboundLocalError.
err = None

# Keep track of whether we cleanly exited the except block. This
# ensures we do proper cleanup in finally.
clean_exit = False

# Rewind body position, if needed. Record current position
# for future rewinds in the event of a redirect/retry.
body_pos = set_file_position(body, body_pos)

try:
```



```
C:\Windows\system32\cmd.exe
**response_kw,
)
venv\Lib\site-packages\urllib3\connectionpool.py:789:
venv\Lib\site-packages\urllib3\connectionpool.py:495: in _make_request
    conn.request(
venv\Lib\site-packages\urllib3\connection.py:441: in request
    self.endheaders()
C:\Users\FAMILIA\AppData\Local\Programs\Python\Python313\Lib\http\client.py:1331: in endheaders
    self._send_output(message_body, encode_chunked=encode_chunked)
C:\Users\FAMILIA\AppData\Local\Programs\Python\Python313\Lib\http\client.py:1091: in _send_output
    self.send(msg)
C:\Users\FAMILIA\AppData\Local\Programs\Python\Python313\Lib\http\client.py:1035: in send
    self.connect()
venv\Lib\site-packages\urllib3\connection.py:279: in connect
    self.sock = self._new_conn()
-----
self = <urllib3.connection.HTTPConnection object at 0x00001437DF467B0>

def _new_conn(self) -> socket.socket:
    """Establish a socket connection and set nodelay settings on it.
    :return: New socket connection.
    """
    try:
        sock = connection.create_connection(
            (self._dns_host, self.port),
            self.timeout,
            source_address=self.source_address,
            socket_options=self.socket_options,
        )
    except socket.gaierror as e:
        raise NameResolutionError(self.host, self, e) from e
    except SocketTimeout as e:
        raise ConnectTimeoutError(
            self,
            f"Connection to {self.host} timed out. (connect timeout={self.timeout})",
        ) from e
    except OSError as e:
        raise NewConnectionError(
            self, f"Failed to establish a new connection: {e}"
        ) from e
urllib3.exceptions.NewConnectionError: <urllib3.connection.HTTPConnection object at 0x00001437DF467B0>: Failed to establish a new connection: [WinError 100
```

```
C:\Windows\system32\cmd.exe
> raise NewConnectionError(
    self, f"Failed to establish a new connection: {e}"
) from e
E urllib3.exceptions.NewConnectionError: <urllib3.connection.HTTPConnection object at 0x00001437DF467B0>: Failed to establish a new connection: [WinError 100
61] No se puede establecer una conexión ya que el equipo de destino denegó expresamente dicha conexión

venv\Lib\site-packages\urllib3\connection.py:214: NewConnectionError

The above exception was the direct cause of the following exception:

self = <requests.adapters.HTTPAdapter object at 0x00001437DE67110>
request = <PreparedRequest [GET]>, stream = False
timeout = Timeout(connect=None, read=None, total=None), verify = True, cert = None
proxies = OrderedDict()

def send(
    self, request, stream=False, timeout=None, verify=True, cert=None, proxies=None
):
    """Sends PreparedRequest object. Returns Response object.

    :param request: The :class:`PreparedRequest` object being sent.
    :param stream: (optional) Whether to stream the request content.
    :param timeout: (optional) How long to wait for the server to send
        data before giving up, as a float, or a :ref:`(connect timeout,
        read timeout)` tuple.
    :type timeout: float or tuple or urllib3.Timeout object
    :param verify: (optional) Either a boolean, in which case it controls whether
        we verify the server's TLS certificate, or a string, in which case it
        must be a path to a CA bundle to use.
    :param cert: (optional) Any user-provided SSL certificate to be trusted.
    :param proxies: (optional) The proxies dictionary to apply to the request.
    :rtype: requests.Response
    """

    try:
        conn = self.get_connection_with_tls_context(
            request, verify, proxies=proxies, cert=cert
        )
    except LocationValueError as e:
        raise InvalidURL(e, request=request)

    self.cert_verify(conn, request.url, verify, cert)
    url = self.request_url(request, proxies)
    self.add_headers(
        request,
        stream=stream,
```

```
C:\Windows\system32\cmd.exe

)

chunked = not (request.body is None or "Content-Length" in request.headers)
if isinstance(timeout, tuple):
    try:
        connect, read = timeout
        timeout = TimeoutSauce(connect=connect, read=read)
    except ValueError:
        raise ValueError(
            f"Invalid timeout {timeout}. Pass a (connect, read) timeout tuple, "
            f"or a single float to set both timeouts to the same value."
        )
elif isinstance(timeout, TimeoutSauce):
    pass
else:
    timeout = TimeoutSauce(connect=timeout, read=timeout)

try:
    resp = conn.urlopen(
        method=request.method,
        url=url,
        body=request.body,
        headers=request.headers,
        redirect=False,
        assert_same_host=False,
        preload_content=False,
        decode_content=False,
        retries=self.max_retries,
        timeout=timeout,
        chunked=chunked,
    )

venv\Lib\site-packages\requests\adapters.py:667:
venv\Lib\site-packages\urllib3\connectionpool.py:843: in urlopen
    retries = retries.increment(
-----
self = Retry(total=0, connect=None, read=False, redirect=None, status=None)
method = 'GET', url = '/Libros', response = None
error = NewConnectionError('<urllib3.connection.HTTPConnection object at 0x000001437DF467B0>: Failed to establish a new connec...: [WinError 10061] No se puede establec
er una conexión ya que el equipo de destino denegó expresamente dicha conexión')
_pool = <urllib3.connectionpool.HTTPConnectionPool object at 0x000001437DF45D30>
_stacktrace = <traceback object at 0x000001437DF9D780>

def increment(
```

```
C:\Windows\system32\cmd.exe

_stacktrace = <traceback object at 0x000001437DF9D780>

def increment(
    self,
    method: str | None = None,
    url: str | None = None,
    response: BaseHTTPResponse | None = None,
    error: Exception | None = None,
    _pool: ConnectionPool | None = None,
    _stacktrace: TracebackType | None = None,
) -> Self:
    """Return a new Retry object with incremented retry counters.

    :param response: A response object, or None, if the server did not
        return a response.
    :type response: :class:`~urllib3.response.BaseHTTPResponse`
    :param Exception error: An error encountered during the request, or
        None if the response was received successfully.

    :return: A new ``Retry`` object.
    """
    if self.total is False and error:
        # Disabled, indicate to re-raise the error.
        raise reraise(type(error), error, _stacktrace)

    total = self.total
    if total is not None:
        total -= 1

    connect = self.connect
    read = self.read
    redirect = self.redirect
    status_count = self.status
    other = self.other
    cause = "unknown"
    status = None
    redirect_location = None

    if error and self._is_connection_error(error):
        # Connect retry?
        if connect is False:
            raise reraise(type(error), error, _stacktrace)
        elif connect is not None:
            connect -= 1
        else:
            connect = 1

    elif error and self._is_read_error(error):
```

```
C:\Windows\system32\cmd.exe
if response_redirect_location:
    redirect_location = response_redirect_location
    status = response.status
else:
    # Incrementing because of a server error like a 500 in
    # status_forcelist and the given method is in the allowed_methods
    cause = ResponseError.GENERIC_ERROR
    if response and response.status:
        if status_count is not None:
            status_count += 1
        cause = ResponseError.SPECIFIC_ERROR.format(status_code=response.status)
    status = response.status

    history = self.history + (
        RequestHistory(method, url, error, status, redirect_location),
    )

    new_retry = self.new(
        total=total,
        connect=connect,
        read=read,
        redirect=redirect,
        status=status_count,
        other=other,
        history=history,
    )

    if new_retry.is_exhausted():
        reason = error or ResponseError(cause)
        raise MaxRetryError(_pool, url, reason) from reason # type: ignore[arg-type]
>       urllib3.exceptions.MaxRetryError: HTTPConnectionPool(host='localhost', port=5001): Max retries exceeded with url: /Libros (Caused by NewConnectionError('<ur
llib3.connection.HTTPConnection object at 0x000001437DF46780>: Failed to establish a new connection: [WinError 10061] No se puede establecer una conexión ya que el equi
po de destino denegó expresamente dicha conexión'))

venv\Lib\site-packages\urllib3\util\retry.py:519: MaxRetryError

During handling of the above exception, another exception occurred:

>     def test_cliente_error_servidor():
>         response = requests.get('http://localhost:5001/Libros') # Puerto incorrecto

tests\test_biblioteca.py:86:
venv\Lib\site-packages\requests\api.py:73: in get
    return request("get", url, params=params, **kwargs)
return request("get", url, params=params, **kwargs)

tests\test_biblioteca.py:86:
venv\Lib\site-packages\requests\api.py:73: in get
    return request("get", url, params=params, **kwargs)
return request("get", url, params=params, **kwargs)
```

```
C:\Windows\system32\cmd.exe
tests\test_biblioteca.py:86:
venv\Lib\site-packages\requests\api.py:73: in get
    return request("get", url, params=params, **kwargs)
return request("get", url, params=params, **kwargs)
venv\Lib\site-packages\requests\api.py:59: in request
    return session.request(method=method, url=url, **kwargs)
venv\Lib\site-packages\requests\sessions.py:589: in request
    resp = self.send(prep, **send_kwargs)
venv\Lib\site-packages\requests\sessions.py:703: in send
    r = adapter.send(request, **kwargs)

-----
self = <requests.adapters.HTTPAdapter object at 0x000001437DE67110>
request = <PreparedRequest [GET]>, stream = False
timeout = Timeout(connect=None, read=None, total=None), verify = True, cert = None
proxies = OrderedDict()

def send(
    self, request, stream=False, timeout=None, verify=True, cert=None, proxies=None
):
    """Sends PreparedRequest object. Returns Response object.

    :param request: The :class:`PreparedRequest ` being sent.
    :param stream: (optional) Whether to stream the request content.
    :param timeout: (optional) How long to wait for the server to send
        data before giving up, as a float, or a :ref:`ref:timeout` tuple.
    :type timeout: float or tuple or urllib3 Timeout object
    :param verify: (optional) Either a boolean, in which case it controls whether
        we verify the server's TLS certificate, or a string, in which case it
        must be a path to a CA bundle to use.
    :param cert: (optional) Any user-provided SSL certificate to be trusted.
    :param proxies: (optional) The proxies dictionary to apply to the request.
    :rtype: requests.Response
    """

    try:
        conn = self.get_connection_with_tls_context(
            request, verify, proxies=proxies, cert=cert
        )
    except LocationValueError as e:
        raise InvalidURL(e, request=request)

    self.cert_verify(conn, request.url, verify, cert)
    url = self.request_url(request, proxies)
    self.add_headers(
```

```
C:\Windows\system32\cmd.exe
request,
stream=stream,
timeout=timeout,
verify=verify,
cert=cert,
proxies=proxies,
)

chunked = not (request.body is None or "Content-Length" in request.headers)
if isinstance(timeout, tuple):
    try:
        connect, read = timeout
        timeout = TimeoutSauce(connect=connect, read=read)
    except ValueError:
        raise ValueError(
            f"Invalid timeout {timeout}. Pass a (connect, read) timeout tuple, "
            f"or a single float to set both timeouts to the same value."
        )
    elif isinstance(timeout, TimeoutSauce):
        pass
    else:
        timeout = TimeoutSauce(connect=timeout, read=timeout)

try:
    resp = conn.urlopen(
        method=request.method,
        url=url,
        body=request.body,
        headers=request.headers,
        redirect=False,
        assert_same_host=False,
        preload_content=False,
        decode_content=False,
        retries=self.max_retries,
        timeout=timeout,
        chunked=chunked,
    )

except (ProtocolError, OSError) as err:
    raise ConnectionError(err, request=request)

except MaxRetryError as e:
    if isinstance(e.reason, ConnectTimeoutError):
        # TODO: Remove this in 3.0.0; see #2811
        if not isinstance(e.reason, NewConnectionError):
            raise ConnectTimeout(e, request=request)

    if isinstance(e.reason, ResponseError):
        raise RetryError(e, request=request)

    if isinstance(e.reason, ProxyError):
        raise ProxyError(e, request=request)

    if isinstance(e.reason, SSLError):
        # This branch is for urllib3 v1.22 and later.
        raise SSLError(e, request=request)

    raise ConnectionError(e, request=request)
```

```
C:\Windows\system32\cmd.exe
try:
    resp = conn.urlopen(
        method=request.method,
        url=url,
        body=request.body,
        headers=request.headers,
        redirect=False,
        assert_same_host=False,
        preload_content=False,
        decode_content=False,
        retries=self.max_retries,
        timeout=timeout,
        chunked=chunked,
    )

except (ProtocolError, OSError) as err:
    raise ConnectionError(err, request=request)

except MaxRetryError as e:
    if isinstance(e.reason, ConnectTimeoutError):
        # TODO: Remove this in 3.0.0; see #2811
        if not isinstance(e.reason, NewConnectionError):
            raise ConnectTimeout(e, request=request)

    if isinstance(e.reason, ResponseError):
        raise RetryError(e, request=request)

    if isinstance(e.reason, ProxyError):
        raise ProxyError(e, request=request)

    if isinstance(e.reason, SSLError):
        # This branch is for urllib3 v1.22 and later.
        raise SSLError(e, request=request)

    raise ConnectionError(e, request=request)

>
requests.exceptions.ConnectionError: HTTPConnectionPool(host='localhost', port=5001): Max retries exceeded with url: /Libros (Caused by NewConnectionError(
'urllib3.connection.HTTPConnection object at 0x0000014370DF467B0': Failed to establish a new connection: [WinError 10061] No se puede establecer una conexión ya que el e
quipo de destino denegó expresamente dicha conexión'))

venv\Lib\site-packages\requests\adapters.py:700: ConnectionError
===== short test summary info =====
FAILED tests/test_biblioteca.py::test_servidor_responde - assert 401 == 200
FAILED tests/test_biblioteca.py::test_formato_json - AssertionError: assert 'text/html; charset=utf-8' == 'application/json'
FAILED tests/test_biblioteca.py::test_registro_transaccion - assert 500 == 201
FAILED tests/test_biblioteca.py::test_cliente_error_servidor - requests.exceptions.ConnectionError: HTTPConnectionPool(host='localhost', port...
===== 4 failed, 6 passed in 5.73s =====
```

Y al final de ese larguero solo nos indicaba que teníamos problemas graves con los disparadores del cliente y servidor, ya que el servidor no se encontraba en la carpeta pero s estaba en ejecución al momento de realizar la entrega de un libro la solución que optamos fue que con ayuda de la IA nos indicara en donde estaban los errores, y la respuesta de la IA fue que en algunas funciones que teníamos en los disparadores de servidores no estaban en el test y algunos que estaban en el test no estaban en el servidor, y el cliente estaba solicitando acciones fantasma que no existen en servidor pero test si y test solicitaba pruebas fantasma porque no estaban dentro del servidor.

Después de mucha paciencia, lloradas y estrés logramos solucionar el error, todo con el fin y propósito de tener un 100% el test lo ejecutamos bajo el ambiente virtual y usando el pytest -v salió este hermoso resultado dando fin a la prueba automática.

```
C:\Windows\system32\cmd.exe
(venv) C:\BIBLIOTECA2>pytest -v
===== test session starts =====
platform win32 -- Python 3.13.0, pytest-8.3.3, pluggy-1.5.0 -- C:\BIBLIOTECA2\venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\BIBLIOTECA2
collected 10 items

tests/test_biblioteca.py::test_servidor_responde PASSED [ 10%]
tests/test_biblioteca.py::test_formato_json PASSED [ 20%]
tests/test_biblioteca.py::test_obtener_libros PASSED [ 30%]
tests/test_biblioteca.py::test_agregar_libro PASSED [ 40%]
tests/test_biblioteca.py::test_actualizar_libro PASSED [ 50%]
tests/test_biblioteca.py::test_eliminar_libro PASSED [ 60%]
tests/test_biblioteca.py::test_registrar_transaccion PASSED [ 70%]
tests/test_biblioteca.py::test_obtener_transacciones PASSED [ 80%]
tests/test_biblioteca.py::test_obtener_estado_caja PASSED [ 90%]
tests/test_biblioteca.py::test_cliente_error_servidor PASSED [100%]

===== 10 passed in 0.90s =====
(venv) C:\BIBLIOTECA2>
```

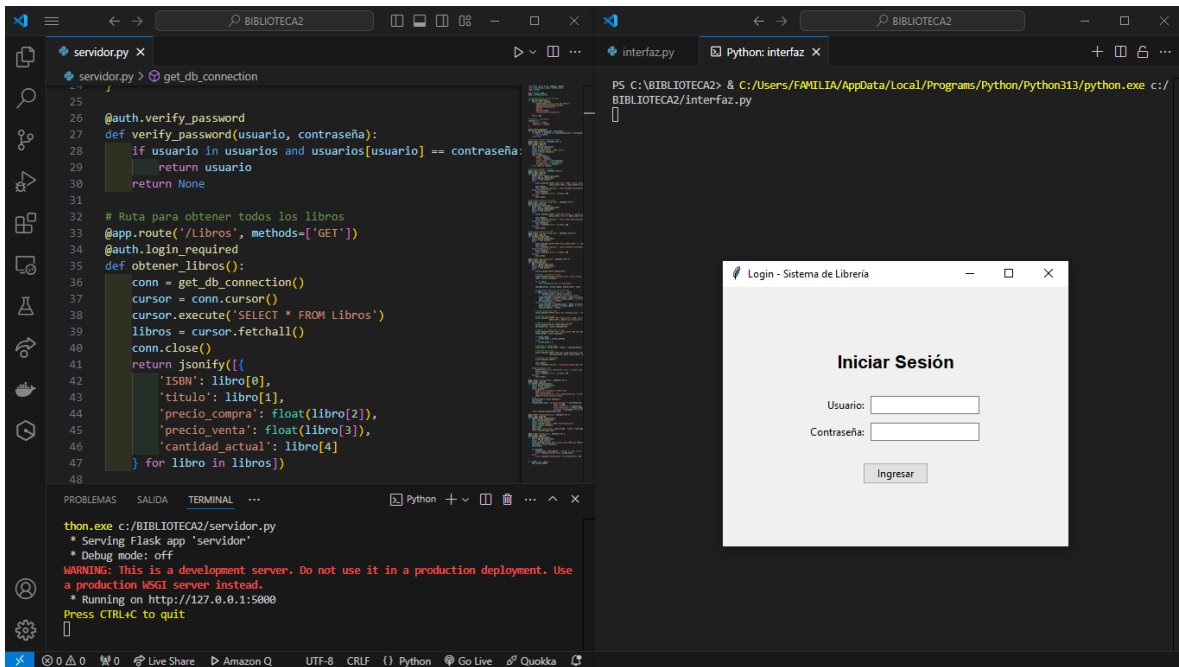
El programa o sistema como lo quieran denominar, aparte de funcionar cumplía con los 10 requerimientos funciones del proyecto.

El programa debe permitir al usuario:

- 1. Registrar un libro en el catálogo.**
- 2. Eliminar un libro del catálogo.**
- 3. Buscar un libro por título.**
- 4. Buscar un libro por ISBN.**
- 5. Abastecer ejemplares de un libro.**
- 6. Vender ejemplares de un libro.**
- 7. Calcular la cantidad de transacciones de abastecimiento de un libro particular.**
- 8. Buscar el libro más costoso.**
- 9. Buscar el libro menos costoso.**
- 10. Buscar el libro más vendido.**

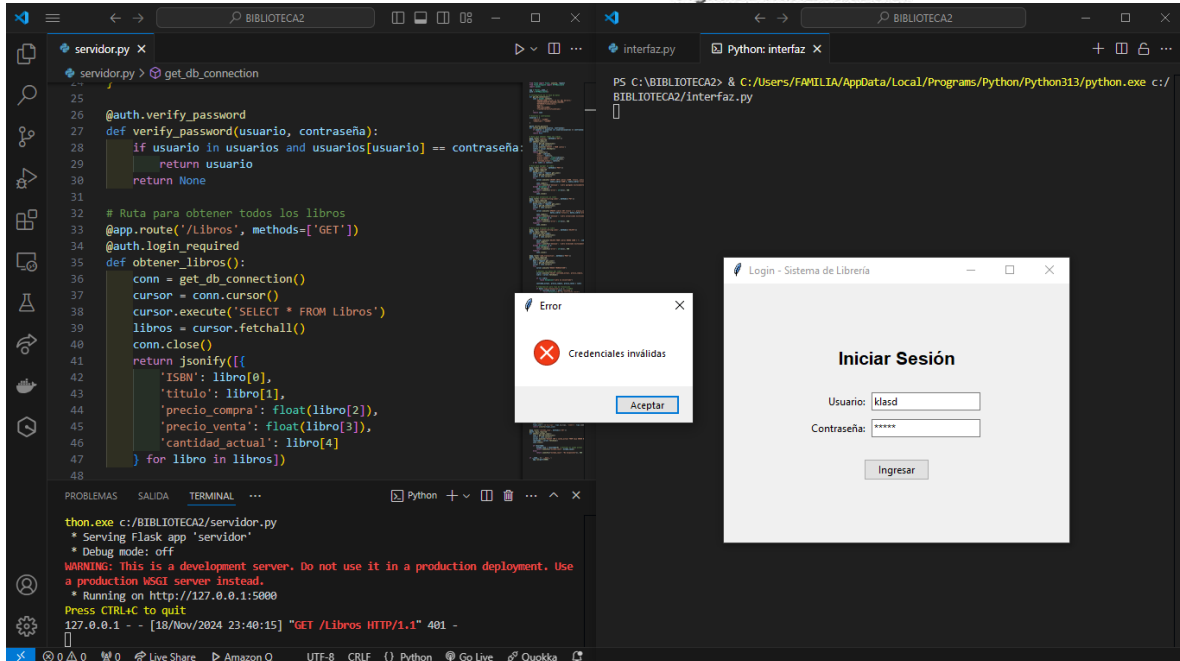
PRUEBAS MANUELES ARCHICO PYTHON

Dentro de las pruebas manuales para el servicio de Python no se pudieron hacer en el ambiente virtual con (venv) puesto que para manejar los métodos de POST o DELETE no permitía además de tener que usar los constantes CRUL junto con el autenticador. Para evitar esto y añadirme más modificaciones al código usamos la terminal de visual estudio code, como en el trabajo de manual de creación hay introducimos parte de la prueba de ejecución así que prácticamente será lo mismo la diferencia es que solo mostraremos ejecución y resultados.

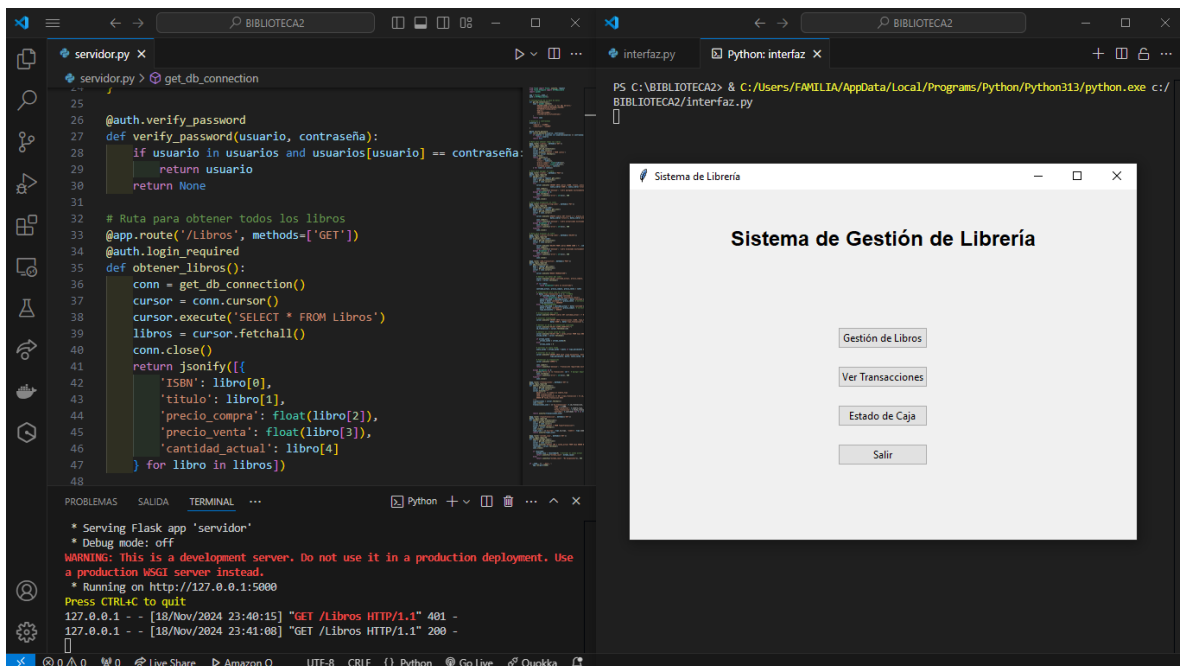


Primero ejecutamos el servidor.py para que este se conecte en la base de datos y habilite los disparadores, y ahora el cliente no se activa ya que como se dijo antes este solo hace el llamado de los disparadores, pero en este caso en vez de verlo por medio de un navegador lo vemos por una interfaz. Este tendrá las funciones de cliente, pero combinándolo con un modelo de vista controlador.

Cuando ejecutamos sale esta ventana como colocamos un autenticador sale esta interfaz para iniciar sesión si ingresamos bien las credenciales nos permite el acceso, pero si no lo hacemos

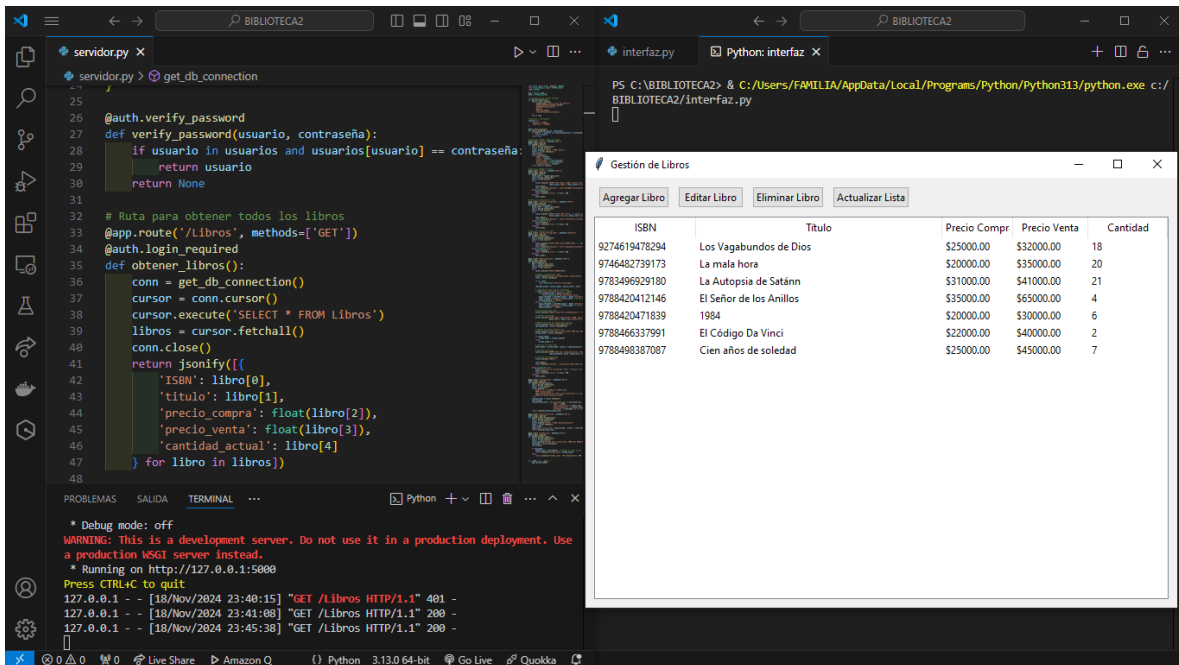


Sale un error de credenciales esto lo hacemos como para tener una mayor seguridad con la información que manejemos. Cuando colocamos bien las credenciales abre un menú con 4 opciones aun que se ve feo esto se verá mejorado más adelante, pero por motivos de este manual documentamos todo desde su proceso de idealización y su proceso de creación



Ahora si notamos bien la imagen vemos que el primer GET que está en rojo es cuando hicimos el ejemplo de las credenciales incorrectas que por cierto para ingresar utilizamos el usuario “camila” y contraseña “ca2004” o el usuario “sebastian” y contraseña “se2003” el segundo GET que está en

blanco nos indica que se inició sesión correctamente ahora navegamos por las interfaces, abrimos gestión de libros.



The screenshot shows a development environment with the following components:

- servidor.py:**

```

25
26 @auth.verify_password
27 def verify_password(usuario, contraseña):
28     if usuario in usuarios and usuarios[usuario] == contraseña:
29         return usuario
30     return None
31
32 # Ruta para obtener todos los libros
33 @app.route('/Libros', methods=['GET'])
34 @auth.login_required
35 def obtener_libros():
36     conn = get_db_connection()
37     cursor = conn.cursor()
38     cursor.execute('SELECT * FROM Libros')
39     libros = cursor.fetchall()
40     conn.close()
41     return jsonify([
42         {'ISBN': libro[0],
43          'titulo': libro[1],
44          'precio_compra': float(libro[2]),
45          'precio_venta': float(libro[3]),
46          'cantidad_actual': libro[4]}
47         for libro in libros])
48

```
- Terminal:**

```

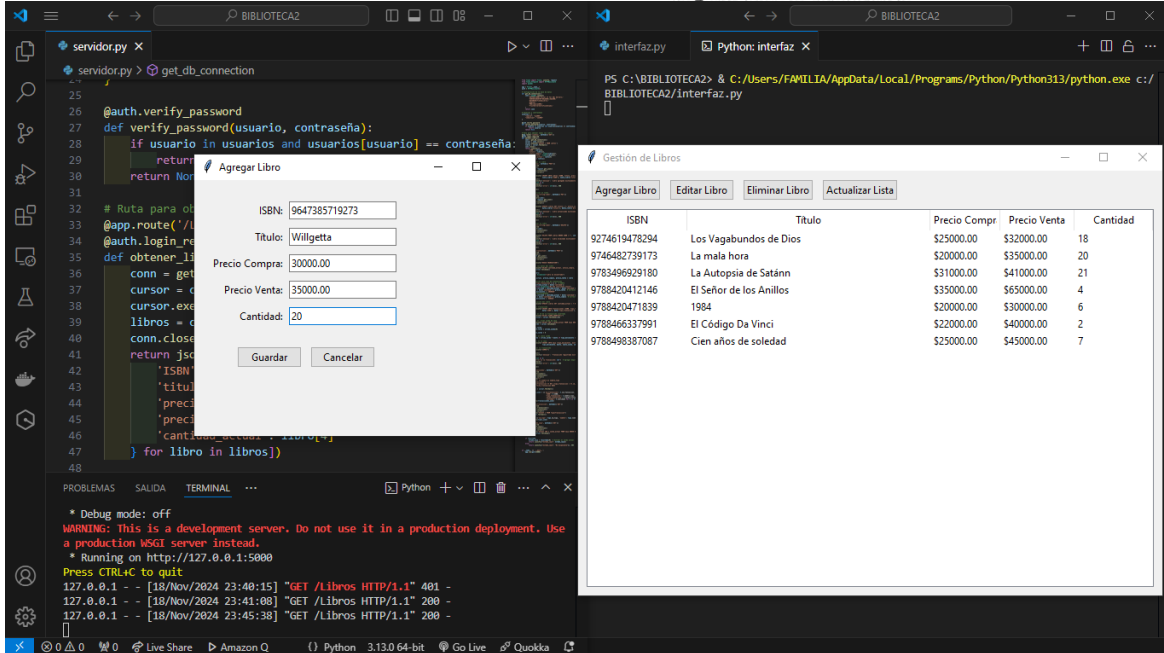
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use
a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [18/Nov/2024 23:40:15] "GET /Libros HTTP/1.1" 401 -
127.0.0.1 - - [18/Nov/2024 23:41:08] "GET /Libros HTTP/1.1" 200 -
127.0.0.1 - - [18/Nov/2024 23:45:38] "GET /Libros HTTP/1.1" 200 -

```
- Gestión de Libros:**

ISBN	Título	Precio Compr	Precio Venta	Cantidad
9274619478294	Los Vagabundos de Dios	\$25000.00	\$32000.00	18
9746482739173	La mala hora	\$20000.00	\$35000.00	20
9783496929180	La Autopsia de Satán	\$31000.00	\$41000.00	21
9788420412146	El Señor de los Anillos	\$35000.00	\$65000.00	4
9788420471839	1984	\$20000.00	\$30000.00	6
9788466337991	El Código Da Vinci	\$22000.00	\$40000.00	2
9788498387087	Cien años de soledad	\$25000.00	\$45000.00	7

Solicita la petición de ver los libros que están en la base de datos y me deje o agregar libros, editarlos, eliminarlos o que me actualice la lista de los mismos, dato curioso el editar libro fue añadido por nuestra parte ya que no es un requerimiento funcional que solicita el proyecto, pero lo colocamos como un plus del mismo.

Ahora probemos los botones, si queremos agregar un libro nos abre una ventana que nos solicita un ISBN, un título, precio de compra y venta y una cantidad de libros, actualmente contamos con 7 libros en la base de datos tanto creados como directamente creados en la SQL, creamos un libro con el "ISBN: 9647385719273" "titulo: Willgetta" "precio compra: 30.000" "precio venta: 35.000" y "cantidad: 20"

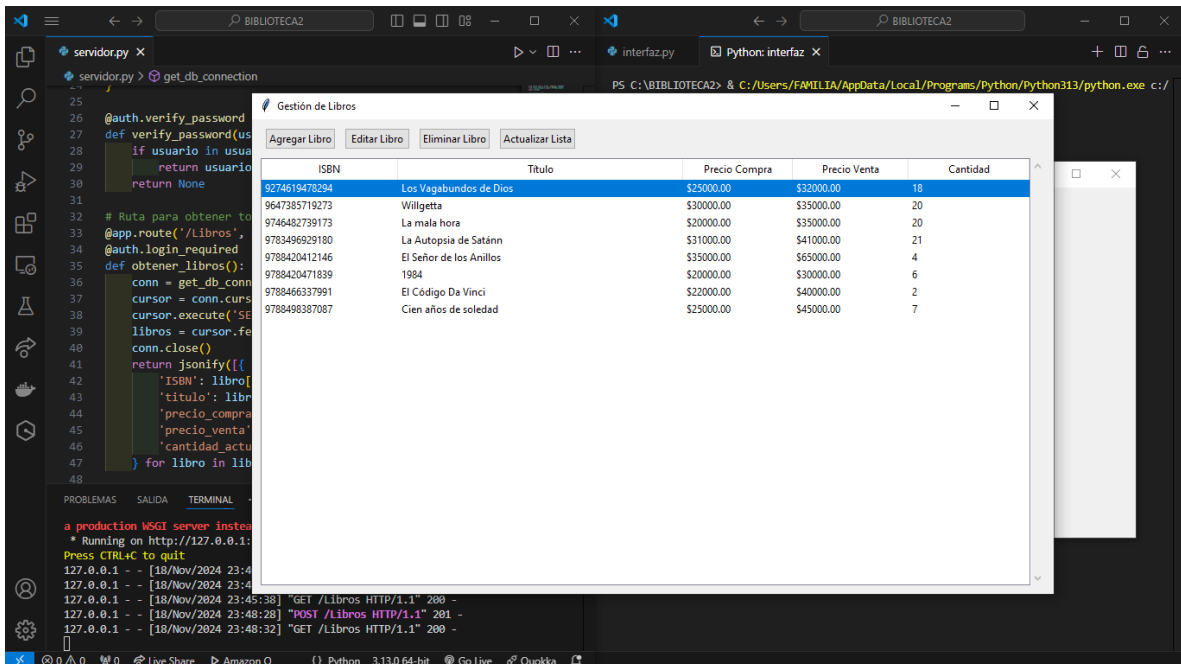


The screenshot shows a web application with a dark theme. On the left, a code editor displays the `servidor.py` file. A modal window titled "Agregar Libro" is open, showing a form with the following fields:

- ISBN: 9647385719273
- Título: Willgetta
- Precio Compra: 30000.00
- Precio Venta: 35000.00
- Cantidad: 20

Buttons for "Guardar" and "Cancelar" are at the bottom of the form. On the right, a window titled "Gestión de Libros" displays a table of books:

ISBN	Título	Precio Compra	Precio Venta	Cantidad
9274619478294	Los Vagabundos de Dios	\$25000.00	\$32000.00	18
9746482739173	La mala hora	\$20000.00	\$35000.00	20
9783496929180	La Autopsia de Satán	\$31000.00	\$41000.00	21
9788420412146	El Señor de los Anillos	\$35000.00	\$65000.00	4
9788420471839	1984	\$20000.00	\$30000.00	6
9788466337991	El Código Da Vinci	\$22000.00	\$40000.00	2
9788498387087	Cien años de soledad	\$25000.00	\$45000.00	7



This screenshot shows the same application after a book has been added. The "Gestión de Libros" window is open, and the table now includes the newly added book at the top:

ISBN	Título	Precio Compra	Precio Venta	Cantidad
9274619478294	Los Vagabundos de Dios	\$25000.00	\$32000.00	18
9647385719273	Willgetta	\$30000.00	\$35000.00	20
9746482739173	La mala hora	\$20000.00	\$35000.00	20
9783496929180	La Autopsia de Satán	\$31000.00	\$41000.00	21
9788420412146	El Señor de los Anillos	\$35000.00	\$65000.00	4
9788420471839	1984	\$20000.00	\$30000.00	6
9788466337991	El Código Da Vinci	\$22000.00	\$40000.00	2
9788498387087	Cien años de soledad	\$25000.00	\$45000.00	7

The code editor on the left shows the `servidor.py` file with a `POST` method handler for adding books. The terminal at the bottom shows the server running on `http://127.0.0.1:5000` and receiving a `POST` request.

Quando creamos el libro aparece agregado, pero como sé que mi base de datos está recibiendo la información de entrada, bueno por dos factores el primero su podemos ver en la esquina inferior izquierda tenemos una línea morada que pone un POST este método lo usamos para agregar un libro. Y bien la segunda cosa que podemos hacer es que en nuestro aplicativo donde estemos usando la base de datos SQL ejecutamos un `SELECT * FROM Libros` para validar si este fue agregado y como esto es para validar pues intentemos.

108 %

Results Messages

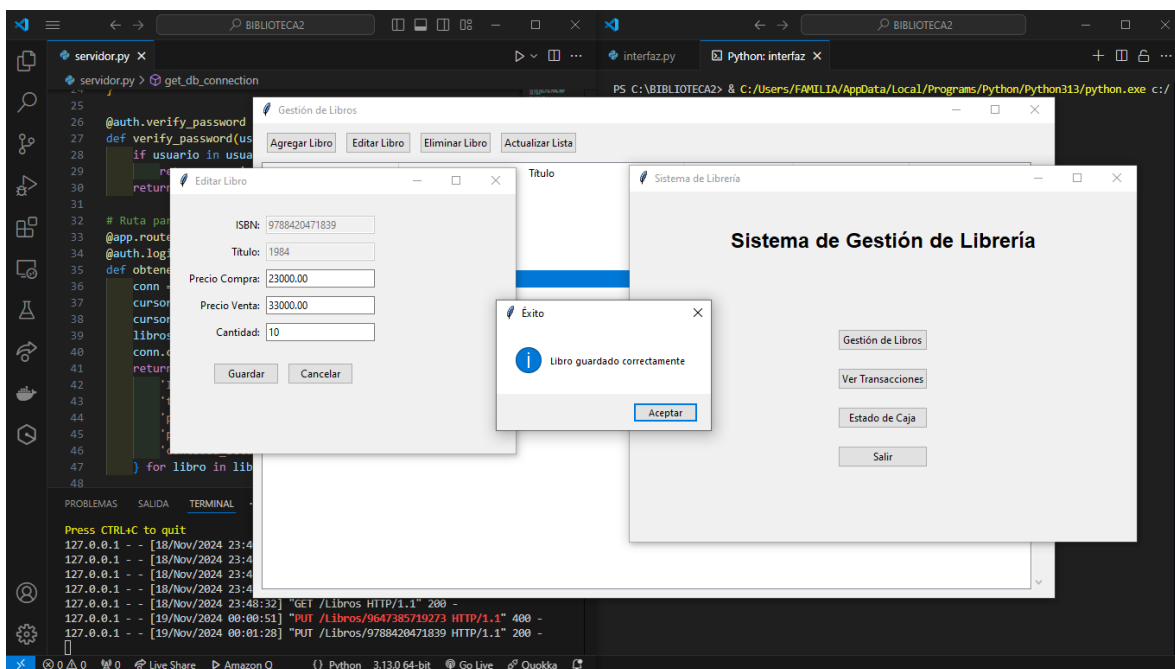
	ISBN	titulo	precio_compra	precio_venta	cantidad_actual
1	9274619478294	Los Vagabundos de Dios	25000.00	32000.00	18
2	9647385719273	Willgetta	30000.00	35000.00	20
3	9746482739173	La mala hora	20000.00	35000.00	20
4	9783496929180	La Autopsia de Satán	31000.00	41000.00	21
5	9788420412146	El Señor de los Anillos	35000.00	65000.00	4
6	9788420471839	1984	20000.00	30000.00	6
7	9788466337991	El Código Da Vinci	22000.00	40000.00	2
8	9788498387087	Cien años de soledad	25000.00	45000.00	7

Query executed successfully.

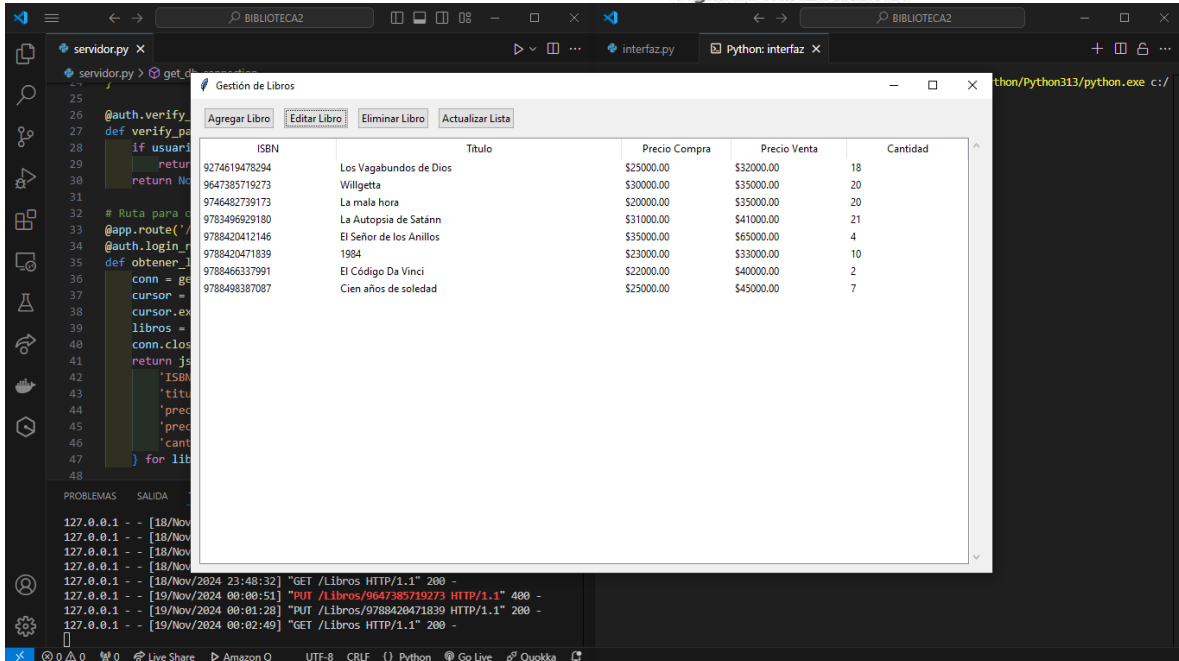
DESKTOP-0E1UK80\SQLNEW (16... sa (71) TiendaLibros 00:00:00 8 rows

Ln 148 Col 1 Ch 1 INS

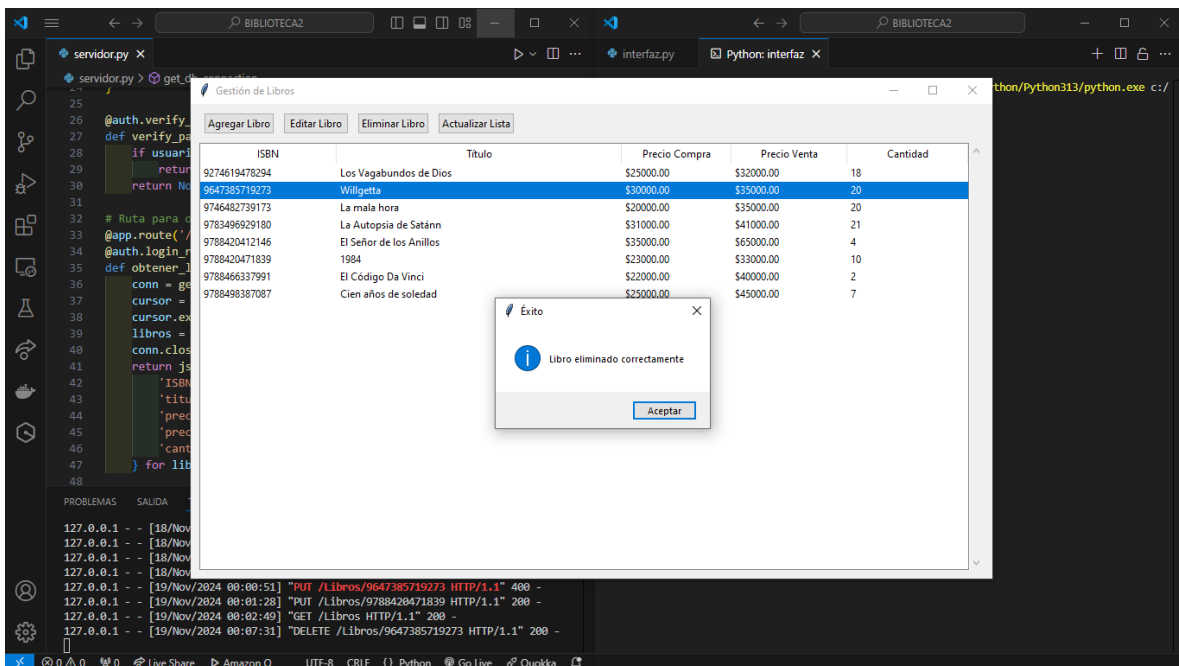
Y como podemos ver acá está el libro que acabamos de agregar, así mismo pasa cuando queremos eliminar para eliminar tenemos que seleccionar el libro si no seleccionamos un libro nos sale una alerta indicando que tenemos que escoger un libro para eliminar y la misma función tiene para editar, así que miremos cómo funciona el editar y eliminar.

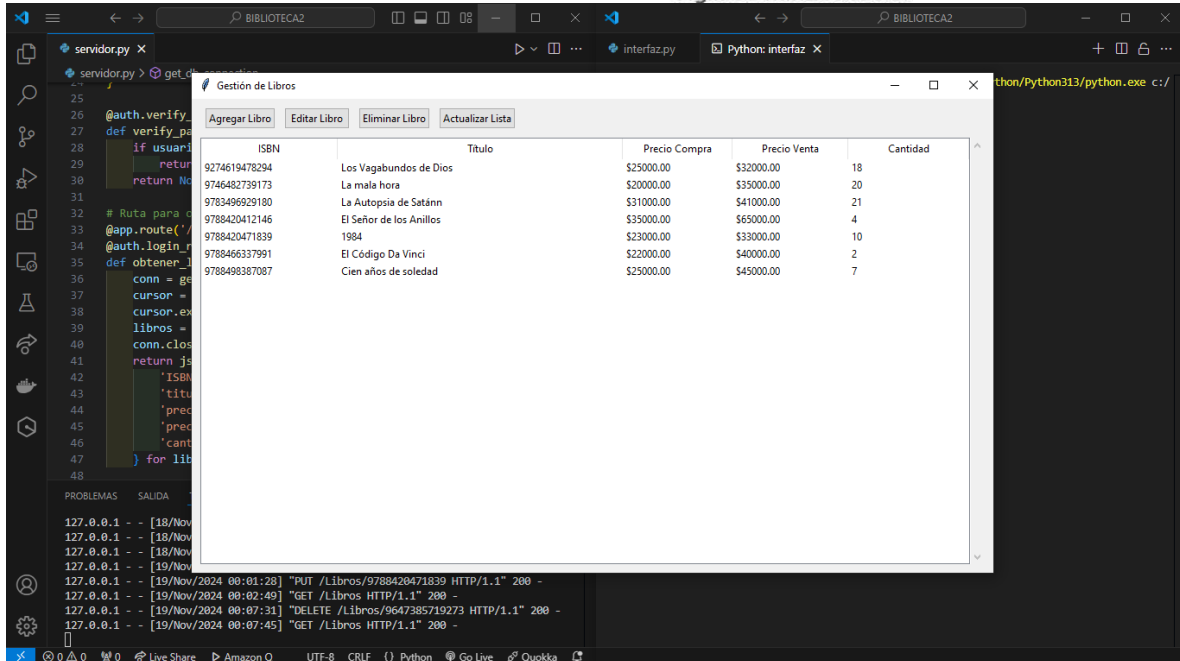


En este caso editamos el libro de 1994 le bajamos los precios y aumentamos las cantidades nos transmite un cambio y utiliza el servidor un método POST para esto. Y como comprobar este libro tenía un precio de compra de 20.000 un precio de venta de 30.000 y tenía 6 unidades ahora al editarlo le decimos que su precio de compra es de 23.000, su precio de venta es de 33.000 y tenemos al momento 10 unidades.

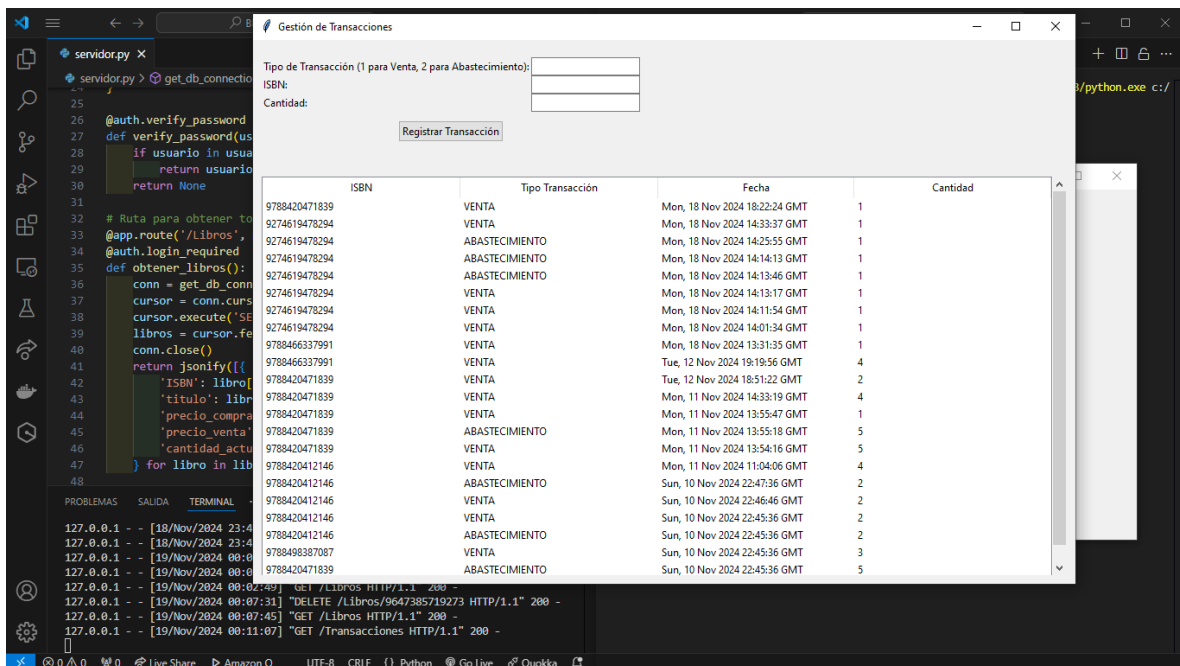


Acá tenemos el cambio y como validamos que si funciona lo pues lo mismo validamos por la base de datos. Con esto uno tiene un control de una base de datos con un cliente servidor combinándolo con un modelo de vista controlador. Ahora probemos el ultimo y es eliminar un libro. Eliminemos el libro que creamos seleccionamos el libro y apretamos el botón de borrar y el libro hace un DELETE





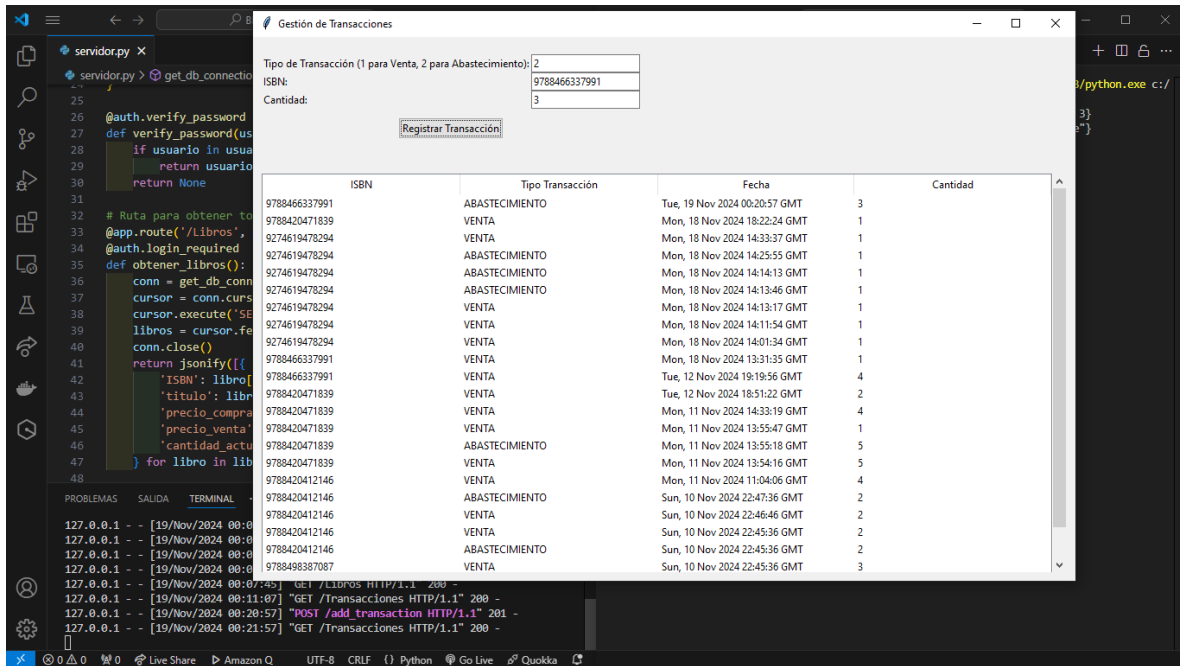
Y como vemos en la esquina inferior izquierda vemos el método DELETE con la ruta DELETE/Libros/#ISBN del libro eliminado en este caso vemos la eliminación de Willgetta. Ahora veamos un poco sobre las interfaces de transacciones. Este tipo de interfaz es un poco sensible puesto que esta trabaja con TREAAGERS que están implementados en la SQL.



Cuando le hacemos la petición a la base de datos y servidor ya no lo reconoce como /Libro, si no como /Transacciones como podemos ver nos permite el ingreso y ver las transacciones que hemos realizado, ya sea por medio de venta o abastecimiento, la fecha en la que se hizo la transacción y la cantidad que se vendió o abasteció de un libro ahora hagamos una prueba muy minúscula cabe

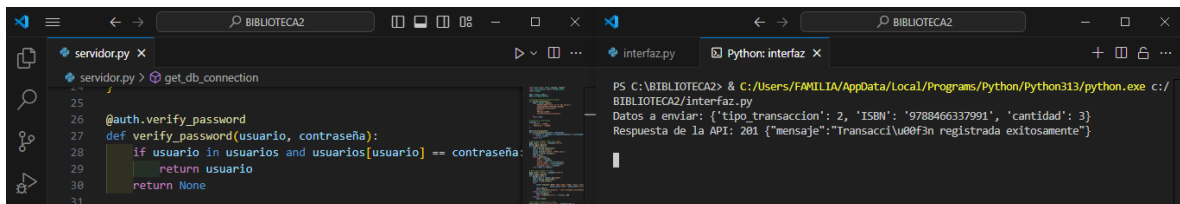
aclarar un error que persiste en esta parte cuando queremos hacer una venta o un abastecimiento no toma los valores indicados es decir tengo un libro con 20 unidades vendiendo una y me deben de quedar 19 unidades pues no, aparece que quedan 18 unidades resta una unidad mas lo mismo pasa en abastecimiento den un libro de 5 unidades recargo 1 unidad y recarga 2, aun no sabemos por que ya miramos y a pesar de tener ayuda de las IA aun no podemos encontrar el error pero de igual manera miremos como funciona.

Vamos a abastecer el libro del **El Código Da Vinci** con el ISBN **9788466337991** conocemos que tiene 2 unidades gracias a la gestión de libro y base de datos verdad, pero vamos a aumentar la cantidad de ejemplares a 3 para el ejemplo



ISBN	Tipo Transacción	Fecha	Cantidad
9788466337991	ABASTECIMIENTO	Tue, 19 Nov 2024 00:20:57 GMT	3
9788420471839	VENTA	Mon, 18 Nov 2024 18:22:24 GMT	1
9274619478294	VENTA	Mon, 18 Nov 2024 14:33:37 GMT	1
9274619478294	ABASTECIMIENTO	Mon, 18 Nov 2024 14:25:55 GMT	1
9274619478294	ABASTECIMIENTO	Mon, 18 Nov 2024 14:14:13 GMT	1
9274619478294	ABASTECIMIENTO	Mon, 18 Nov 2024 14:13:46 GMT	1
9274619478294	VENTA	Mon, 18 Nov 2024 14:13:17 GMT	1
9274619478294	VENTA	Mon, 18 Nov 2024 14:11:54 GMT	1
9274619478294	VENTA	Mon, 18 Nov 2024 14:01:34 GMT	1
9788466337991	VENTA	Mon, 18 Nov 2024 13:31:35 GMT	1
9788466337991	VENTA	Tue, 12 Nov 2024 19:19:56 GMT	4
9788420471839	VENTA	Tue, 12 Nov 2024 18:51:22 GMT	2
9788420471839	VENTA	Mon, 11 Nov 2024 14:33:19 GMT	4
9788420471839	VENTA	Mon, 11 Nov 2024 13:55:47 GMT	1
9788420471839	ABASTECIMIENTO	Mon, 11 Nov 2024 13:55:18 GMT	5
9788420471839	VENTA	Mon, 11 Nov 2024 13:54:16 GMT	5
9788420412146	VENTA	Mon, 11 Nov 2024 11:04:06 GMT	4
9788420412146	ABASTECIMIENTO	Sun, 10 Nov 2024 22:47:36 GMT	2
9788420412146	VENTA	Sun, 10 Nov 2024 22:46:46 GMT	2
9788420412146	VENTA	Sun, 10 Nov 2024 22:45:36 GMT	2
9788420412146	ABASTECIMIENTO	Sun, 10 Nov 2024 22:45:36 GMT	2
9788498387087	VENTA	Sun, 10 Nov 2024 22:45:36 GMT	3

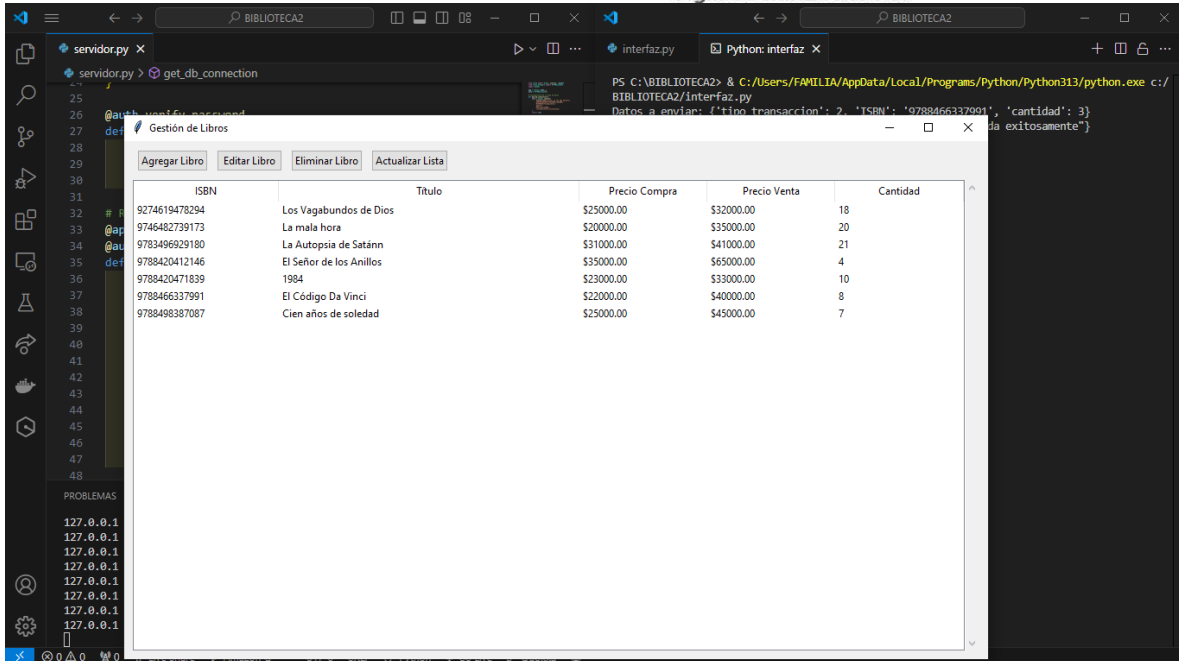
Ahora bien, una forma de validar sin necesidad de ver la SQL dentro de nuestro panel donde ejecutamos la interfaz esta nos indica que se hizo un cambio en la API realizando un registro y mandando un hermoso mensaje que indica que la transacción fue registrada correctamente.



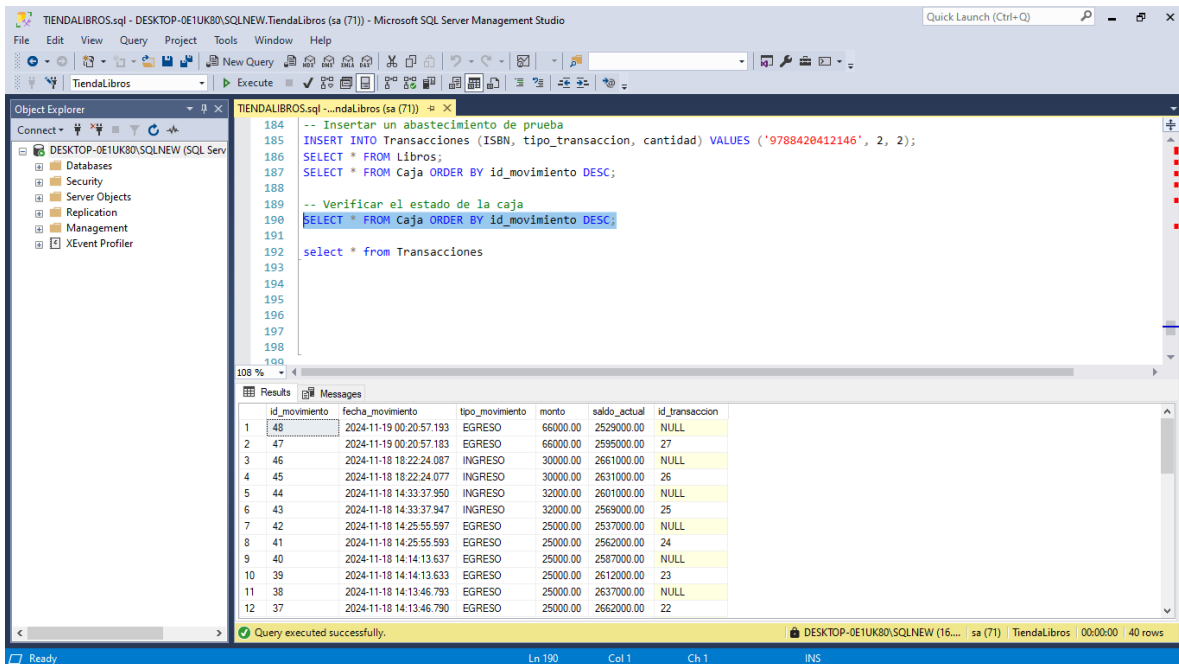
```

PS C:\BIBLIOTECA2> & C:/Users/FAMILIA/AppData/Local/Programs/Python/Python313/python.exe c:/BIBLIOTECA2/interfaz.py
Datos a enviar: {'tipo_transaccion': 2, 'ISBN': '9788466337991', 'cantidad': 3}
Respuesta de la API: 201 {'mensaje': 'Transacción registrada exitosamente'}
  
```

Como vemos realiza el abastecimiento de los libros, pero ahora debería de aparecer que tengo 5 libros de esta ejemplar verdad. Pero cuando consultamos a la base de datos o a la gestión de libros aparece que tenemos 8 ejemplares en vez de 5 aun no sabemos por que esto sucede pero seguimos trabajando en encontrar el error o la con función porque es como si confundiera los signos de las operaciones.

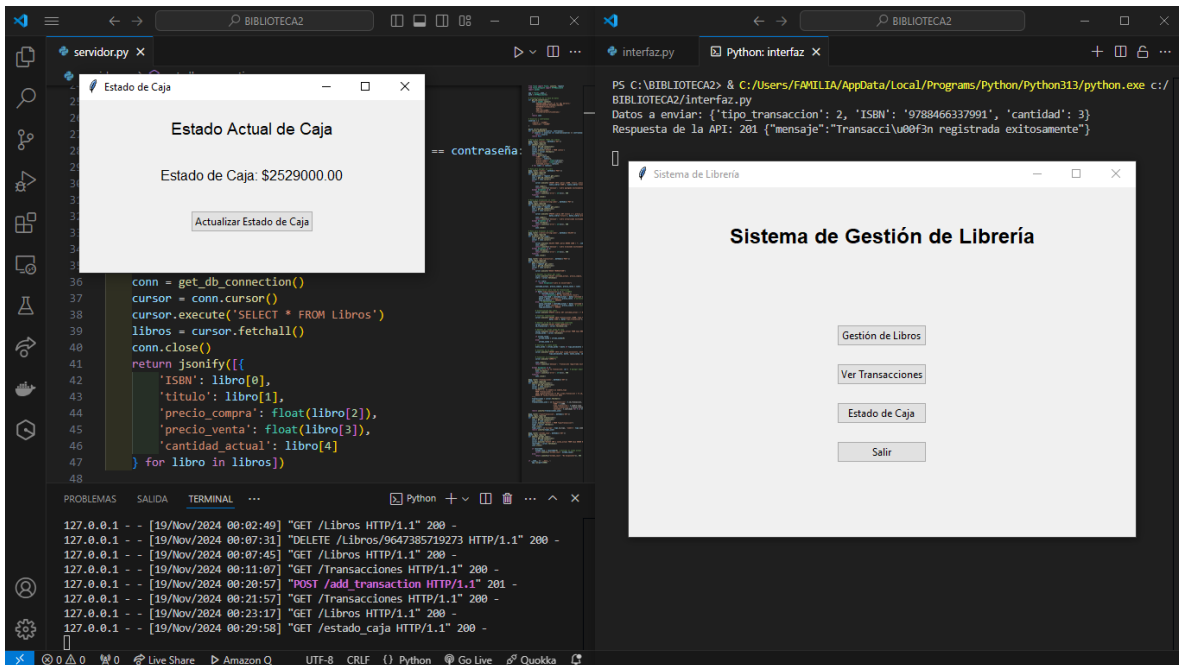


Ahora bien, ya por último es ver el estado de caja como su nombre lo indica, nos deja ver cuanto tenemos en caja. Entonces miremos en la base de datos y después una prueba para validar que tenemos.



Dentro de nuestra base de datos tenemos que tener 2'529.000 de tanto que hemos hecho transacciones de venta y abastecimiento. Y nos debe de aparecer esta misma cantidad en el estado de caja, nuestro estado de caja no muestra como en la base de datos ya que tomamos la decisión de que solo se vea el dinero de caja por 2 razones una por estética y la otra es que se confundía el sistema cuando intentamos agregar nuevos campos de valores ya que como en la base de datos la

tabla de caja no esta como tal relacionada a algo fuerte si no que se maneja por TRIGGERS pues e confundía en ocasiones. Y dándonos resultados como estos.



Como podemos ver el cliente servidor mantienen una comunicación continua además de que sus interfaces funcionan para realizar este tipo de interpretaciones.

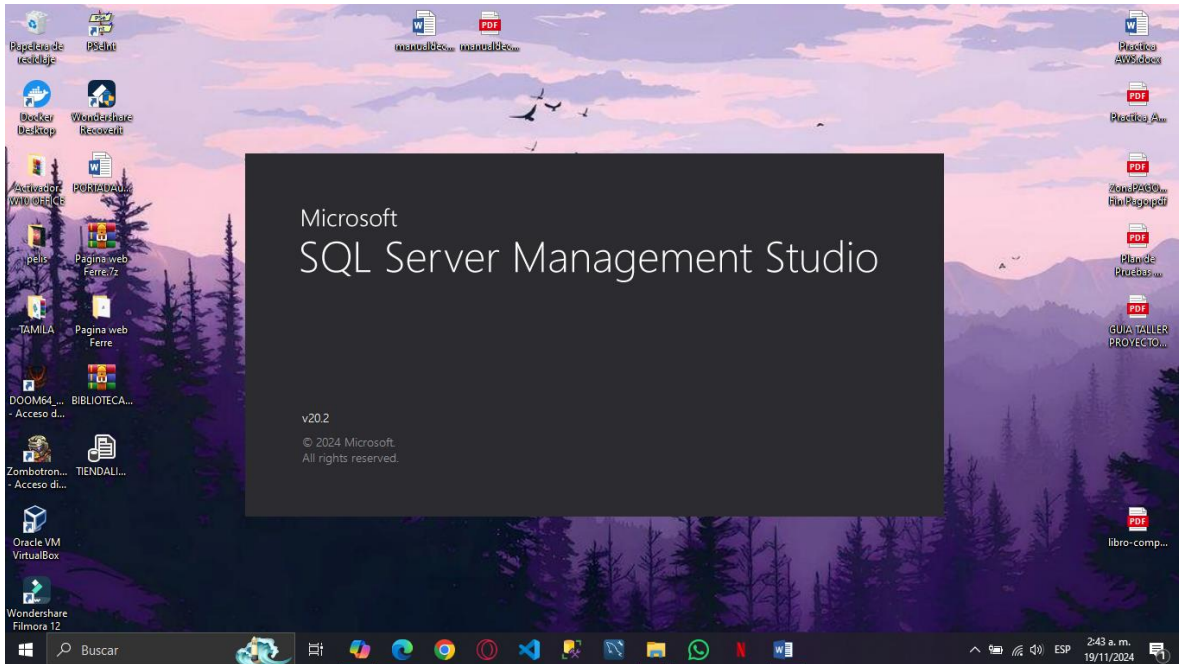
PRUEBAS MANUALES ARCHIVO SQL

Para la prueba de la SQL vamos a hacerla de manera manual como un clásico así mismo para comprobar que la base de datos está configurada desde un principio para cumplir los requerimientos funciones que solicita el proyecto.

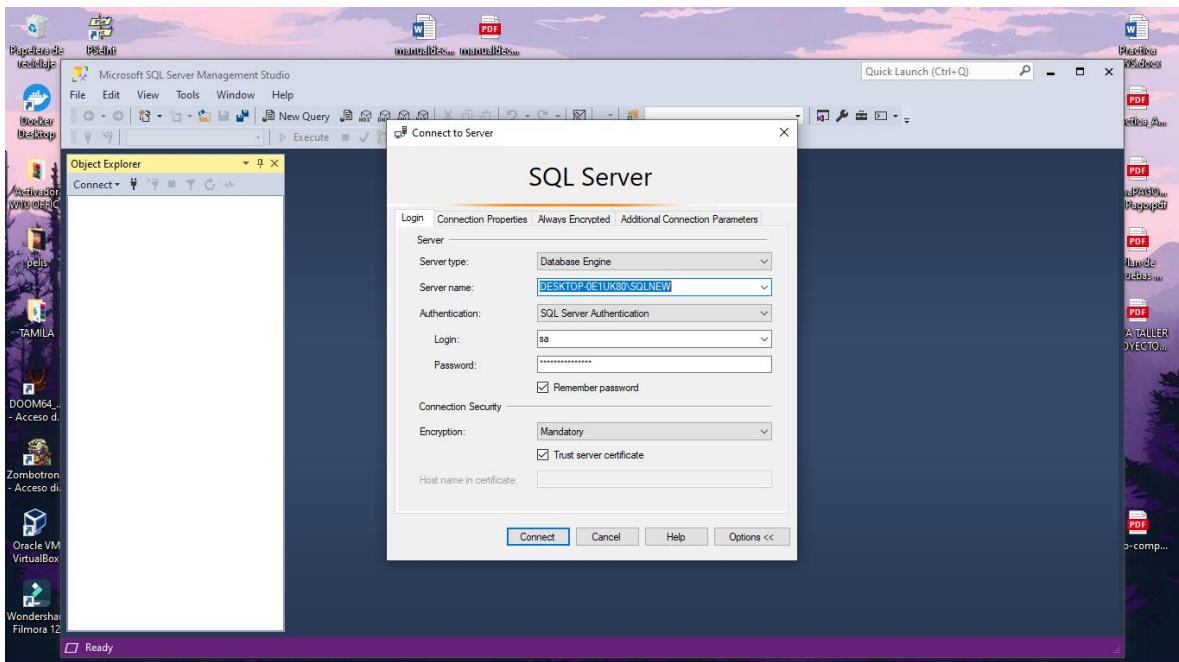
El programa debe permitir al usuario:

- 1. Registrar un libro en el catálogo.**
- 2. Eliminar un libro del catálogo.**
- 3. Buscar un libro por título.**
- 4. Buscar un libro por ISBN.**
- 5. Abastecer ejemplares de un libro.**
- 6. Vender ejemplares de un libro.**
- 7. Calcular la cantidad de transacciones de abastecimiento de un libro particular.**
- 8. Buscar el libro más costoso.**
- 9. Buscar el libro menos costoso.**
- 10. Buscar el libro más vendido.**

Para iniciar tenemos que recordar que este tipo de proyectos es más fácil manejarlos por medio de un usuario y contraseña esto con la finalidad que tenga un enfoque más específico así mismo para que este obtenga permisos como de editar que sea público, etc. Además, usamos el servicio de Microsoft SQL



Ingresamos por medio de un usuario y contraseña y para tener una mayor privacidad con el proyecto usamos una nueva extensión llamada SQLNEW



A continuación te mostramos el código fuente de la base de datos y te explicamos como funciona.

CODIGO FUENTE BASE DE DATOS: TIENDALIBROS.SQL

```
-- Crear base de datos y usarla
CREATE DATABASE TiendaLibros;
USE TiendaLibros;

-- Tabla para libros
CREATE TABLE Libros (
    ISBN VARCHAR(13) PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    precio_compra DECIMAL(10, 2) NOT NULL,
    precio_venta DECIMAL(10, 2) NOT NULL,
    cantidad_actual INT NOT NULL DEFAULT 0,
    CONSTRAINT chk_precios CHECK (precio_venta >= precio_compra),
    CONSTRAINT chk_cantidad CHECK (cantidad_actual >= 0)
);

-- Tabla para tipos de transacciones
CREATE TABLE TiposTransaccion (
    id_tipo INT PRIMARY KEY,
    nombre VARCHAR(20) NOT NULL,
    CONSTRAINT chk_tipo CHECK (nombre IN ('VENTA', 'ABASTECIMIENTO'))
);

-- Insertar tipos de transacciones válidos
INSERT INTO TiposTransaccion (id_tipo, nombre) VALUES
(1, 'VENTA'),
(2, 'ABASTECIMIENTO');

-- Tabla para transacciones
CREATE TABLE Transacciones (
    id_transaccion INT IDENTITY(1,1) PRIMARY KEY,
    ISBN VARCHAR(13) NOT NULL,
    tipo_transaccion INT NOT NULL,
    fecha_transaccion DATETIME NOT NULL DEFAULT GETDATE(),
    cantidad INT NOT NULL,
    FOREIGN KEY (ISBN) REFERENCES Libros(ISBN) ON DELETE CASCADE,
    FOREIGN KEY (tipo_transaccion) REFERENCES TiposTransaccion(id_tipo),
    CONSTRAINT chk_cantidad_transaccion CHECK (cantidad > 0)
);

-- Tabla para caja
CREATE TABLE Caja (
    id_movimiento INT IDENTITY(1,1) PRIMARY KEY,
    fecha_movimiento DATETIME NOT NULL DEFAULT GETDATE(),
    tipo_movimiento VARCHAR(20) NOT NULL,
    monto DECIMAL(10, 2) NOT NULL,
    saldo_actual DECIMAL(10, 2) NOT NULL,
    id_transaccion INT,
    FOREIGN KEY (id_transaccion) REFERENCES Transacciones(id_transaccion),
    CONSTRAINT chk_tipo_movimiento CHECK (tipo_movimiento IN ('INGRESO',
'EGRESO'))
);

-- Insertar el saldo inicial en caja
INSERT INTO Caja (tipo_movimiento, monto, saldo_actual)
VALUES ('INGRESO', 1000000.00, 1000000.00);
```

```
-- Trigger para ventas: actualiza el inventario y suma a la caja
CREATE TRIGGER trg_Venta_Transaccion
ON Transacciones
AFTER INSERT
AS
BEGIN
    DECLARE @ISBN VARCHAR(13), @tipo_transaccion INT, @cantidad INT, @precio_venta
    DECIMAL(10, 2);

    SELECT @ISBN = ISBN, @tipo_transaccion = tipo_transaccion, @cantidad =
    cantidad FROM inserted;

    IF @tipo_transaccion = 1
    BEGIN
        SELECT @precio_venta = precio_venta FROM Libros WHERE ISBN = @ISBN;

        -- Actualizar inventario
        UPDATE Libros
        SET cantidad_actual = cantidad_actual - @cantidad
        WHERE ISBN = @ISBN;

        -- Registrar ingreso en caja
        INSERT INTO Caja (tipo_movimiento, monto, saldo_actual, id_transaccion)
        SELECT 'INGRESO', @cantidad * @precio_venta,
        (SELECT TOP 1 saldo_actual FROM Caja ORDER BY id_movimiento DESC) +
        (@cantidad * @precio_venta),
        inserted.id_transaccion
        FROM inserted;
    END
END;

-- Trigger para abastecimientos: actualiza inventario y resta de la caja
CREATE TRIGGER trg_Abastecimiento_Transaccion
ON Transacciones
AFTER INSERT
AS
BEGIN
    DECLARE @ISBN VARCHAR(13), @tipo_transaccion INT, @cantidad INT,
    @precio_compra DECIMAL(10, 2);

    SELECT @ISBN = ISBN, @tipo_transaccion = tipo_transaccion, @cantidad =
    cantidad FROM inserted;

    IF @tipo_transaccion = 2
    BEGIN
        SELECT @precio_compra = precio_compra FROM Libros WHERE ISBN = @ISBN;

        -- Actualizar inventario
        UPDATE Libros
        SET cantidad_actual = cantidad_actual + @cantidad
        WHERE ISBN = @ISBN;

        -- Registrar egreso en caja
        INSERT INTO Caja (tipo_movimiento, monto, saldo_actual, id_transaccion)
        SELECT 'EGRESO', @cantidad * @precio_compra,
        (SELECT TOP 1 saldo_actual FROM Caja ORDER BY id_movimiento DESC) -
        (@cantidad * @precio_compra),
```

```

        inserted.id_transaccion
    FROM inserted;
END
END;

-- DATOS DE PRUEBA --

--1. PRUEBAS DE REGISTROS (LIBROS Y TRANSACCIONES)
-- Insertar libros de prueba
INSERT INTO Libros (ISBN, titulo, precio_compra, precio_venta, cantidad_actual)
VALUES
('9788498387087', 'Cien años de soledad', 25000.00, 45000.00, 10),
('9788420471839', '1984', 20000.00, 35000.00, 15),
('9788420412146', 'El Señor de los Anillos', 35000.00, 65000.00, 8),
('9788466337991', 'El Codigo Da Vinci', 22000.00, 40000.00, 12);
-- Insertar transacciones de prueba
INSERT INTO Transacciones (ISBN, tipo_transaccion, cantidad) VALUES
('9788498387087', 1, 3),
('9788420471839', 2, 5);

--2. PRUEBA DE ELIMINAR LIBRO DEL CATALOGO
--eliminar libro
DELETE FROM Libros WHERE ISBN = '9788498387087';
SELECT * FROM Libros WHERE ISBN = '9788498387087';

--3. PRUEBA DE BUSCAR LIBRO POR TITULO
-- Búsqueda por título
SELECT * FROM Libros WHERE titulo = '1984';

--4. PRUEBA DE BUSCAR LIBRO POR ISBN
-- Búsqueda por ISBN
SELECT * FROM Libros WHERE ISBN = '9788420471839';

--5. PRUEBA DE ABASTECIMIENTO
--abastecimiento de el señor de los anillos
INSERT INTO Transacciones (ISBN, tipo_transaccion, cantidad) VALUES
('9788420412146', 2, 2);
SELECT * FROM Libros;
SELECT * FROM Caja ORDER BY id_movimiento DESC;

--6 PRUEBA VENTA DE LIBRO
INSERT INTO Transacciones (ISBN, tipo_transaccion, cantidad) VALUES
('9788420412146', 1, 2);
SELECT * FROM Libros;
SELECT * FROM Caja ORDER BY id_movimiento DESC;

--7 Calcular la cantidad de transacciones de abastecimiento de un libro
DECLARE @ISBN_buscar VARCHAR(20) = '9788420412146';
SELECT COUNT(*) AS cantidad_abastecimientos
FROM Transacciones
WHERE ISBN = @ISBN_buscar AND tipo_transaccion = 2;

--8 Buscar libro mas costoso

SELECT TOP 1 ISBN, titulo, precio_venta
FROM Libros

```

```

ORDER BY precio_venta DESC;

--9 buscar libro menos costoso

SELECT TOP 1 ISBN, titulo, precio_venta
FROM Libros
ORDER BY precio_venta ASC;

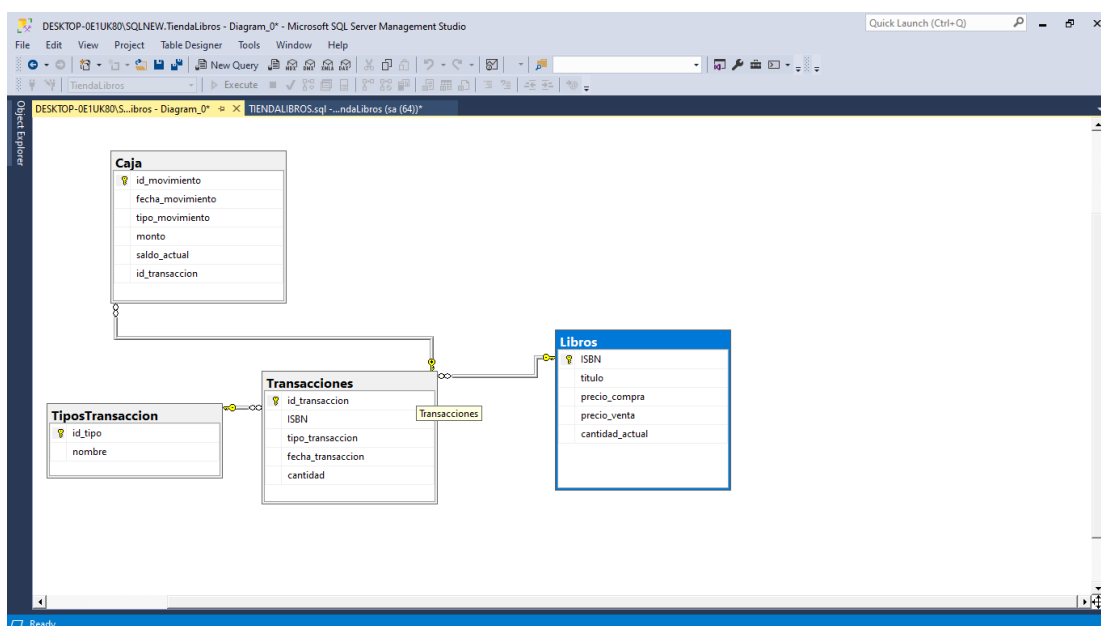
--10 buscar libro mas vendido

SELECT TOP 1 l.ISBN, l.titulo, SUM(t.cantidad) AS total_vendidos
FROM Transacciones t
JOIN Libros l ON t.ISBN = l.ISBN
WHERE t.tipo_transaccion = 1
GROUP BY l.ISBN, l.titulo
ORDER BY total_vendidos DESC;
  
```

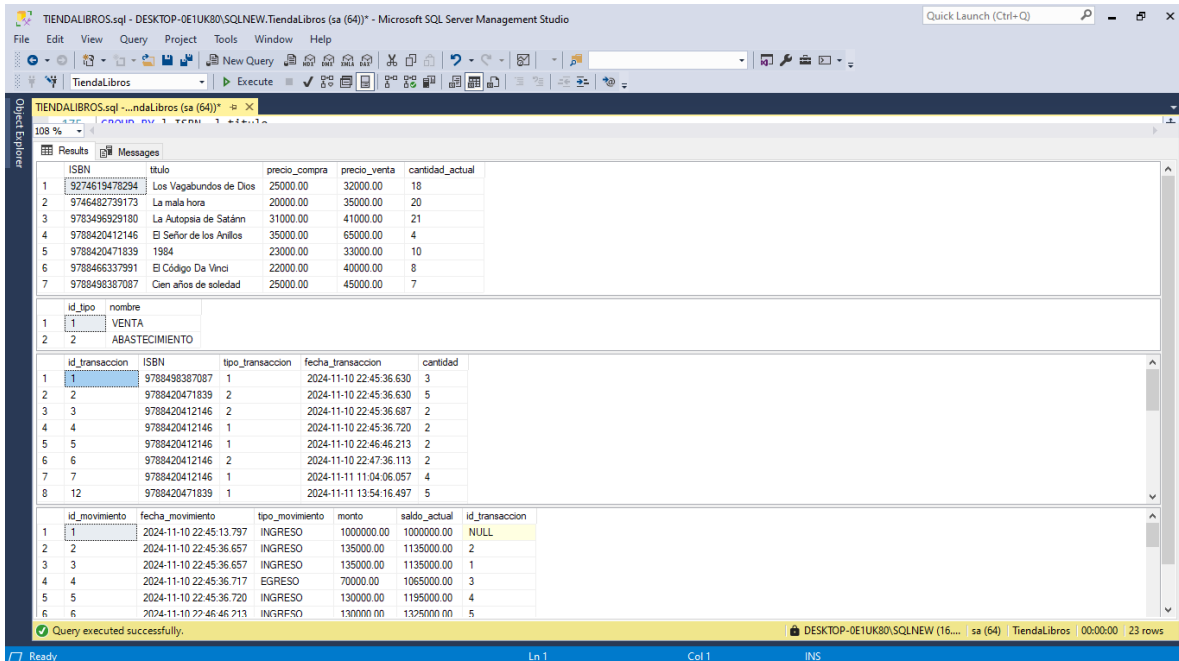
El código funciona de la siguiente manera primero la entrada de crear base de datos lo hacemos para crear un entorno en donde podamos trabajar verdad, posteriormente creamos campos de almacenamiento o también llamadas tablas en donde se almacenarán datos dentro de estas mismas tenemos las siguientes tablas:

- Tabla de libros
- Tabla de tipos de transacción
- Tabla de transacciones
- Tabla de caja

Dentro de esta base de datos tendremos disparadores, si claro que los tenemos disparadores puesto que tenemos registros actualizados de los libros cantidades, valores verdad. Aparte de esto cada tabla tiene su llave primaria y llave foránea y cada TRIGGER tiene su estructura para realizar el llamado esto con el fin de que la base de datos este bien estructurada. Ahora bien, cada base de datos tiene su modelado UML para mirar la distribución ¿verdad? El nuestro es así



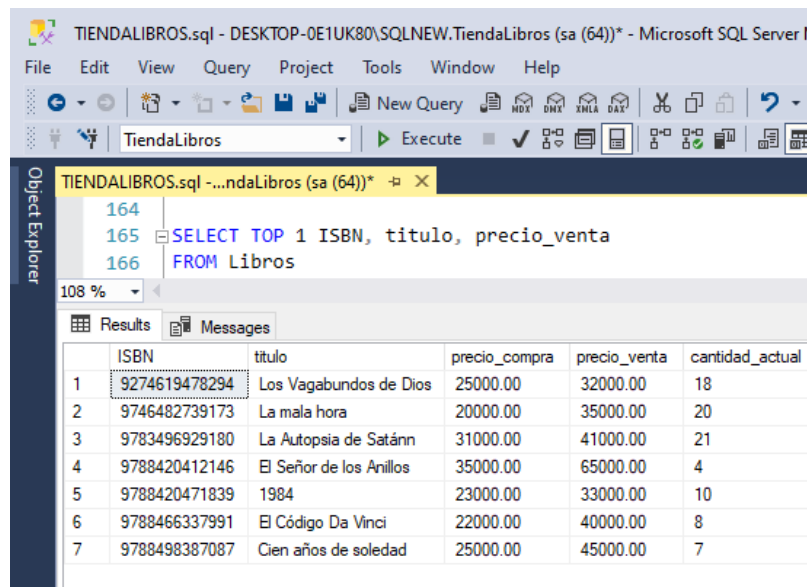
Se ve sencillo y lo es, pero para este proyecto de biblioteca se realizó lo que se solicitó en el enunciado así mismo para cumplir el funcionamiento de todos los requerimientos. Como prueba inicial hacemos un llamado de todas las tablas para validar el funcionamiento de llaves foráneas con llaves primarias



ISBN	titulo	precio_compra	precio_venta	cantidad_actual
9274619478294	Los Vagabundos de Dios	25000.00	32000.00	18
9746482739173	La mala hora	20000.00	35000.00	20
9783496929180	La Autopsia de Satán	31000.00	41000.00	21
9788420412146	El Señor de los Anillos	35000.00	65000.00	4
9788420471839	1984	23000.00	33000.00	10
9788466337991	El Código Da Vinci	22000.00	40000.00	8
9788498387087	Cien años de soledad	25000.00	45000.00	7

Esto lo hacemos antes de generar consultas específicas y tener una idea de que tiene cada tabla para si mismo crear las consultas que nos permita cumplir los requerimientos funcionales.

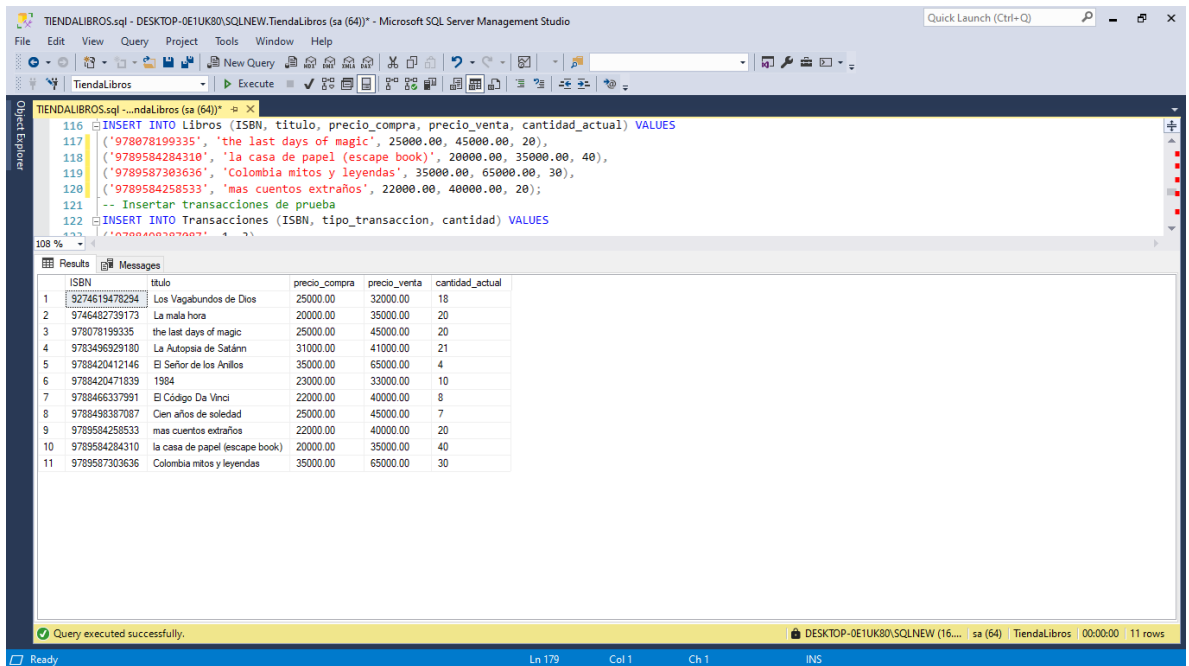
1. Registrar un libro en catalogo



ISBN	titulo	precio_compra	precio_venta	cantidad_actual
9274619478294	Los Vagabundos de Dios	25000.00	32000.00	18
9746482739173	La mala hora	20000.00	35000.00	20
9783496929180	La Autopsia de Satán	31000.00	41000.00	21
9788420412146	El Señor de los Anillos	35000.00	65000.00	4
9788420471839	1984	23000.00	33000.00	10
9788466337991	El Código Da Vinci	22000.00	40000.00	8
9788498387087	Cien años de soledad	25000.00	45000.00	7

Estos son los libros actuales dentro de la base de datos ahora agregaremos los siguientes libros para que nuestro catalogo sea mucho más amplio.

```
--1. PRUEBAS DE REGISTROS (LIBROS Y TRANSACCIONES)
-- Insertar libros de prueba
INSERT INTO Libros (ISBN, titulo, precio_compra, precio_venta,
cantidad_actual) VALUES
('978078199335', 'the last days of magic', 25000.00, 45000.00, 20),
('9789584284310', 'la casa de papel (escape book)', 20000.00, 35000.00,
40),
('9789587303636', 'Colombia mitos y leyendas', 35000.00, 65000.00, 30),
('9789584258533', 'mas cuentos extraños', 22000.00, 40000.00, 20);
```



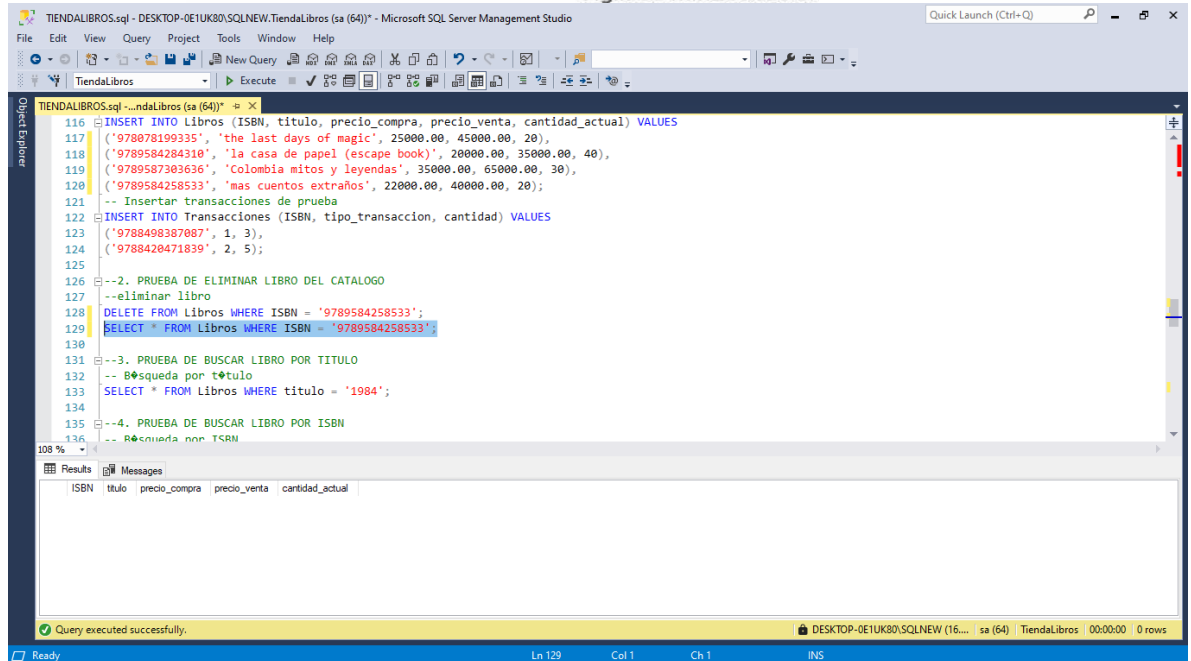
Realizamos el ingreso de los libros nuevos y aumenta nuestro catálogo de libros lo cual es un buen paso, siendo el 1 de 10

2. Eliminar un libro del catálogo.

- Utilizando una sentencia tan fácil como lo es “DELETE” se puede generar una eliminación de un libro dentro de nuestro catálogo. Para esta función usamos la siguiente función.

```
--2. PRUEBA DE ELIMINAR LIBRO DEL CATALOGO
--eliminar libro
DELETE FROM Libros WHERE ISBN = '9789584258533';
SELECT * FROM Libros WHERE ISBN = '9789584258533';
```

Esto nos permite realizar una eliminación en este caso elimina el libro más cuentos extraños, y al ejecutarlos tenemos lo siguiente



Hacemos el llamado del libro, pero como se elimino es un libro que no existe en la biblioteca así cumpliendo el 2 requerimiento por el momento van 2 de 10

3. PRUEBA 3 Y 4 buscar un libro por ISBN o titulo

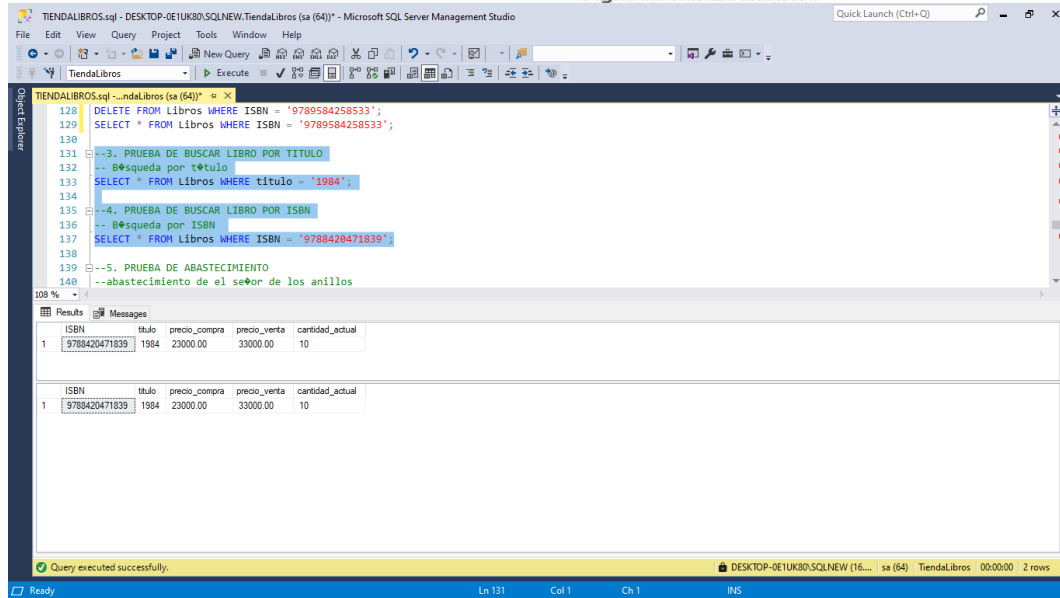
Colocamos esta prueba en un mismo ítem puesto que son mismo solo cambia la manera de buscar para realizar esta función lo que aremos es crear la siguiente consulta.

```

--3. PRUEBA DE BUSCAR LIBRO POR TITULO
-- Búsqueda por título
SELECT * FROM Libros WHERE titulo = '1984';

--4. PRUEBA DE BUSCAR LIBRO POR ISBN
-- Búsqueda por ISBN
SELECT * FROM Libros WHERE ISBN = '9788420471839';
  
```

Esto genera una consulta especifica permitiendo encontrar el libro 1994 ya sea por titulo o por el ISBN



Con esta función hacemos el cumplimiento del requerimiento 3 y 4 de 10 requerimiento

4. PRUEBA 5 Y 6 ABASTECIMIENTO Y VENTA DE UN LIBRO

para generar una venta de un ejemplar será fácil ya que solo tenemos que usar una sentencia

```
--5. PRUEBA DE ABASTECIMIENTO
--abastecimiento de el 1994
INSERT INTO Transacciones (ISBN, tipo_transaccion, cantidad) VALUES
('9788420471839', 2, 2);
SELECT * FROM Libros;
SELECT * FROM Caja ORDER BY id_movimiento DESC;

--6 PRUEBA VENTA DE LIBRO
INSERT INTO Transacciones (ISBN, tipo_transaccion, cantidad) VALUES
('9788420471839', 1, 2);
SELECT * FROM Libros;
SELECT * FROM Caja ORDER BY id_movimiento DESC;
```

Si recordamos la tabla de transacciones esta tiene dato de fecha, pero no hay que colocarlo, al contrario, eso es automático puesto que toma la fecha del sistema y la hora del mismo al momento de realizar la transacción. Cuando ejecutamos esto sale algo como esto.

TIENDALIBROS.sql - DESKTOP-0E1UK80\SQLNEW\TiendaLibros (sa (64)) - Microsoft SQL Server Management Studio

Quick Launch (Ctrl+Q)

File Edit View Query Project Tools Window Help

TIENDALIBROS

Execute

Object Explorer

108 %

TIENDALIBROS.sql - TiendaLibros (sa (64))

Results Messages

	ISBN	titulo	precio_compra	precio_venta	cantidad_actual
3	978078199335	the last days of magic	25000.00	45000.00	20
4	9783496929180	La Autopsia de Satán	31000.00	41000.00	21
5	9788420412146	El Señor de los Anillos	35000.00	65000.00	4
6	9788420471839	1984	23000.00	33000.00	12
7	9788466337991	El Código Da Vinci	22000.00	40000.00	8
8	9788498387087	Cien años de soledad	25000.00	45000.00	7
9	9789584284310	la casa de papel lesca...	20000.00	35000.00	40
10	9789587303636	Colombia mitos y leyen...	35000.00	65000.00	30

	id_movimiento	fecha_movimiento	tipo_movimiento	monto	saldo_actual	id_transaccion
1	49	2024-11-19 03:29:48.543	EGRESO	46000.00	2483000.00	28
2	48	2024-11-19 00:20:57.193	EGRESO	66000.00	2529000.00	NULL
3	47	2024-11-19 00:20:57.183	EGRESO	66000.00	2595000.00	27
4	46	2024-11-18 18:22:24.087	INGRESO	30000.00	2661000.00	NULL
5	45	2024-11-18 18:22:24.077	INGRESO	30000.00	2631000.00	26
6	44	2024-11-18 14:33:37.950	INGRESO	32000.00	2601000.00	NULL
7	43	2024-11-18 14:33:37.947	INGRESO	32000.00	2569000.00	25
8	42	2024-11-18 14:25:55.597	EGRESO	25000.00	2537000.00	NULL

	ISBN	titulo	precio_compra	precio_venta	cantidad_actual
1	9274619478294	Los Vagabundos de Dios	25000.00	32000.00	18
2	9746482739173	La mala hora	20000.00	35000.00	20
3	978078199335	the last days of magic	25000.00	45000.00	20
4	9783496929180	La Autopsia de Satán	31000.00	41000.00	21
5	9788420412146	El Señor de los Anillos	35000.00	65000.00	4
6	9788420471839	1984	23000.00	33000.00	10
7	9788466337991	El Código Da Vinci	22000.00	40000.00	8
8	9788498387087	Cien años de soledad	25000.00	45000.00	7

Query executed successfully.

DESKTOP-0E1UK80\SQLNEW (16... sa (64) TiendaLibros 00:00:00 10 rows

Ready Ln 1 Col 1 INS

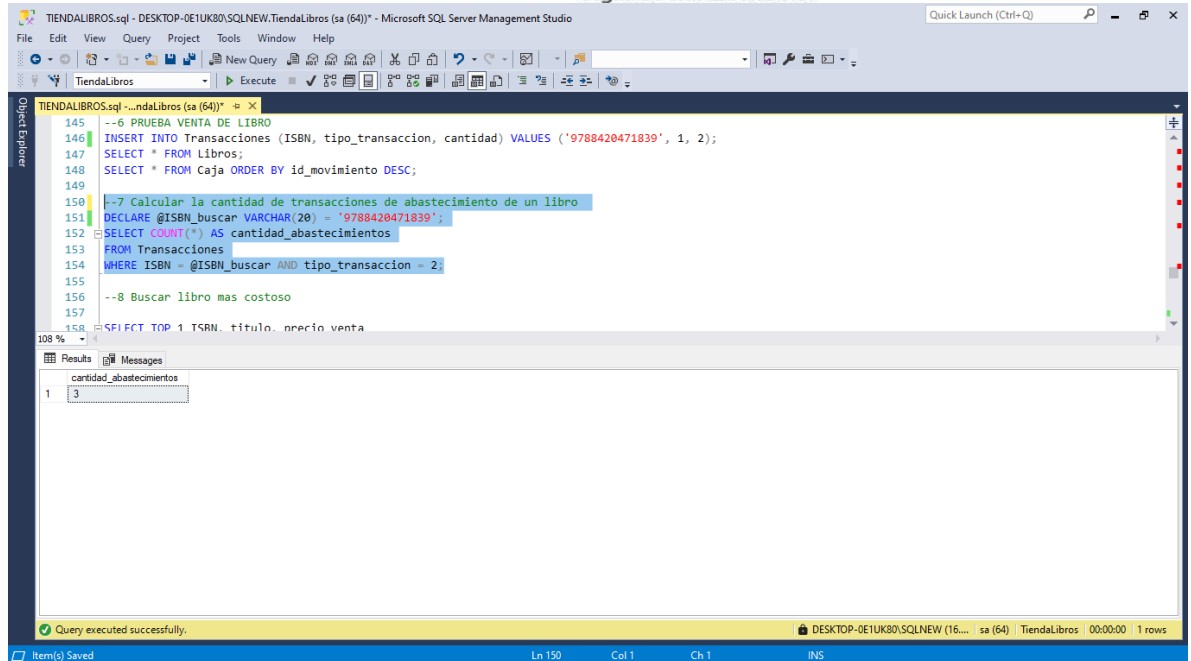
Donde nos muestra la venta de ejemplares del libro y el abastecimiento además como un plus colocamos de una vez que nos muestre el estado de caja y nos muestre si es ingreso o egreso. Así cumpliendo el requerimiento funcional 5 y 6 de 10 requerimientos.

5. PRUEBA 7 Calcular la cantidad de transacciones de abastecimiento de un libro

Para esta consulta sobre cantidades usaremos el siguiente código donde usamos un declare que es una función de declaración.

```
--7 Calcular la cantidad de transacciones de abastecimiento de un libro
DECLARE @ISBN_buscar VARCHAR(20) = '9788420471839';
SELECT COUNT(*) AS cantidad_abastecimientos
FROM Transacciones
WHERE ISBN = @ISBN_buscar AND tipo_transaccion = 2;
```

Esta función al ejecutarla nos muestra cuantas veces se abasteció un libro en este caso 1994 se abasteció 3 veces, así mismo nuestro cualquier dato de un libro que este en el catálogo de la biblioteca, con esta función hacemos el cumplimiento del requerimiento funcional 7 de 10 requerimientos.



6. PRUEBAS 8 Y 9 BUSCAR LIBRO MAS Y MENOS COSTOSO

- para este punto usamos el un top 1 para validar quien es el mas costoso y menos costoso a esto lo complementamos con un orden by de precio_venta uno de manera DESC para el mas costoso y otro con un ASC para el más costoso.

--8 Buscar libro mas costoso

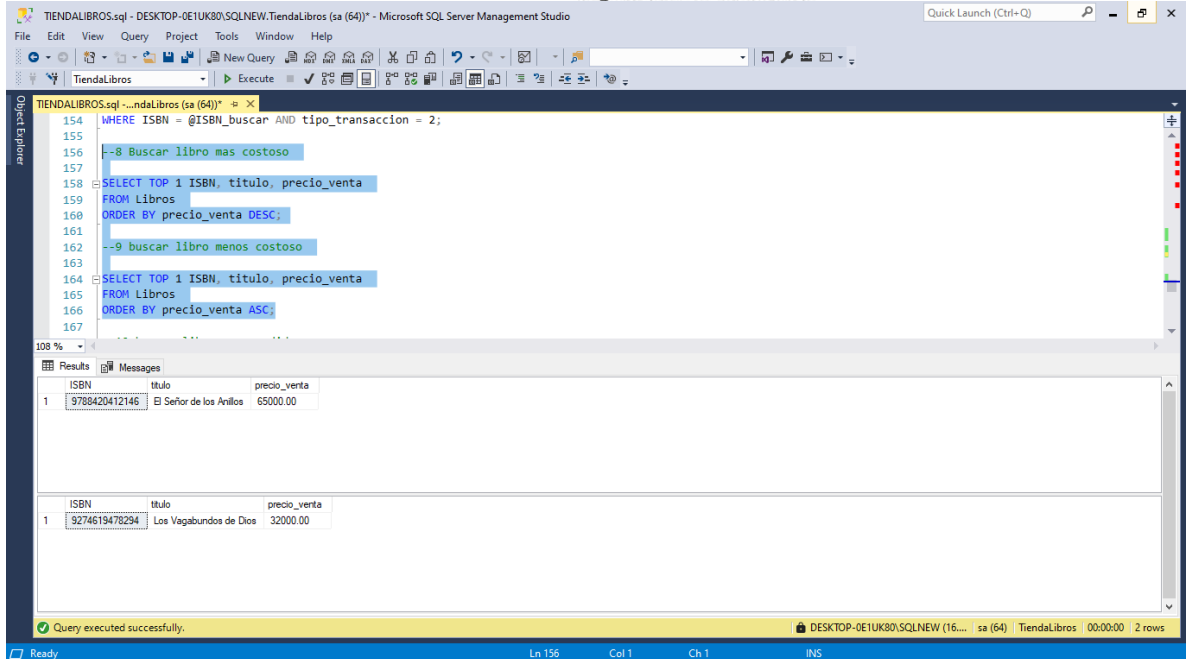
```

SELECT TOP 1 ISBN, titulo, precio_venta
FROM Libros
ORDER BY precio_venta DESC;
  
```

--9 buscar libro menos costoso

```

SELECT TOP 1 ISBN, titulo, precio_venta
FROM Libros
ORDER BY precio_venta ASC;
  
```



```

154 WHERE ISBN = @ISBN_buscar AND tipo_transaccion = 2;
155
156 --8 Buscar libro mas costoso
157
158 SELECT TOP 1 ISBN, titulo, precio_venta
159 FROM Libros
160 ORDER BY precio_venta DESC;
161
162 --9 buscar libro menos costoso
163
164 SELECT TOP 1 ISBN, titulo, precio_venta
165 FROM Libros
166 ORDER BY precio_venta ASC;
167
  
```

	ISBN	titulo	precio_venta
1	9788420412146	El Señor de los Anillos	65000.00

	ISBN	titulo	precio_venta
1	9274619478294	Los Vagabundos de Dios	32000.00

Query executed successfully.

7. PRUEBA 10 BUSCAR LIBRO MAS VENDIDO

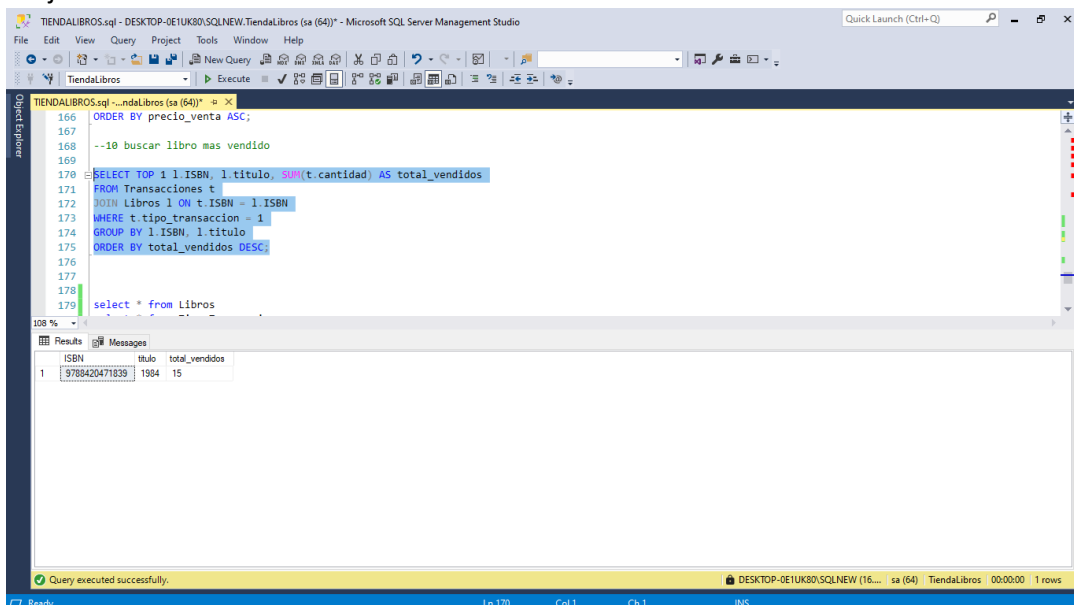
- para buscar el libro mas vendido usamos la siguiente sentencia

```

--10 buscar libro mas vendido

SELECT TOP 1 l.ISBN, l.titulo, SUM(t.cantidad) AS total_vendidos
FROM Transacciones t
JOIN Libros l ON t.ISBN = l.ISBN
WHERE t.tipo_transaccion = 1
GROUP BY l.ISBN, l.titulo
ORDER BY total_vendidos DESC;
  
```

al ejecutar este debería de salir esto.



```

166 ORDER BY precio_venta ASC;
167
168 --10 buscar libro mas vendido
169
170 SELECT TOP 1 l.ISBN, l.titulo, SUM(t.cantidad) AS total_vendidos
171 FROM Transacciones t
172 JOIN Libros l ON t.ISBN = l.ISBN
173 WHERE t.tipo_transaccion = 1
174 GROUP BY l.ISBN, l.titulo
175 ORDER BY total_vendidos DESC;
176
177
178 select * from Libros
179
  
```

	ISBN	titulo	total_vendidos
1	9788420471839	1904	15

Query executed successfully.