TRABAJO INTEGRADOR FINAL

Parte 1: Tutorial Construyendo una API RESTful con Python, Flask y MySQL

A continuación, se presenta un instructivo detallado para la creación de un sistema de CRUD de productos utilizando las siguientes tecnologías: HTML5, CSS3, Bootstrap 5, VUE3, MySQL, Python 3.x y Flask. En este proyecto, se implementará la arquitectura de software Modelo Vista Controlador (MVC), que divide las responsabilidades en tres partes fundamentales:

1. Modelo:
    - Se encarga de manejar los datos y la lógica de negocios.
    - Aquí se definirán las estructuras de datos necesarias para representar los productos y sus atributos.
    - Además, se implementarán las funciones para realizar las operaciones de creación, lectura, actualización y eliminación de productos en la base de datos.
2. Vista:
    - Responsable del diseño y la presentación de la interfaz de usuario.
    - Se utilizarán tecnologías como HTML5, CSS3 y Bootstrap 5 para crear una interfaz atractiva y fácil de usar.
    - Aquí se mostrarán los productos en forma de listas o tablas, se proporcionarán formularios para agregar y editar productos, y se incluirán botones para eliminar productos.
3. Controlador:
    - Encargado de manejar las interacciones entre el modelo y la vista.
    - El controlador recibirá las solicitudes del usuario a través de la interfaz y las enviará al modelo correspondiente para que se realicen las operaciones necesarias.
    - Luego, el controlador actualizará la vista con los cambios realizados en los productos.

A lo largo de este instructivo, se proporcionarán pasos detallados y ejemplos de código para cada una de las etapas del desarrollo. Se recomienda seguir el orden sugerido y revisar los ejemplos de código para comprender mejor cada paso.

¡Asegúrate de seguir las instrucciones y no dudes en consultar cualquier duda que surja durante el proceso! Recuerda que este proyecto te brindará la oportunidad de

aplicar los conocimientos adquiridos en el curso y fortalecer tus habilidades de desarrollo web.

¡Manos a la obra y disfruta del proceso de creación de tu propio sistema gestor de productos!

Material del BackEnd para generar la API

Aquí te proporcionamos el material necesario para crear el BackEnd de tu API, que permitirá implementar los métodos GET, DELETE, POST y PUT. Utiliza los siguientes recursos:

1. Video explicativo del BackEnd:
    - Accede al video en el siguiente enlace: Video explicativo del BackEnd.
    - Este video te guiará paso a paso en la generación de la API y te mostrará cómo implementar los métodos GET, DELETE, POST y PUT.
2. Repositorio en GitHub (BackEnd):
    - Ingresa al repositorio en GitHub haciendo clic en el enlace: Repositorio del BackEnd.
    - Aquí encontrarás el código fuente y los archivos necesarios para desarrollar el BackEnd de tu API.

Recuerda que estos recursos son fundamentales para comprender y llevar a cabo la creación del BackEnd de tu proyecto. Asegúrate de revisar el video explicativo detalladamente y utilizar el repositorio de GitHub como referencia durante el proceso de desarrollo.

Verificación de versiones de Python y pip en la consola de Windows

Antes de comenzar con el desarrollo de tu proyecto, es importante verificar que tienes instaladas las versiones correctas de Python y pip en tu sistema. Sigue estos pasos:

1. Abre la consola de Windows. Puedes hacerlo buscando "cmd" en el menú de inicio y seleccionando "Símbolo del sistema".
2. En la consola, escribe el siguiente comando y presiona Enter:

   ```
   python --version
   ```

   Este comando mostrará la versión de Python instalada en tu sistema. Asegúrate de que sea la versión correcta para el desarrollo de tu proyecto.
3. A continuación, escribe el siguiente comando y presiona Enter:

   ```
   pip --version
   ```

   Este comando mostrará la versión de pip instalada en tu sistema. Verifica que la versión sea la adecuada y coincida con la versión de Python.

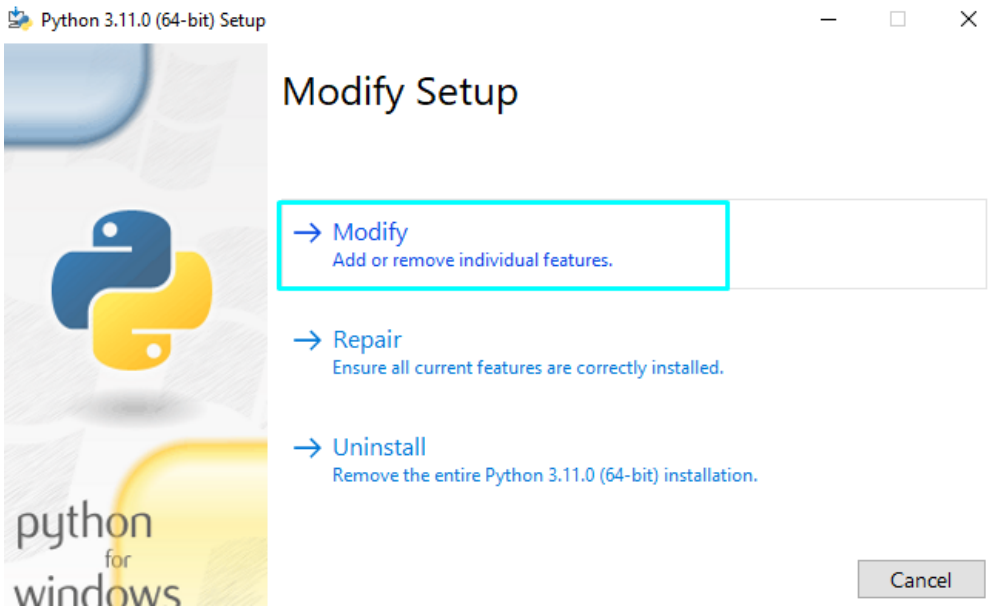
Si los comandos anteriores muestran las versiones correspondientes de Python y pip, puedes continuar con tu proyecto. En caso contrario, si no se muestran las versiones o hay algún error, es necesario solucionarlo.

Asegúrate de tener las versiones correctas de Python y pip instaladas en tu sistema para evitar posibles problemas durante el desarrollo de tu proyecto.

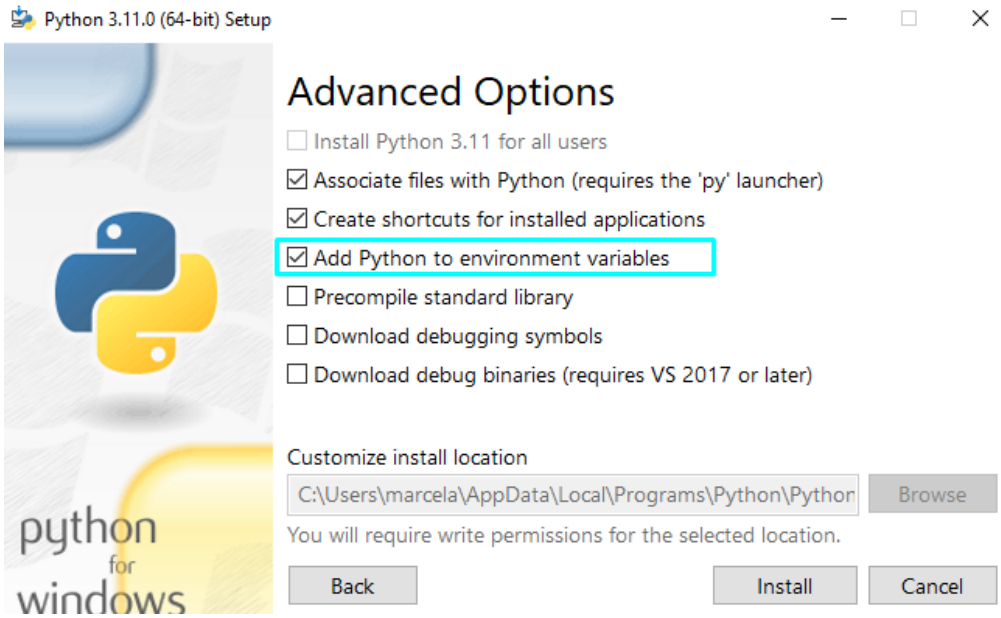
Solución de problemas en caso de errores de instalación

Si al ejecutar los comandos mencionados anteriormente no se muestran las versiones de Python y pip, es posible que haya un problema de instalación. A continuación, te indicamos cómo solucionarlo:

1. Si no se muestra la versión de Python y pip, es necesario ejecutar el instalador de Python nuevamente.
2. Ejecuta el instalador de Python y selecciona la opción "Modify" (Modificar).



3. Asegúrate de marcar la opción "Add Python to environment variables" (Agregar Python a las variables de entorno). Esto permitirá que puedas acceder a Python y pip desde cualquier ubicación en la consola de Windows.



4. Haz clic en "Next" (Siguiente) y sigue las instrucciones para completar el proceso de instalación.

5. Una vez finalizada la instalación, regresa al punto anterior y verifica nuevamente las versiones de Python y pip ejecutando los comandos correspondientes.

Si aún después de seguir estos pasos el problema persiste y las versiones no se muestran correctamente, puedes intentar solucionarlo utilizando la opción "Repair" (Reparar) en el instalador de Python. Selecciona esta opción y sigue las indicaciones para reparar la instalación.

Recuerda que es fundamental contar con las versiones correctas de Python y pip para poder instalar las dependencias necesarias y llevar a cabo tu proyecto sin inconvenientes.

Configuración del entorno virtual para el proyecto Back-End con Python y MySQL

Para poder trabajar en el proyecto de Back-End con Python y MySQL, vamos a crear un entorno virtual donde instalaremos los paquetes necesarios. Un entorno virtual es una herramienta que nos permite tener un entorno de trabajo aislado para cada proyecto, evitando conflictos entre paquetes y versiones.

Si aún no tienes instalado el paquete virtualenv, puedes instalarlo ejecutando el siguiente comando en la consola:

```
pip install virtualenv
```

```
Collecting virtualenv
  Downloading virtualenv-20.23.1-py3-none-any.whl (3.3 MB)
     |████████████████████████████████| 3.3 MB 2.2 MB/s
Collecting distlib<1,>=0.3.6
  Downloading distlib-0.3.6-py2.py3-none-any.whl (468 kB)
     |████████████████████████████████| 468 kB 6.4 MB/s
Collecting platformdirs<4,>=3.5.1
  Downloading platformdirs-3.8.0-py3-none-any.whl (16 kB)
Collecting filelock<4,>=3.12
  Downloading filelock-3.12.2-py3-none-any.whl (10 kB)
Installing collected packages: distlib, platformdirs, filelock, virtualenv
Successfully installed distlib-0.3.6 filelock-3.12.2 platformdirs-3.8.0 virtualenv-20.
23.1
```

Si al ejecutar el comando anterior recibes un error (o una advertencia), puedes ejecutar la siguiente línea para solucionarlo:

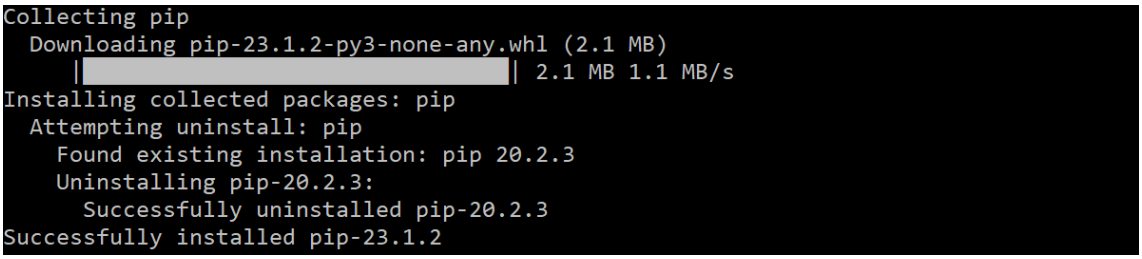
```
ERROR: Could not find a version that satisfies the requirement viertualenv (from versions:
ERROR: No matching distribution found for viertualenv

[notice] A new release of pip available: 22.3.1 -> 23.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
WARNING: You are using pip version 20.2.3; however, version 23.1.2 is available.
You should consider upgrading via the 'c:\users\marti\appdata\local\programs\python\py
thon39\python.exe -m pip install --upgrade pip' command.
```

```
python -m pip install --upgrade pip
```

Esto iniciará el proceso de actualización de **pip**. El comando buscará la versión más reciente de **pip** disponible en los repositorios y la instalará en tu sistema. Si ya tienes la última versión instalada, mostrará un mensaje indicando que no hay actualizaciones disponibles.



Explicación del comando:

- **python**: Hace referencia al intérprete de Python instalado en el sistema.
- **-m**: Es una opción que indica que se desea ejecutar un módulo como un script independiente.
- **pip**: Es el módulo de Python utilizado para la gestión de paquetes.
- **install**: Es un subcomando de **pip** que se utiliza para instalar paquetes.
- **--upgrade**: Es una opción que indica que se desea actualizar el paquete especificado.
- **pip**: Es el nombre del paquete que se desea actualizar.

En resumen, el comando **python -m pip install --upgrade pip** se utiliza para actualizar la herramienta **pip** de Python a su última versión, lo cual es recomendable hacer periódicamente para tener acceso a las últimas funcionalidades y mejoras.

Es importante tener en cuenta que el comando **python** debe estar asociado con la instalación de Python en tu sistema y debes asegurarte de tener los permisos necesarios para instalar o actualizar paquetes. Si estás utilizando un entorno virtual, asegúrate de activarlo antes de ejecutar el comando para actualizar **pip** en ese entorno específico.

Creación y Activación del Entorno Virtual de Trabajo

Para crear el entorno de trabajo, vamos a seguir los siguientes pasos:

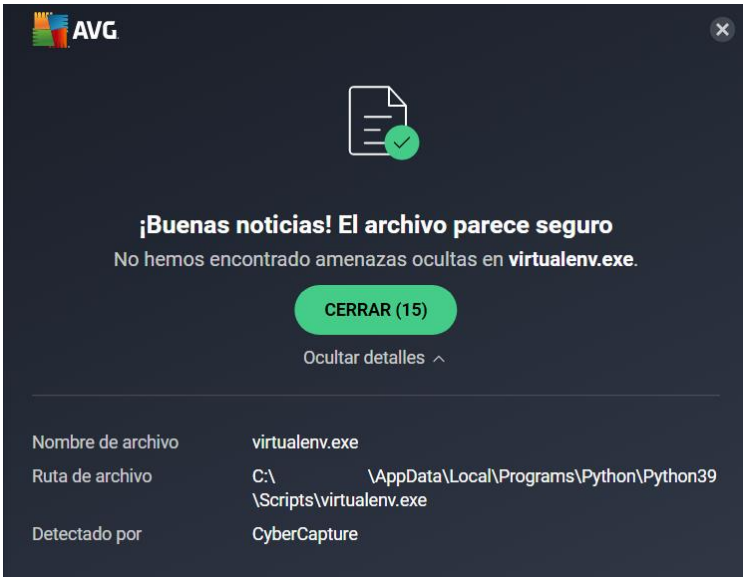
1. Abrir la consola del sistema.
2. Navegar hasta la carpeta de trabajo donde deseamos crear el entorno virtual. Por ejemplo, si queremos crear un entorno de trabajo virtual en una carpeta llamada "proyecto", nos ubicamos en esa carpeta en la consola del sistema.
3. Una vez en la carpeta de trabajo, ejecutamos el siguiente comando en la consola:

```
virtualenv proyecto
```

Esto creará el entorno de trabajo virtual en la carpeta "proyecto". Verás que se generan varios archivos y carpetas, incluyendo la carpeta "proyecto".

```
created virtual environment CPython3.9.1.final.0-64 in 3859ms
  creator CPython3Windows(dest=D:\TIF\proyecto, clear=False, no_vcs_ignore=False, glob
al=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=
copy, app_data_dir=C:\Users\marti\AppData\Local\pypa\virtualenv)
    added seed packages: pip==23.1.2, setuptools==67.8.0, wheel==0.40.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellAct
ivator,PythonActivator
```

Aviso: Ten en cuenta que durante esta parte del proceso es posible que el antivirus o el software de seguridad de tu sistema realicen análisis y verificaciones. Esto es completamente normal y forma parte de su función para garantizar la detección de posibles amenazas o actividades sospechosas. No te preocupes si notas que el antivirus se está ejecutando o si recibes notificaciones al respecto. Puedes continuar siguiendo los pasos del instructivo sin problemas.



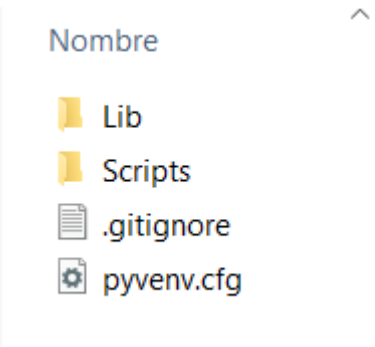
Si ejecutamos el comando dir dentro de la carpeta “proyecto” veremos las carpetas y archivos generados.

```
D:\TIF>cd proyecto

D:\TIF\proyecto>dir
 El volumen de la unidad D es DATA
 El número de serie del volumen es: B227-9DE2

 Directorio de D:\TIF\proyecto

24/06/2023  09:01 p. m.    <DIR>          .
24/06/2023  09:01 p. m.    <DIR>          ..
24/06/2023  09:01 p. m.                42 .gitignore
24/06/2023  09:01 p. m.    <DIR>          Lib
24/06/2023  09:01 p. m.               405 pyvenv.cfg
24/06/2023  09:01 p. m.    <DIR>          Scripts
               2 archivos            447 bytes
               4 dirs  110.515.355.648 bytes libres
```



4. Para activar el entorno virtual, ejecutamos el siguiente comando en la consola:

   En Windows:

```
proyecto\Scripts\activate
```

   En macOS o Linux:

```
source proyecto/bin/activate
```

```
D:\TIF\proyecto>cd Scripts

D:\TIF\proyecto\Scripts>activate

(proyecto) D:\TIF\proyecto\Scripts>
```

Al ejecutar este comando, verás que el prompt de la consola cambia y muestra el nombre del entorno virtual activado.

Una vez activado el entorno virtual, podrás instalar y utilizar las bibliotecas y paquetes específicos para tu proyecto sin interferir con las instalaciones globales de Python en tu sistema.

Recuerda que siempre debes activar el entorno virtual antes de trabajar en tu proyecto y desactivarlo una vez que hayas terminado. Para desactivar el entorno virtual, simplemente ejecuta el comando **deactivate** en la consola.

Recuerda reemplazar "proyecto" por el nombre que hayas elegido para tu entorno virtual.

Si deseas obtener más información sobre los entornos virtuales y su uso en Python, puedes consultar el siguiente enlace: https://omes-va.com/virtualenv-python/

Recuerda que el uso de un entorno virtual es una buena práctica para el desarrollo de proyectos en Python, ya que ayuda a mantener la organización de las dependencias y evita conflictos con otros proyectos o versiones del sistema.

Configuración del BackEnd: Instalación de Flask y paquetes adicionales

Comencemos con la configuración del BackEnd del proyecto. Abre Visual Studio Code (VSC) y abre la carpeta "proyecto" que acabamos de crear.

A continuación, vamos a instalar Flask, el framework que nos ayudará a ser más productivos en el desarrollo de Python y bases de datos. Dependiendo de tu sistema operativo, sigue estos pasos en la terminal de VSC:

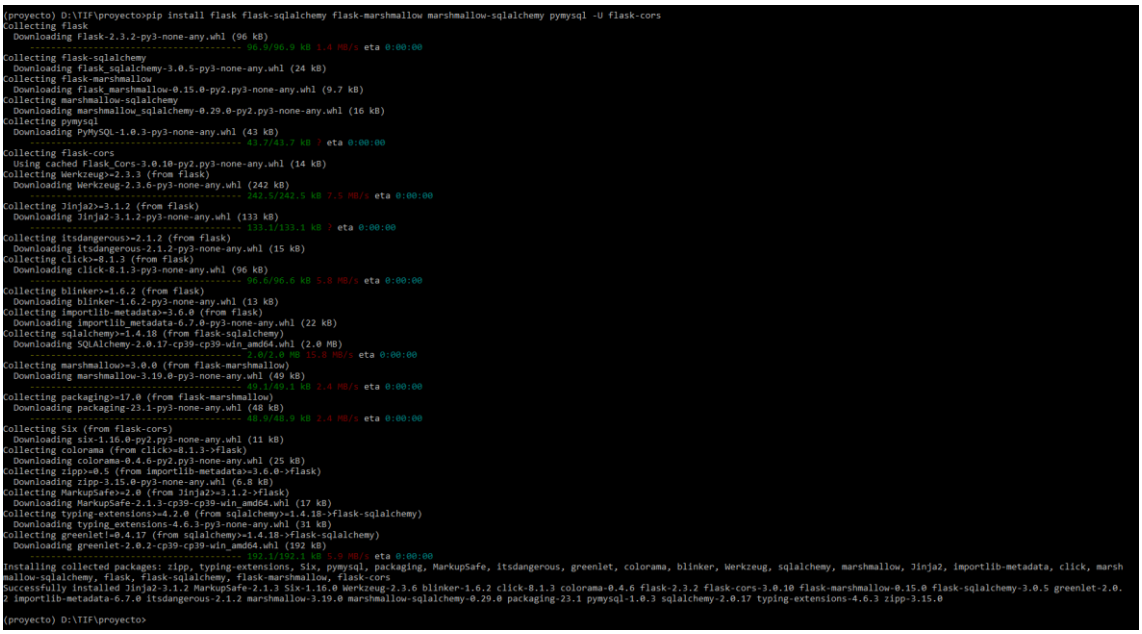
- Para Windows: Ejecuta el siguiente comando en la terminal: **pip install flask**
- Para macOS: Ejecuta el siguiente comando en la terminal: **pip3 install flask**
- Para Linux: Ejecuta el siguiente comando en la terminal: **sudo apt install python3-pip**

Luego, instala los siguientes paquetes en la terminal, dependiendo de tu sistema operativo:

```
pip install flask flask-sqlalchemy flask-marshmallow marshmallow-sqlalchemy pymysql -U flask-cors
```

Es importante mencionar que estos paquetes nos brindan funcionalidades adicionales para nuestro proyecto. Aquí tienes más información sobre cada uno de ellos:

- Flask: Es un framework que nos permite desarrollar aplicaciones web de manera eficiente con Python. Puedes obtener más información en: https://flask-es.readthedocs.io/
- Marshmallow: Es una biblioteca que nos ayuda a manejar la serialización y deserialización de objetos complejos en Python. Es decir, es una biblioteca independiente de ORM/ODM/framework para convertir tipos de datos complejos, como objetos, hacia y desde tipos de datos nativos de Python. Puedes obtener más información en: https://marshmallow.readthedocs.io/en/stable/
- SQL Alchemy: Es una herramienta que nos permite trabajar con bases de datos relacionales en Python. Proporciona un ORM (Object Relational Mapper) para asociar nuestras clases en Python con las tablas en la base de datos. Es decir, es un kit de herramientas SQL y un ORM (Object Relational Mapper) que permite a las clases en Python asociarlas a tablas en bases de datos relacionales. Puedes obtener más información en: https://flask-es.readthedocs.io/patterns/sqlalchemy/
- Flask-CORS: Es una extensión de Flask que nos permite manejar las solicitudes de recursos cruzados (CORS) en nuestra aplicación. Esto nos permitirá acceder al JSON generado por el BackEnd desde el FrontEnd.


(proyecto) D:\TIF\proyecto>pip install flask flask-sqlalchemy flask-marshmallow marshmallow-sqlalchemy pymysql -U flask-cors
Collecting flask
Downloading flask-2.3.2-py3-none-any.whl (96 kB)
96.9/96.9 kB 1.4 MB/s eta 0:00:00
Collecting flask-sqlalchemy
Downloading flask_sqlalchemy-3.0.5-py3-none-any.whl (24 kB)
Collecting flask-marshmallow
Downloading flask_marshmallow-0.15.0-py2.py3-none-any.whl (9.7 kB)
Collecting marshmallow-sqlalchemy
Downloading marshmallow_sqlalchemy-0.29.0-py2.py3-none-any.whl (16 kB)
Collecting pymysql
Downloading PyMySQL-1.0.3-py3-none-any.whl (43 kB)
43.7/43.7 kB ? eta 0:00:00
Collecting flask-cors
Using cached Flask_Cors-3.0.10-py2.py3-none-any.whl (14 kB)
Collecting Werkzeug>=2.3.3 (from flask)
Downloading Werkzeug-2.3.6-py3-none-any.whl (242 kB)
242.5/242.5 kB 7.5 MB/s eta 0:00:00
Collecting Jinja2>=3.1.2 (from flask)
Downloading Jinja2-3.1.2-py3-none-any.whl (133 kB)
133.1/133.1 kB ? eta 0:00:00
Collecting itsdangerous>=2.1.2 (from flask)
Downloading itsdangerous-2.1.2-py3-none-any.whl (15 kB)
Collecting click>=8.1.3 (from flask)
Downloading click-8.1.3-py3-none-any.whl (96 kB)
96.6/96.6 kB 5.8 MB/s eta 0:00:00
Collecting blinker>=1.6.2 (from flask)
Downloading blinker-1.6.2-py3-none-any.whl (13 kB)
Collecting importlib-metadata>=3.6.0 (from flask)
Downloading importlib_metadata-6.7.0-py3-none-any.whl (22 kB)
Collecting sqlalchemy>=1.4.18 (from flask-sqlalchemy)
Downloading SQLAlchemy-2.0.17-cp39-cp39-win_amd64.whl (2.0 MB)
2.0/2.0 MB 15.8 MB/s eta 0:00:00
Collecting marshmallow>=3.0.0 (from flask-marshmallow)
Downloading marshmallow-3.19.0-py3-none-any.whl (49 kB)
49.1/49.1 kB 2.4 MB/s eta 0:00:00
Collecting packaging>=17.0 (from flask-marshmallow)
Downloading packaging-23.1-py3-none-any.whl (48 kB)
48.9/48.9 kB 2.4 MB/s eta 0:00:00
Collecting Six (from flask-cors)
Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting colorama (from click>=8.1.3->flask)
Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Collecting zipp>=0.5 (from importlib-metadata>=3.6.0->flask)
Downloading zipp-3.15.0-py3-none-any.whl (6.8 kB)
Collecting MarkupSafe>=2.0 (from Jinja2>=3.1.2->flask)
Downloading MarkupSafe-2.1.3-cp39-cp39-win_amd64.whl (17 kB)
Collecting typing-extensions>=4.2.0 (from sqlalchemy>=1.4.18->flask-sqlalchemy)
Downloading typing_extensions-4.6.3-py3-none-any.whl (31 kB)
Collecting greenlet!=0.4.17 (from sqlalchemy>=1.4.18->flask-sqlalchemy)
Downloading greenlet-2.0.2-cp39-cp39-win_amd64.whl (192 kB)
192.1/192.1 kB 5.9 MB/s eta 0:00:00
Installing collected packages: zipp, typing-extensions, Six, pymysql, packaging, MarkupSafe, itsdangerous, greenlet, colorama, blinker, Werkzeug, sqlalchemy, marshmallow, Jinja2, importlib-metadata, click, marsh
mallow-sqlalchemy, flask, flask-sqlalchemy, flask-marshmallow, flask-cors
Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.3 Six-1.16.0 Werkzeug-2.3.6 blinker-1.6.2 click-8.1.3 colorama-0.4.6 flask-2.3.2 flask-cors-3.0.10 flask-marshmallow-0.15.0 flask-sqlalchemy-3.0.5 greenlet-2.0.
2 importlib-metadata-6.7.0 itsdangerous-2.1.2 marshmallow-3.19.0 marshmallow-sqlalchemy-0.29.0 packaging-23.1 pymysql-1.0.3 sqlalchemy-2.0.17 typing-extensions-4.6.3 zipp-3.15.0
(proyecto) D:\TIF\proyecto>

Creación de la base de datos MySQL

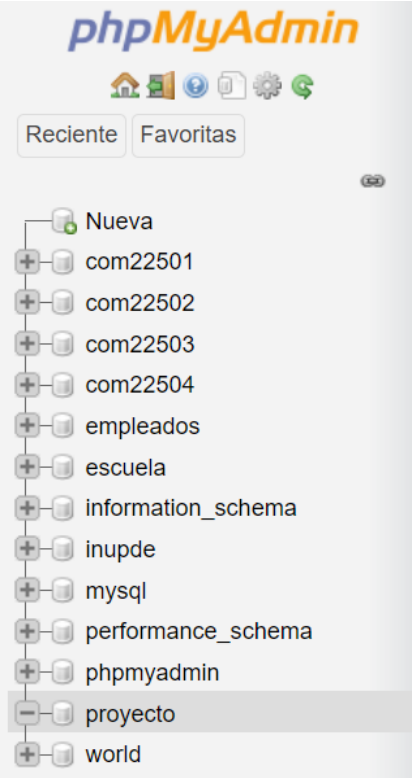
Para continuar con el proceso, vamos a crear una base de datos en MySQL llamada "proyecto". Sigue los pasos a continuación:

1. Abre tu gestor de bases de datos MySQL (por ejemplo, phpMyAdmin).
2. Busca la opción para crear una nueva base de datos y proporciona el nombre "proyecto".
3. Guarda los cambios y refresca la lista de bases de datos para verificar que se haya creado correctamente.

Ahora deberías ver la base de datos "proyecto" en la lista de bases de datos disponibles. Esto confirma que se ha creado correctamente y está lista para ser utilizada en nuestro proyecto.

Recuerda que este nombre es solo un ejemplo, puedes utilizar el nombre que desees para tu base de datos. Asegúrate de usar el mismo nombre en tu configuración de conexión con la base de datos en el código del proyecto.

Si tienes alguna pregunta o enfrentas algún problema durante este proceso, no dudes en consultar a tu docente o compañeros de clase para recibir asistencia.


phpMyAdmin
Reciente
Favoritas
Nueva
com22501
com22502
com22503
com22504
empleados
escuela
information_schema
inupde
mysql
performance_schema
phpmyadmin
proyecto
world

Creación de la aplicación Flask

Vamos a continuar con el proyecto.

1. Dentro de la carpeta "proyecto", crea un archivo llamado "app.py" utilizando Visual Studio Code u otro editor de texto de tu elección.

2. Abre el archivo "app.py" y copia el siguiente código: **app.py**

```python
'''
Este código importa diferentes módulos y clases necesarios para el desarrollo de una aplicación Flask.

Flask: Es la clase principal de Flask, que se utiliza para crear instancias de la aplicación Flask.
jsonify: Es una función que convierte los datos en formato JSON para ser enviados como respuesta desde la API.
request: Es un objeto que representa la solicitud HTTP realizada por el cliente.
CORS: Es una extensión de Flask que permite el acceso cruzado entre dominios (Cross-Origin Resource Sharing), lo cual es útil cuando se desarrollan aplicaciones web con frontend y backend separados.
SQLAlchemy: Es una biblioteca de Python que proporciona una abstracción de alto nivel para interactuar con bases de datos relacionales.
Marshmallow: Es una biblioteca de serialización/deserialización de objetos Python a/desde formatos como JSON.
Al importar estos módulos y clases, estamos preparando nuestro entorno de desarrollo para utilizar las funcionalidades que ofrecen.

'''
# Importa las clases Flask, jsonify y request del módulo flask
from flask import Flask, jsonify, request
# Importa la clase CORS del módulo flask_cors
from flask_cors import CORS
# Importa la clase SQLAlchemy del módulo flask_sqlalchemy
from flask_sqlalchemy import SQLAlchemy
# Importa la clase Marshmallow del módulo flask_marshmallow
from flask_marshmallow import Marshmallow
```

3. Guarda el archivo "app.py".

4. Abre la terminal o línea de comandos y navega hasta la ubicación de tu proyecto.

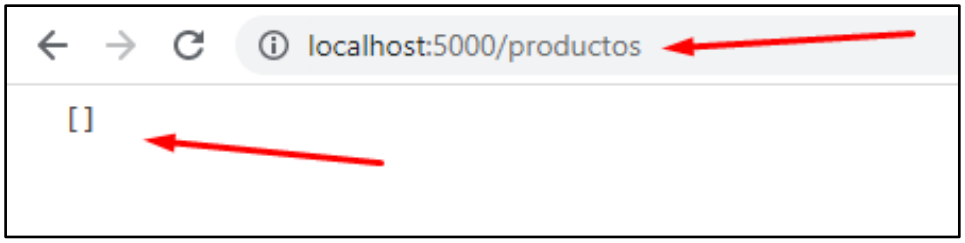
5. En la terminal, ejecuta el siguiente comando para iniciar el servidor Flask:

```
python app.py
```

6. Verás un mensaje en la terminal indicando que el servidor se ha iniciado correctamente.

```
(proyecto) D:\TIF\proyecto>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
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
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 119-597-640
```

7. Abre tu navegador web y visita la siguiente dirección: http://localhost:5000/productos



Si ves **[ ]** en tu navegador, ¡eso significa que todo está funcionando correctamente! Este resultado indica que la API ha devuelto una lista vacía en formato JSON. Esto sucede porque la tabla de productos en la base de datos está recién creada y aún no contiene ningún dato.

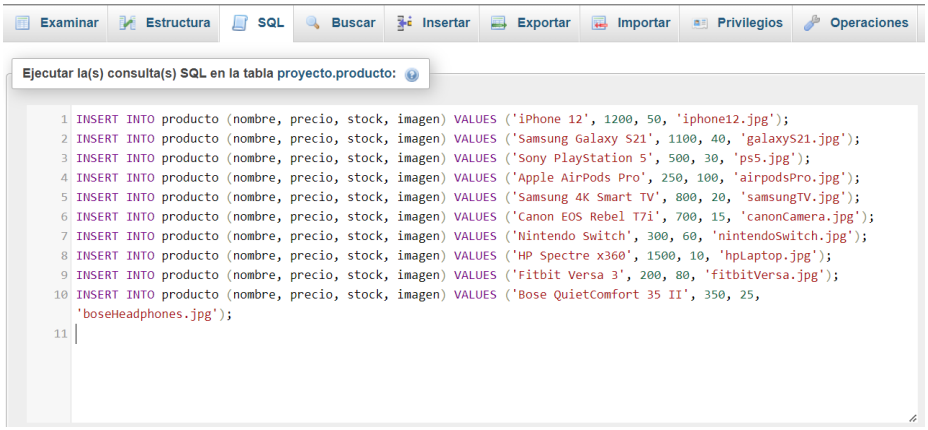
Este paso te confirma que la API está lista para manejar y enviar datos de la tabla de productos. Ahora puedes proceder a realizar otras operaciones CRUD, como agregar nuevos productos, actualizar registros existentes o eliminar productos.

Pruebas con la Aplicación: Carga de Productos y Pruebas en Navegador y Herramientas de API

A continuación, te mostramos los pasos para cargar productos en la base de datos, realizar pruebas y verificar los resultados en el navegador y en las herramientas Thunder Client o Postman.

1. Cargar productos en la base de datos:

    - Utiliza tu cliente de base de datos para crear y guardar algunos productos en la base de datos. Asegúrate de incluir todos los datos necesarios, como nombre, precio, stock e imagen. Podés usar el siguiente script para realizar esta tarea: **INSERT INTO producto.sql**



2. Probar en el navegador:

    - Abre tu navegador web y accede a la siguiente URL: **http://localhost:5000/productos** . Verás un JSON con los datos de los productos que acabas de crear. Si los productos no se muestran correctamente, verifica que hayas guardado los productos correctamente en la base de datos.

3. Probar en el navegador con un ID específico:

    - Accede a la siguiente URL en tu navegador: **http://localhost:5000/productos/1** . Se mostrará un JSON con el producto que tenga el ID igual a 1. Puedes probar con otros IDs para obtener los productos correspondientes.

4. Utilizar Thunder Client o Postman:

    - Abre la extensión Thunder Client en Visual Studio Code o el programa Postman en tu escritorio.

    - Configura una nueva solicitud para cada uno de los siguientes métodos: GET, DELETE, POST y PUT.

    - Para cada solicitud, utiliza la URL **http://localhost:5000/productos** y agrega los parámetros necesarios según el método:

- GET: Realiza una solicitud GET para obtener todos los productos. Deberías recibir un JSON con todos los registros de productos.
- DELETE: Realiza una solicitud DELETE para eliminar un producto específico. Puedes especificar el ID del producto en la URL, por ejemplo, **http://localhost:5000/productos/1** para eliminar el producto con ID igual a 1.
- POST: Realiza una solicitud POST para crear un nuevo producto. Envía los datos del nuevo producto en formato JSON en el cuerpo de la solicitud.
- PUT: Realiza una solicitud PUT para actualizar un producto existente. Especifica el ID del producto en la URL y envía los datos actualizados en formato JSON en el cuerpo de la solicitud.

- Verifica las respuestas recibidas para cada solicitud y asegúrate de que los resultados coincidan con tus expectativas.

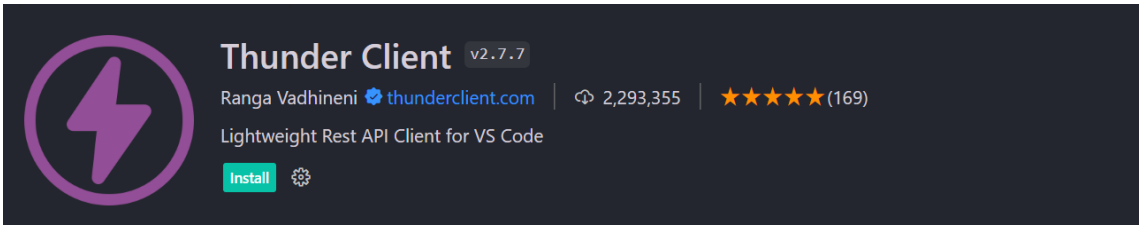
Recuerda que tanto en el navegador como en Thunder Client o Postman, debes asegurarte de tener el servidor Flask en ejecución para poder interactuar con la aplicación y la base de datos.

Sobre Thunder Client

Thunder Client es una extensión de Visual Studio Code que permite realizar solicitudes HTTP directamente desde el editor. Es una herramienta útil para probar y depurar servicios web, ya que brinda la capacidad de enviar solicitudes GET, POST, PUT y DELETE, así como configurar encabezados y enviar datos en formato JSON o formulario. Thunder Client muestra las respuestas de manera clara y organizada, lo que facilita la revisión de los resultados. Con esta extensión, los desarrolladores pueden realizar pruebas de API de manera eficiente y colaborar en el desarrollo de servicios web.

Website: www.thunderclient.com

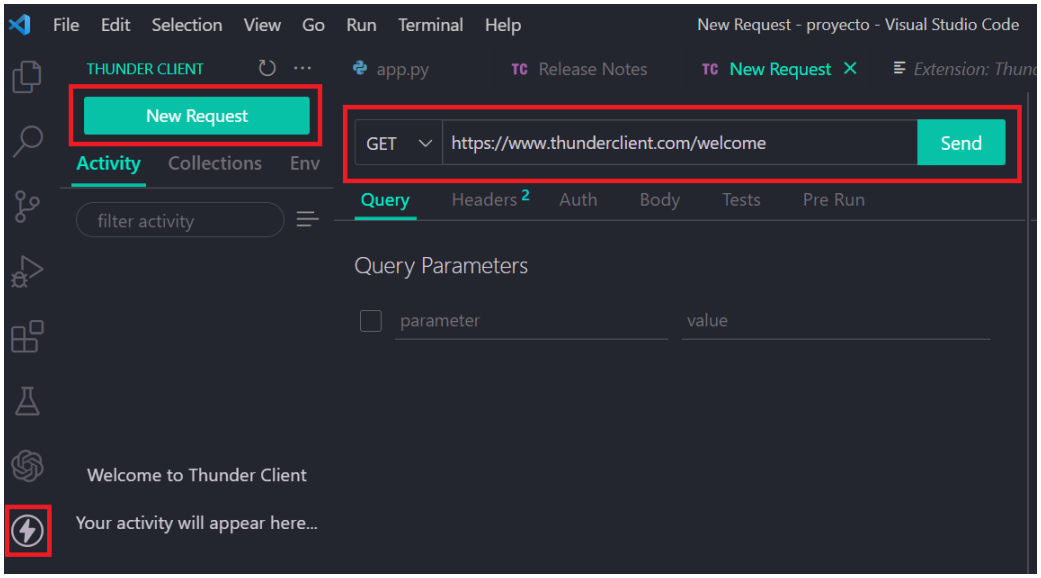
Extensión: https://marketplace.visualstudio.com/items?itemName=rangav.vscode-thunder-client



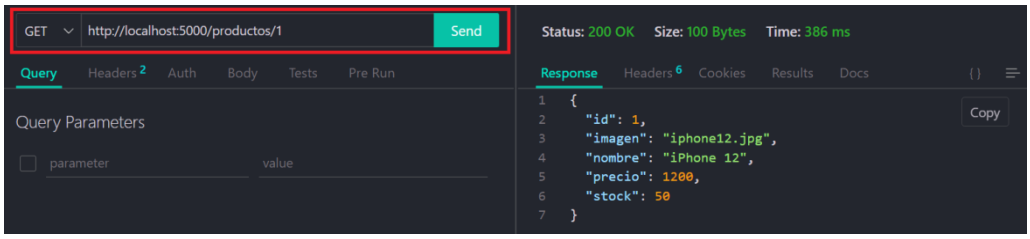
Uso de Thunder Client

1. Instala la extensión Thunder Client en Visual Studio Code.

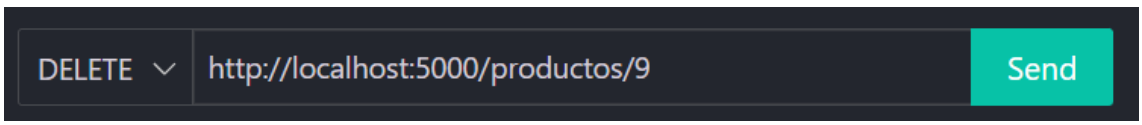
2. Abre un archivo de código en Visual Studio Code.
3. Abre la vista de Thunder Client haciendo clic en el icono "Thunder Bolt" en la barra lateral izquierda.

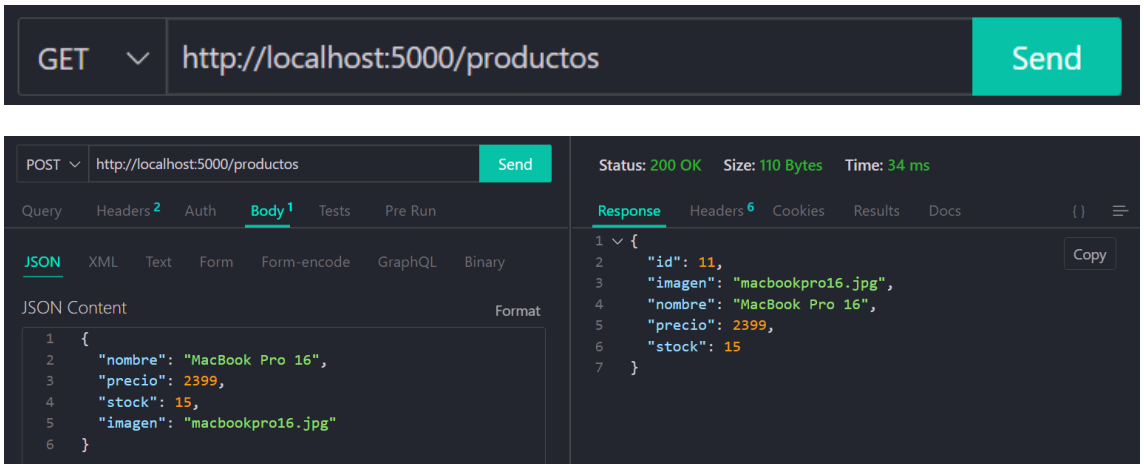


4. En la vista de Thunder Client, crea una nueva solicitud haciendo clic en el botón "+ New Request".
5. Completa la URL del servicio web que deseas probar en el campo correspondiente.
6. Selecciona el método HTTP adecuado (GET, POST, PUT, DELETE, etc.).
7. Agrega los encabezados y los parámetros necesarios según las especificaciones del servicio web.
8. Haz clic en el botón "Send" para enviar la solicitud al servicio web.
9. Thunder Client mostrará la respuesta del servicio web en la vista correspondiente, incluyendo el código de estado, los encabezados y el cuerpo de la respuesta.



10. Puedes usar Thunder Client para realizar pruebas adicionales, modificar las solicitudes existentes o crear nuevas solicitudes según tus necesidades.





Para este último ejemplo, pueden utilizar el formato JSON que figura en la imagen. De esa forma estarán insertando un nuevo registro en la base de datos.

Recuerda que Thunder Client proporciona una interfaz amigable para probar y depurar servicios web directamente desde Visual Studio Code, lo que te permite ahorrar tiempo y simplificar el proceso de desarrollo de aplicaciones web.

Posibles Errores Comunes y Soluciones

Durante el desarrollo del proyecto, es posible que te encuentres con algunos errores que pueden afectar el funcionamiento de la API. A continuación, se presentan algunos ejemplos de posibles errores y sus soluciones:

1. Error de conexión a la base de datos: Si encuentras dificultades para conectar la API a la base de datos, asegúrate de que la configuración de la URL, nombre de usuario y contraseña de la base de datos sean correctos. Verifica también si el servicio de la base de datos está en ejecución.

2. Error de dependencias faltantes: Si al intentar ejecutar la API obtienes mensajes de error relacionados con paquetes o módulos faltantes, es posible que debas instalar las dependencias requeridas. Asegúrate de tener todas las bibliotecas necesarias instaladas en tu entorno virtual ejecutando el comando **pip install -r requirements.txt** en la terminal.

3. Error de conflicto de versiones: Puede ocurrir que las versiones de las bibliotecas utilizadas en el proyecto no sean compatibles entre sí, lo que puede generar errores de ejecución. En ese caso, es recomendable revisar las versiones de las dependencias especificadas en el archivo **requirements.txt** y asegurarse de que sean compatibles. Puedes probar actualizar las versiones de las bibliotecas o buscar soluciones alternativas en la documentación oficial de cada biblioteca.

4. Error de puerto en uso: Si al intentar ejecutar la API obtienes un error que indica que el puerto está en uso, puede ser porque otro servicio o instancia de la API ya está en ejecución. Cambia el número de puerto en el archivo **app.py** y asegúrate de que no esté siendo utilizado por otro proceso en tu sistema.

5. Error al reiniciar el servicio/API: Si el servicio/API se detiene o necesitas reiniciarlo, sigue estos pasos para reactivarlo:

    - Activa tu entorno virtual ejecutando el comando correspondiente en la terminal.
    - Navega hasta la ubicación del archivo **app.py** en tu proyecto.
    - Ejecuta el comando **python app.py** para iniciar el servicio/API.

Recuerda que estos son solo ejemplos de posibles errores. Si encuentras otros problemas, te recomendamos buscar soluciones en línea, consultar la documentación oficial de las bibliotecas utilizadas o buscar ayuda en comunidades de desarrollo.

Reactivando el Servicio/API: Instructivo para reiniciar y activar el entorno virtual

A continuación, te presentamos algunos errores comunes que podrías encontrar durante el desarrollo. Si te encuentras con alguno de estos errores, puedes seguir este instructivo para resolverlo. Ten en cuenta que también podrían surgir otros errores como los mencionados anteriormente, pero los siguientes ejemplos te servirán como

referencia.

```
D:\>cd TIF

D:\TIF>cd proyecto

D:\TIF\proyecto>python app.py
Traceback (most recent call last):
  File "D:\TIF\proyecto\app.py", line 18, in <module>
    from flask_sqlalchemy import SQLAlchemy
ModuleNotFoundError: No module named 'flask_sqlalchemy'
```

En este caso, el entorno virtual no se encuentra activo.

Veamos otro ejemplo:

```
D:\TIF\proyecto>cd Scripts

D:\TIF\proyecto\Scripts>activate

(proyecto) D:\TIF\proyecto\Scripts>python app.py
             \appdata\local\programs\python\python39\python.exe: can't open
file 'D:\TIF\proyecto\Scripts\app.py': [Errno 2] No such file or directory
```

En este caso, no se encuentran ubicados en el directorio correspondiente para ejecutar la app.

Para volver a activar el servicio/API en caso de que se detenga o sea necesario reiniciar el entorno virtual pueden seguir los siguientes pasos:

1. Abre una terminal o línea de comandos en tu sistema operativo.
2. Navega hasta el directorio donde se encuentra el proyecto de la API.
3. Activa el entorno virtual ejecutando el siguiente comando, dependiendo del entorno virtual

```
D:\>cd TIF

D:\TIF>cd proyecto
```

```
D:\TIF\proyecto>cd Scripts

D:\TIF\proyecto\Scripts>activate
```

4. Una vez que el entorno virtual esté activado, ejecuta el archivo app.py para iniciar el servicio/API nuevamente. Utiliza el siguiente comando:

```
(proyecto) D:\TIF\proyecto\Scripts>cd..

(proyecto) D:\TIF\proyecto>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
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
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 119-597-640
127.0.0.1 - - [25/Jun/2023 10:37:48] "GET /productos HTTP/1.1" 200 -
127.0.0.1 - - [25/Jun/2023 10:37:48] "GET /favicon.ico HTTP/1.1" 404
```

5. Verifica que el servicio/API se haya reiniciado correctamente. Puedes probar accediendo a las rutas/endpoint definidos en tu API desde el navegador o utilizando herramientas como Thunder Client o Postman para enviar solicitudes GET, POST, DELETE, PUT, etc.

Recuerda que cada vez que necesites volver a iniciar el servicio/API, primero debes activar el entorno virtual correspondiente.

Recursos Adicionales y Material de Referencia

- Backend MVC: Enlace al Video
- Repositorio de GitHub del Backend MVC: Enlace al Repositorio

Estos recursos adicionales proporcionan información adicional y guías paso a paso para comprender el concepto y la implementación del patrón MVC en el backend. El repositorio de GitHub contiene el código fuente del Backend MVC como referencia para su estudio y práctica.