

C.R.U.D. de Productos

Create Read Update Delete

Instructivo para proyecto API REST de CRUD de productos con las siguientes tecnologías: HTML5, CSS3, Bootstrap 5, VUE3, MySQL, Python 3.11 y Flask. con Arquitectura de Software: Modelo Vista Controlador. Las tres partes del patrón de diseño de software **MVC** se pueden describir de la siguiente manera: **Modelo**: Maneja datos y lógica de negocios. Vista: Se encarga del diseño y presentación. Controlador: Enruta comandos a los **modelos** y vistas.

Productos				
<button>Nuevo</button>				
Id	Nombre	Precio	Stock	Acciones
1	prueba	100	10	<button>Modificar</button> <button>Eliminar</button>
5	tv 48"	100000	50	<button>Modificar</button> <button>Eliminar</button>
6	notebook 15"	170000	15	<button>Modificar</button> <button>Eliminar</button>

Video explicativo del BackEnd para generar la Api que permita los métodos GET, DELETE, POST y PUT: <https://youtu.be/KC1ELsgg5tM>

GitHub (backend): https://github.com/MarcelaCerde/backend_productos.git

1- Antes que nada, en la consola de Windows probar:

```
python --version
```

```
pip --version
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2251]
(c) Microsoft Corporation. Todos los derechos reservados.

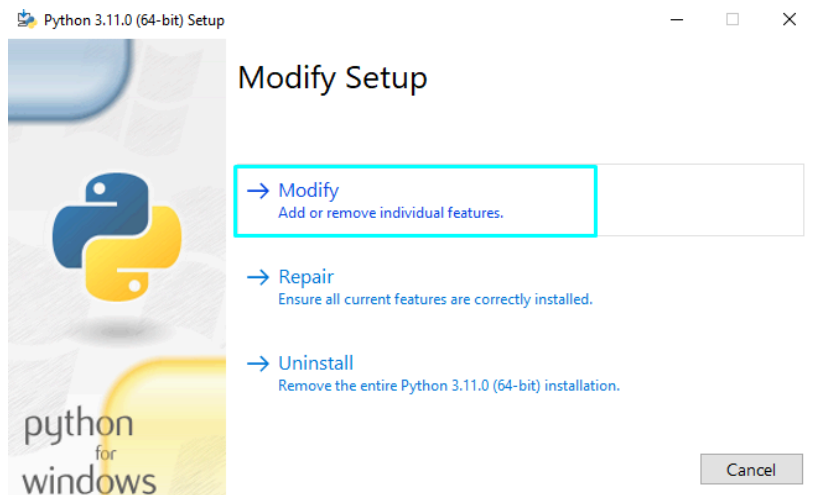
C:\Users\marcela\Desktop\CAC20222doCuatri\python>python --version
Python 3.11.0

C:\Users\marcela\Desktop\CAC20222doCuatri\python>pip --version
pip 22.3.1 from C:\Users\marcela\AppData\Local\Programs\Python\Python311\Lib\site-packages\pip (python 3.11)

C:\Users\marcela\Desktop\CAC20222doCuatri\python>
```

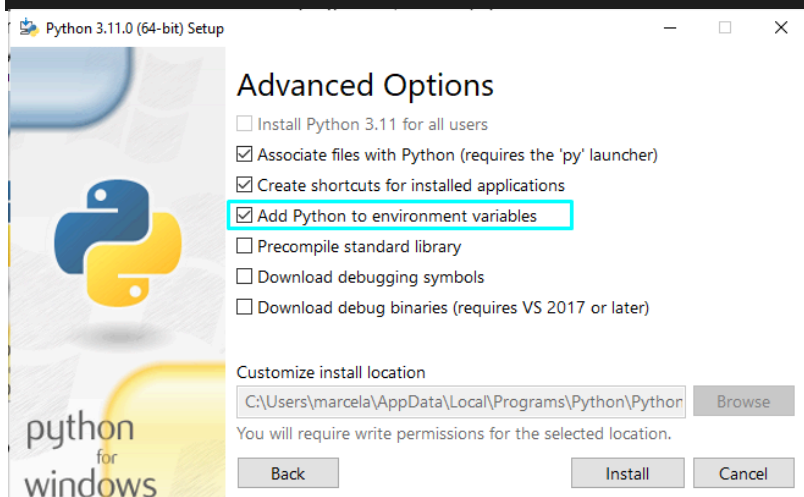
y tiene que mostrar la versión de ambos programas, sino hay que solucionarlo !!!

Si da error, no van a poder instalar lo necesario para el proyecto, van a tener que ejecutar el instalador de Python y hacer click en Modify



y tildar **Add Python to environment variables**

```
window.location.href = "index.html";
```



y volver al punto 1 y si sigue sin funcionar, en el instalador. presionar **Reparar**

2- Back End con Python y MySQL que luego subiremos al Hosting PythonAnywhere

Vamos a crear un entorno virtual para instalar los paquetes del proyecto

Un *entorno virtual* es un entorno de Python parcialmente aislado que permite instalar paquetes para que los use una aplicación en particular, en lugar de instalarlos en todo el sistema

Si quieren seguir leyendo: <https://omes-va.com/virtualenv-python/>

Si no tienen instalado virtualenv:

En la consola escribir:

```
pip install virtualenv
```

Si da el siguiente error:

```
C:\Users\marcela\Desktop\CAC 2023 1er Cuatrimestre\CRUD>pip install virtualenv
ERROR: Could not find a version that satisfies the requirement virtualenv (from versions:
ERROR: No matching distribution found for virtualenv

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

C:\Users\marcela\Desktop\CAC 2023 1er Cuatrimestre\CRUD>python -m pip install --upgrade pip
```

ejecutar la última línea

Vamos a crear el entorno de trabajo

En la consola **nos movemos a la carpeta de trabajo** y si queremos crear un entorno de trabajo virtual en la carpeta que se llame por ejemplo “proyecto” escribimos en la consola del sistema

```
virtualenv proyecto
```

notamos que nos creó muchos archivos y carpetas, entre ellas una carpeta “proyecto”. luego hay que activar el entorno virtual, ejecutando “**activate**” que se encuentra dentro de la carpeta **Scripts** en window

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19045.2251]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\marcela\Desktop\CAC2022doCuatri\python>virtualenv proyecto
created virtual environment CPython3.11.0.final.0-64 in 7114ms
creator CPython3Windows(dest=C:\Users\marcela\Desktop\CAC2022doCuatri\python\proyecto, global=False)
seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, appdata=(C:\Users\marcela\AppData\Local\pip\cache\virtualenv))
added seed packages: pip==22.3.1, setuptools==65.5.1, wheel==0.38.4
activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

C:\Users\marcela\Desktop\CAC2022doCuatri\python>cd proyecto

C:\Users\marcela\Desktop\CAC2022doCuatri\python\proyecto>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 9E63-AB4B

Directorio de C:\Users\marcela\Desktop\CAC2022doCuatri\python\proyecto

28/11/2022  22:20    <DIR>          .
28/11/2022  22:20    <DIR>          ..
28/11/2022  22:20                42 .gitignore
28/11/2022  22:20    <DIR>          Lib
28/11/2022  22:21        418 pyenv.cfg
28/11/2022  22:21    <DIR>          Scripts
                2 archivos          460 bytes
                4 dirs 12.497.035.264 bytes libres

C:\Users\marcela\Desktop\CAC2022doCuatri\python\proyecto>cd Scripts
El sistema no puede encontrar la ruta especificada.

C:\Users\marcela\Desktop\CAC2022doCuatri\python\proyecto>cd Scripts

C:\Users\marcela\Desktop\CAC2022doCuatri\python\proyecto\Scripts>activate

(proyecto) C:\Users\marcela\Desktop\CAC2022doCuatri\python\proyecto\Scripts>_
```

luego de activar el entorno de trabajo nos lo indica en el recuadro celeste **(proyecto)**

Arranquemos con el BackEnd del proyecto

Desde VSC abrimos la carpeta que acabamos de crear “proyecto”

Vamos a instalar Flask

Ir al menú de VSC, abrir la terminal e instalar según el sistema operativo:

- para windows: `pip install flask`
- para mac: `pip3 install flask`
- para linux `sudo apt install python3-pip`

Instalar en la terminal (según el sistema operativo):

Flask es un framework que nos ayuda a ser más productivos con python y bases de datos

para más información : <https://flask-es.readthedocs.io/>

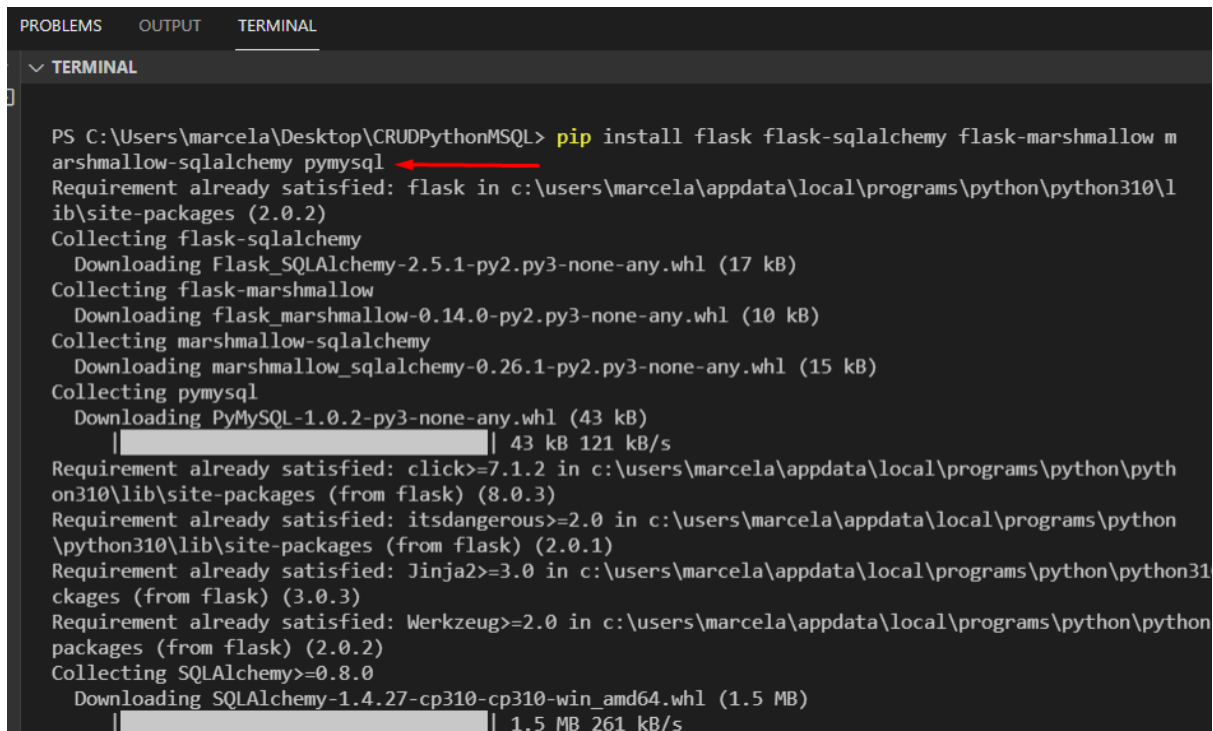
Marshmallow es una biblioteca independiente de ORM/ODM/framework para convertir tipos de datos complejos, como objetos, hacia y desde tipos de datos nativos de Python.

para más información <https://marshmallow.readthedocs.io/en/stable/>

SQLAlchemy 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

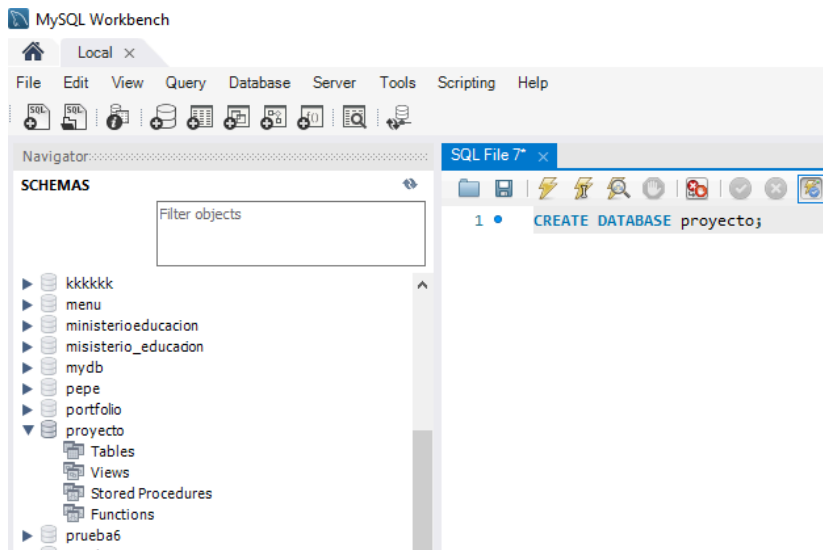
para más información: <https://flask-es.readthedocs.io/patterns/sqlalchemy/>

Cors me va a permitir acceder desde el frontend al Json que genera el backend



```
PROBLEMS  OUTPUT  TERMINAL
▼ TERMINAL
PS C:\Users\marcela\Desktop\CRUDPythonMSQL> pip install flask flask-sqlalchemy flask-marshmallow m
marshmallow-sqlalchemy pymysql
Requirement already satisfied: flask in c:\users\marcela\appdata\local\programs\python\python310\l
ib\site-packages (2.0.2)
Collecting flask-sqlalchemy
  Downloading Flask_SQLAlchemy-2.5.1-py2.py3-none-any.whl (17 kB)
Collecting flask-marshmallow
  Downloading flask_marshmallow-0.14.0-py2.py3-none-any.whl (10 kB)
Collecting marshmallow-sqlalchemy
  Downloading marshmallow_sqlalchemy-0.26.1-py2.py3-none-any.whl (15 kB)
Collecting pymysql
  Downloading PyMySQL-1.0.2-py3-none-any.whl (43 kB)
  | 43 kB 121 kB/s
Requirement already satisfied: click>=7.1.2 in c:\users\marcela\appdata\local\programs\python\pyth
on310\lib\site-packages (from flask) (8.0.3)
Requirement already satisfied: itsdangerous>=2.0 in c:\users\marcela\appdata\local\programs\python
\python310\lib\site-packages (from flask) (2.0.1)
Requirement already satisfied: Jinja2>=3.0 in c:\users\marcela\appdata\local\programs\python\python31
ckages (from flask) (3.0.3)
Requirement already satisfied: Werkzeug>=2.0 in c:\users\marcela\appdata\local\programs\python\python
packages (from flask) (2.0.2)
Collecting SQLAlchemy>=0.8.0
  Downloading SQLAlchemy-1.4.27-cp310-cp310-win_amd64.whl (1.5 MB)
  | 1.5 MB 261 kB/s
```

-Crear una base de datos que se llame por ejemplo `proyecto`, refrescar y verificar que se haya creado



Volver al proyecto

Dentro de la carpeta “proyecto”, crear un archivo `app.py`

dentro del archivo `app.py` copiar lo siguiente:

```
from flask import Flask , jsonify , request
# del modulo flask importar la clase Flask y los métodos jsonify,request
from flask_cors import CORS          # del modulo flask_cors importar CORS
from flask_sqlalchemy import SQLAlchemy
from flask_marshmallow import Marshmallow
app=Flask(__name__) # crear el objeto app de la clase Flask
CORS(app) #modulo cors es para que me permita acceder desde el frontend al backend

# configuro la base de datos, con el nombre el usuario y la clave
app.config['SQLALCHEMY_DATABASE_URI']='mysql+pymysql://root:root@localhost/proyecto'
# URI de la BBDD                                driver de la BD   user:clave@URLBBDD/nombreBBDD
'''
conn = pymysql.connect( host='projectocac.mysql.pythonanywhere-services.com', user='projectocac',
password='codocodo20', db='projectocac$default' )
'''

app.config['SQLALCHEMY_TRACK_MODIFICATIONS']=False #none
db= SQLAlchemy(app)      #crea el objeto db de la clase SQLAlchemy
ma=Marshmallow(app)      #crea el objeto ma de de la clase Marshmallow

# defino la tabla
class Producto(db.Model):    # la clase Producto hereda de db.Model
```

```

id=db.Column(db.Integer, primary_key=True)    #define los campos de la tabla
nombre=db.Column(db.String(100))
precio=db.Column(db.Integer)
stock=db.Column(db.Integer)
imagen=db.Column(db.String(400))

def __init__(self,nombre,precio,stock,imagen):    #crea el constructor de la
clase
    self.nombre=nombre    # no hace falta el id porque lo crea sola mysql por
ser auto_incremento
    self.precio=precio
    self.stock=stock
    self.imagen=imagen

# si hay que crear mas tablas , se hace aqui

with app.app_context():
    db.create_all()    # aqui crea todas las tablas
# *****
class ProductoSchema(ma.Schema):
    class Meta:
        fields=('id','nombre','precio','stock','imagen')

producto_schema=ProductoSchema()    # El objeto producto_schema es para
traer un producto
productos_schema=ProductoSchema(many=True)    # El objeto productos_schema es para
traer multiples registros de producto

# crea los endpoint o rutas (json)
@app.route('/productos',methods=['GET'])
def get_Productos():
    all_productos=Producto.query.all()    # el metodo query.all() lo hereda de
db.Model
    result=productos_schema.dump(all_productos)    # el metodo dump() lo hereda de
ma.schema y
    # trae todos los registros de la
tabla
    return jsonify(result)    # retorna un JSON de todos los
registros de la tabla

```

```
@app.route('/productos/<id>', methods=['GET'])
def get_producto(id):
    producto=Producto.query.get(id)
    return producto_schema jsonify(producto)    # retorna el JSON de un producto
recibido como parametro

@app.route('/productos/<id>', methods=['DELETE'])
def delete_producto(id):
    producto=Producto.query.get(id)
    db.session.delete(producto)
    db.session.commit()
    return producto_schema jsonify(producto)    # me devuelve un json con el
registro eliminado

@app.route('/productos', methods=['POST']) # crea ruta o endpoint
def create_producto():
    #print(request.json) # request.json contiene el json que envio el cliente
    nombre=request.json['nombre']
    precio=request.json['precio']
    stock=request.json['stock']
    imagen=request.json['imagen']
    new_producto=Producto(nombre,precio,stock,imagen)
    db.session.add(new_producto)
    db.session.commit()
    return producto_schema jsonify(new_producto)

@app.route('/productos/<id>' ,methods=['PUT'])
def update_producto(id):
    producto=Producto.query.get(id)

    producto.nombre=request.json['nombre']
    producto.precio=request.json['precio']
    producto.stock=request.json['stock']
    producto.imagen=request.json['imagen']

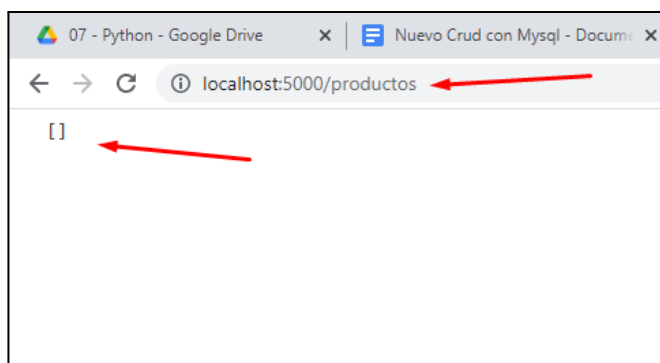
    db.session.commit()
    return producto_schema jsonify(producto)
```

```
@app.route('/')
def hello_world():
    return 'Hello from Flask!'

# programa principal *****
if __name__ == '__main__':
    app.run(debug=True, port=5000) # ejecuta el servidor Flask en el puerto 5000
```

y correr en la terminal: `python app.py`

probar en el navegador <http://localhost:5000/productos>, si devuelve [] es que esta' correcto, es el json que contiene una lista vacía donde trajo los datos la tabla productos de la base de datos, que como la tabla fue recién creada, está vacío



cargar en la base de datos 2 o 3 productos y volver a probar en el navegador <http://localhost:5000/productos> y tiene que mostrar el json con los datos que acaban de crear

Probar en el navegador <http://localhost:5000/productos/1>, y devuelve un Json con el registro con id=1.

Probar en Visual Studio en la extensión Thunder Client o en el programa de escritorio Postman los métodos GET, DELETE, POST y PUT

Video explicativo de Backend MVC: <https://youtu.be/3kvbD4p5GUE>

GitHub BackEnd MVC: https://github.com/MarcelaCerde/backend_MVC.git

Video explicativo de como subir el backend al servidor de PythonAnywhere: <https://youtu.be/sHsM7q2eRII>

Tutorial de FrontEnd del CRUD en el siguiente enlace:

<https://docs.google.com/document/d/1xSCipES1yrGqfKhMzJCzRy4aJLw7Vw-tNJPk5-tKEAQ/edit?usp=sharing>