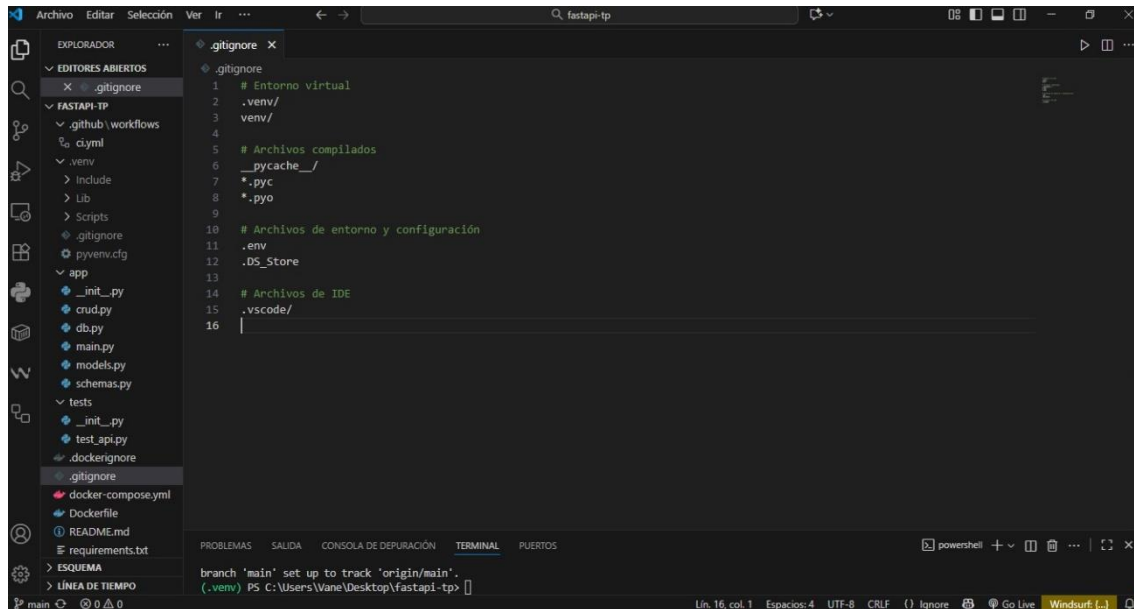


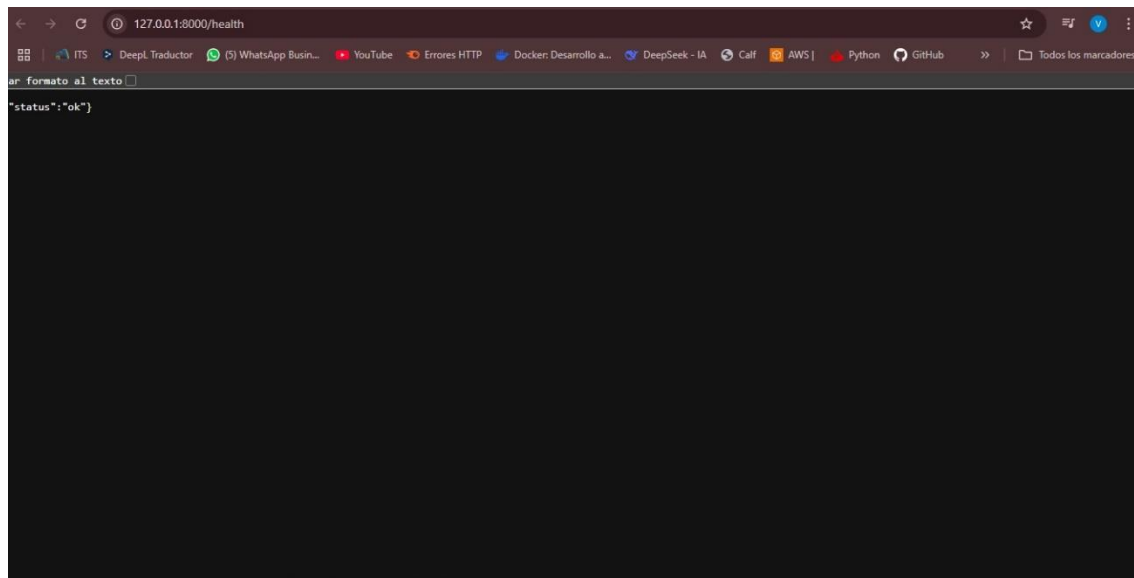
# Trabajo Final Obligatorio

## Evidencia de ejecución

- Estructura inicial del proyecto

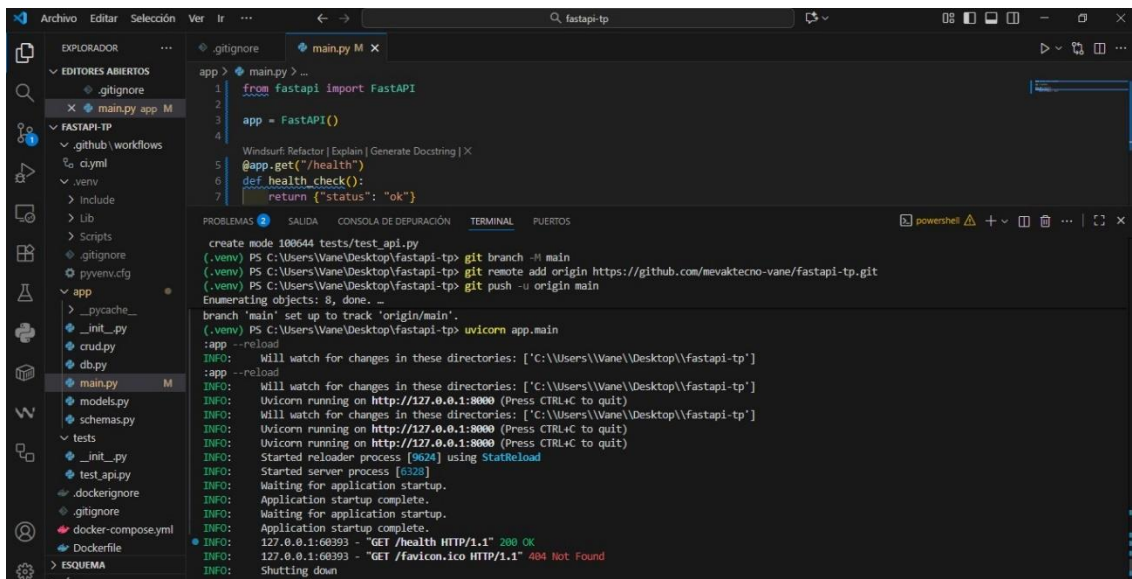


- La API corriendo en <http://localhost:8000/health>



# Trabajo Final Obligatorio

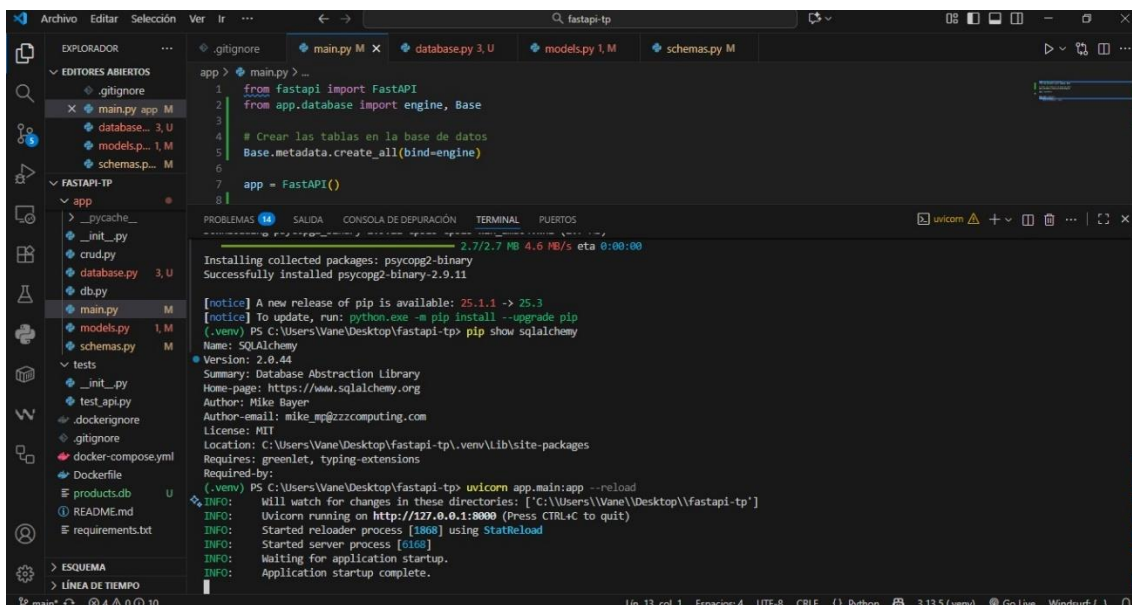
Y en la terminal:



```
app > main.py > ...
1 from fastapi import FastAPI
2
3 app = FastAPI()
4
5 @app.get("/health")
6 def health_check():
7     return {"status": "ok"}
```

```
create mode 100644 tests/test_api.py
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> git branch -M main
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> git remote add origin https://github.com/mevakteco-vane/fastapi-tp.git
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> git push -u origin main
Enumerating objects: 8, done.
branch 'main' set up to track 'origin/main'.
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> uvicorn app:main
app --reload
INFO: Will watch for changes in these directories: ['C:\Users\Vane\Desktop\fastapi-tp']
app --reload
INFO: Will watch for changes in these directories: ['C:\Users\Vane\Desktop\fastapi-tp']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Will watch for changes in these directories: ['C:\Users\Vane\Desktop\fastapi-tp']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [9624] using StatReload
INFO: Started server process [6328]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:68393 - "GET /health HTTP/1.1" 200 OK
INFO: 127.0.0.1:68393 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO: Shutting down
```

- Se crea database y el archivo products.db



```
1 from fastapi import FastAPI
2 from app.database import engine, Base
3
4 # Crear las tablas en la base de datos
5 Base.metadata.create_all(bind=engine)
6
7 app = FastAPI()
8
```

```
Installing collected packages: psycopg2-binary
Successfully installed psycopg2-binary-2.9.11
[notice] A new release of pip is available: 25.1.1 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> pip show sqlalchemy
Name: SQLAlchemy
Version: 2.0.44
Summary: Database Abstraction Library
Home-page: https://www.sqlalchemy.org
Author: Mike Bayer
Author-email: mike_mp@zzzcomputing.com
License: MIT
Locations: C:\Users\Vane\Desktop\fastapi-tp\.venv\Lib\site-packages
Requires: greenlet, typing-extensions
Required-by:
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> uvicorn app:main --reload
INFO: Will watch for changes in these directories: ['C:\Users\Vane\Desktop\fastapi-tp']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [1808] using StatReload
INFO: Started server process [6168]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

- Endpoints CRUD de Productos



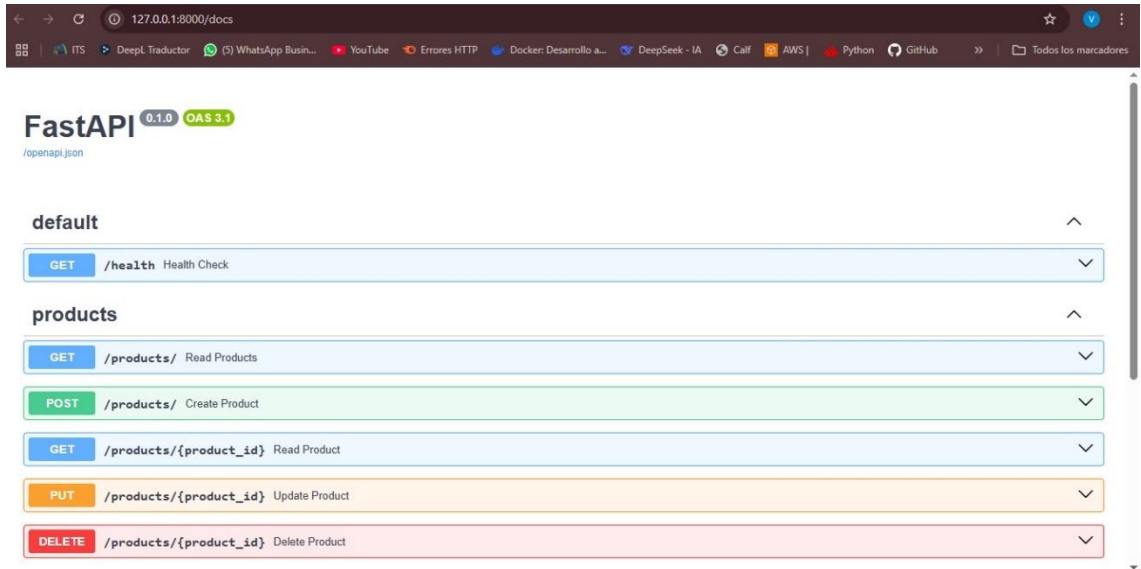
```
> ...pycache_
> routes
> products.py 5, U
> _init_.py
> crud.py 1
> database.py 3
> db.py
> main.py
> models.py 1
> ESQUEMA
> LÍNEA DE TIEMPO
```

```
app --reload
INFO: Will watch for changes in these directories: ['C:\Users\Vane\Desktop\fastapi-tp']
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO: Started reloader process [2552] using StatReload
INFO: Started server process [3690]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:56859 - "GET /docs HTTP/1.1" 200 OK
INFO: 127.0.0.1:56859 - "GET /openapi.json HTTP/1.1" 200 OK
```

# Trabajo Final Obligatorio

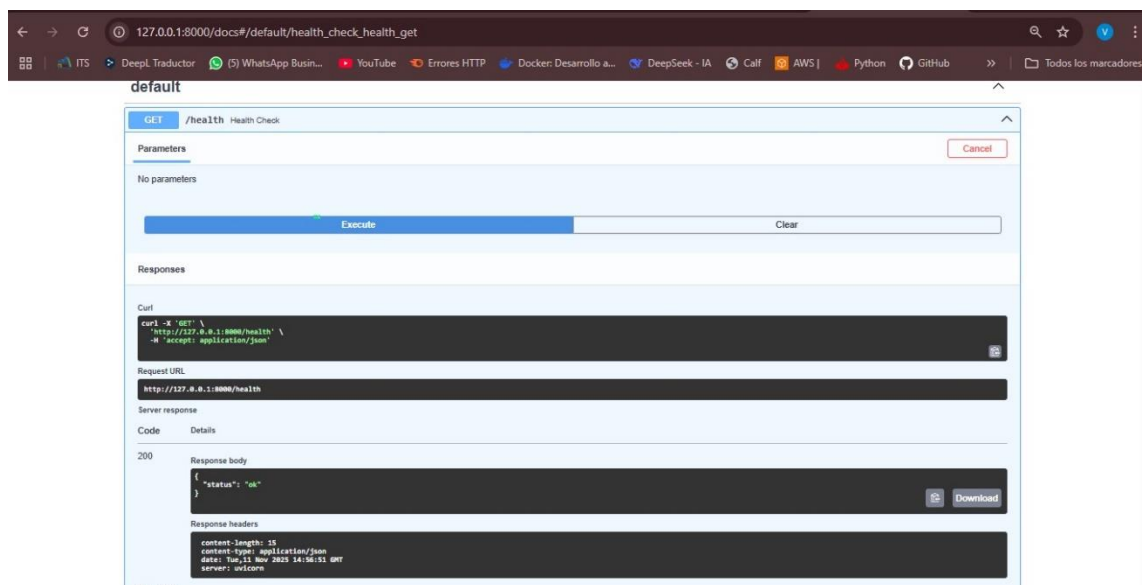
- Registro de las rutas creadas CRUD en app/main.py

En <http://127.0.0.1:8000/docs>



Pruebas del CRUD de productos: la API funciona correctamente desde el navegador o Swagger UI. <http://127.0.0.1:8000/docs>

- GET /health — Health Check: devuelve un JSON simple que confirma que la API está viva



- POST /products/ — *Crear producto*: crea un nuevo registro en la base de datos.

# Trabajo Final Obligatorio

Resultado de post:

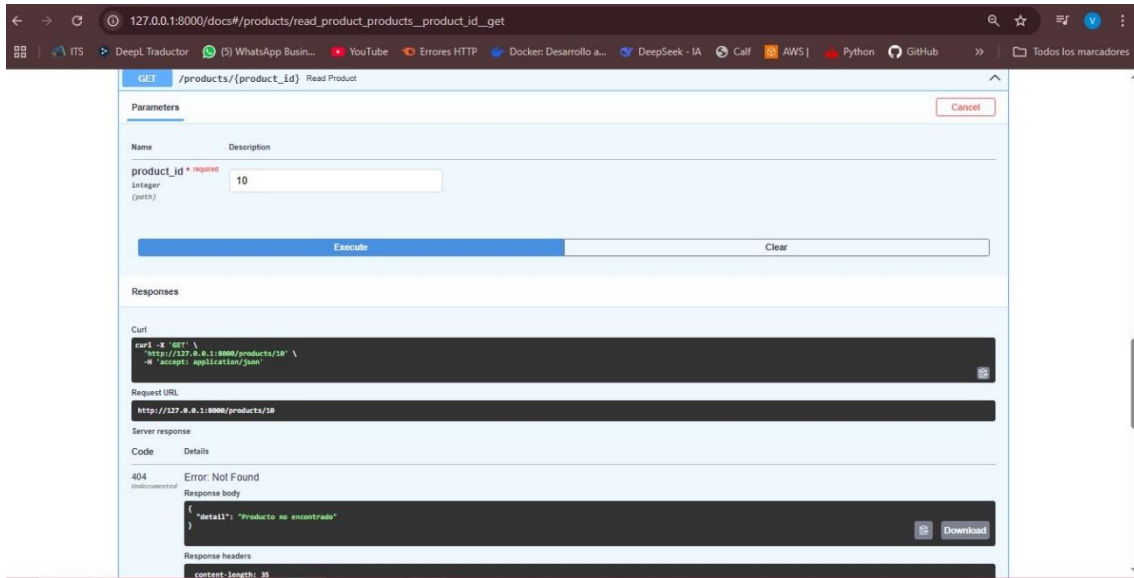
The screenshot shows a REST client interface with a browser address bar at 127.0.0.1:8000/docs#. The interface includes an 'Execute' button and a 'Clear' button. Below these, the 'Responses' section displays the results of a POST request. The 'Curl' section shows the command: `curl -X 'POST' \ 'http://127.0.0.1:8000/products/' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d '{ "name": "Tachado", "description": "Retrolimado", "price": 2000 }'`. The 'Request URL' is `http://127.0.0.1:8000/products/`. The 'Server response' section shows a 200 status code. The 'Response body' is a JSON object: `{ "name": "Tachado", "description": "Retrolimado", "price": 1000, "id": 4 }`. The 'Response headers' show: `content-length: 72, content-type: application/json, date: Tue, 11 Nov 2025 15:01:07 GMT, server: uvicorn`. At the bottom, a table lists the response with a 200 status code and a 'Successful Response' description.

- GET /products/{product\_id} — Leer producto por id: devuelve el producto con el id dado. (en este caso el id 4, el producto creado antes)

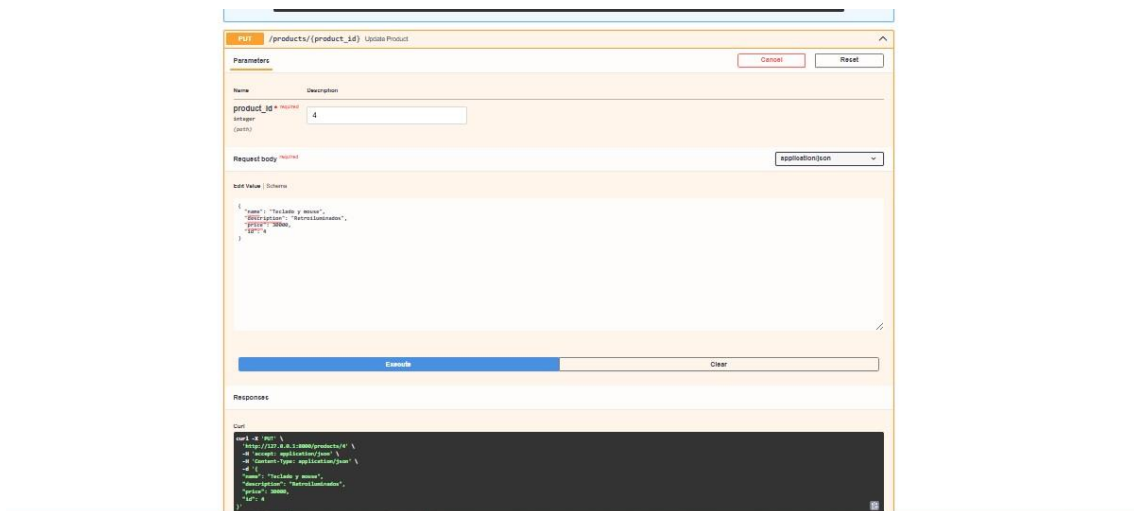
The screenshot shows a REST client interface with a browser address bar at 127.0.0.1:8000/docs#/products/read\_product\_products\_product\_id\_get. The interface includes a 'Cancel' button. Below this, the 'Parameters' section shows a 'product\_id' parameter with a value of 4. The 'Execute' button is visible. The 'Responses' section displays the results of a GET request. The 'Curl' section shows the command: `curl -X 'GET' \ 'http://127.0.0.1:8000/products/4' \ -H 'accept: application/json'`. The 'Request URL' is `http://127.0.0.1:8000/products/4`. The 'Server response' section shows a 200 status code. The 'Response body' is a JSON object: `{ "name": "Tachado", "description": "Retrolimado", "price": 1000, "id": 4 }`. The 'Response headers' are visible at the bottom.

# Trabajo Final Obligatorio

Ahora coloco un id de un producto inexistente para que devuelva error (id10, error 404)

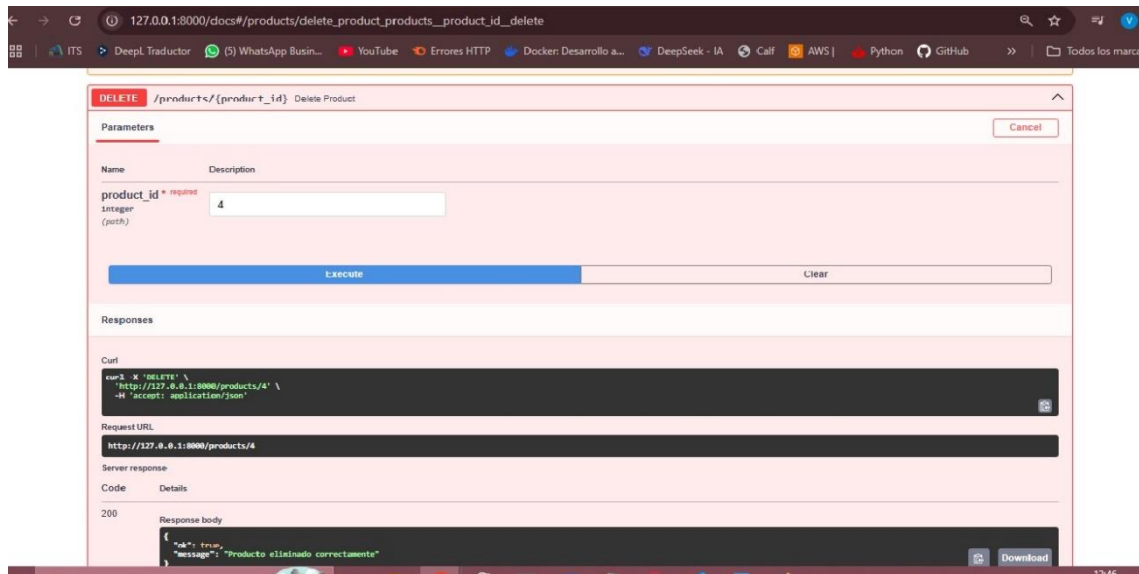


- PUT `/products/{product_id}` — Actualizar producto: actualiza los campos del producto indicado por `product_id`.



# Trabajo Final Obligatorio

- DELETE /products/{product\_id} — *Eliminar producto*: borra el producto con ese id



- Post desde terminal:

```
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> Invoke-RestMethod -Uri "http://127.0.0.1:8000/products/" -Method POST -ContentType "application/json" -Body '{"name":"Mouse","price":1200,"description":"Ergonomico"}'

name description price id
-----
Mouse Ergonomico 1200,0 5
```

- Get desde terminal: muestra los distintos productos creados como prueba desde el navegador o Swagger UI y los creados desde terminal

```
name description price id
-----
Teclado Retroiluminado Teclado retroiluminado RGB con switches rojos 25000,0 1
Teclado Retroiluminado Teclado retroiluminado RGB con switches rojos 25000,0 2
string string 0,0 3
Mouse Retroiluminado 10000,0 4
Mouse Ergonomico 1200,0 5

(.venv) PS C:\Users\Vane\Desktop\fastapi-tp>
```

- PUT desde terminal:

```
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> Invoke-RestMethod -Uri "http://127.0.0.1:8000/products/1" -Method PUT -ContentType "application/json" -Body '{"name":"Mouse Ergonomico","price":1500,"description":"Version mejorada con sensor optico"}'

name description price id
-----
Mouse Ergonomico Version mejorada con sensor optico 1500,0 1
```

- DELETE desde terminal:

```
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> Invoke-RestMethod -Uri "http://127.0.0.1:8000/products/5" -Method DELETE

ok message
--
True Producto eliminado correctamente
```

# Trabajo Final Obligatorio

- Creo y pruebo tests

```

> .pytest_cache
> .venv
> app
  tests
    > _pycache_
    > _init_py
    > test_api.py
    > .dockerignore

platform win32 -- Python 3.13.5, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\Vane\Desktop\Fastapi-tp\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Vane\Desktop\Fastapi-tp
plugins: anyio-4.11.0
collected 3 items

tests/test_api.py::test_health PASSED
tests/test_api.py::test_create_product PASSED
tests/test_api.py::test_get_products PASSED

```

- Creación del contenedor

The screenshot displays the Docker Desktop application window. The top bar shows standard macOS navigation icons and the title "Archivo Editar Selección Ver Ir ...". Below the title bar, there are tabs for "Archivo", "Editar", "Selección", "Ver", and "Ir". The main area is divided into several sections:

- Dockerfile**: A tab labeled ".dockerignore X" is active, showing a file named ".dockerignore" with its contents.
- PROBLEMAS**, **SALIDA**, **CONSOLA DE DEPURACIÓN**, **TERMINAL**, and **PUERTOS**: These are tabs at the bottom of the main area. The **TERMINAL** tab is currently selected, displaying the output of a Docker build command.

The terminal output shows the following steps:

```
[+] Building 191.1s (13/13) FINISHED
-> transferring dockerfile: 821B
-> [internal] load metadata for docker.io/library/python:3.11-slim
-> [auth] library/python:pull token for registry-1.docker.io
-> [internal] load .dockerignore
-> transferring context: 180B
-> [internal] load build context
-> transferring context: 18.12kB
-> [builder 1/5] FROM docker.io/library/python:3.11-slim@sha256:e4676722fba839e2e5cd
-> resolve docker.io/library/python:3.11-slim@sha256:e4676722fba839e2e5cd&sha256:
-> sha256:6c8edbd1e6015ca1bf5fa7f5a1ba0d274f556971813dd047c95480b70bc_2488 /
-> sha256:f90d217b63fe8a47f8a66f20cfaf3aec4fc6d2a44869f05b6290ebafc22ab1 14.36M
-> sha256:1ee9c106547f05aa380c4dc2837c546349943d73d965a3fc40f228dc8e993 1.29MB
-> sha256:d7eced7702a5dbfd0f79a17edc34b534d08f3051980e2c948fba72db197fc 29.78M
-> extracting sha256:d7eced7702a5dbfd0f79a17edc34b534d08f3051980e2c948fba72db1
-> extracting sha256:1ee9c106547f05aa380c4dc2837c546349943d73d965a3fc40f228dc8b
-> extracting sha256:f90d217b63fe8a47f8a66f20cfaf3aec4fc6d2a44869f05b6290ebafc2
-> extracting sha256:6082d1b63fe8a47f8a66f20cfaf3aec4fc6d2a44869f05b6290ebafc2
-> exporting sha256:6082d1b63fe8a47f8a66f20cfaf3aec4fc6d2a44869f05b6290ebafc2
-> [builder 2/5] WORKDIR /app
-> [builder 3/5] RUN apt-get update && apt-get install -y --no-install-recommends bu
-> [builder 4/5] COPY requirements.txt .
-> [stage-1 3/4] COPY --from=builder /usr/local /usr/local
-> [stage-1 4/4] COPY . .
-> exporting to image
-> exporting layers
-> exporting manifest sha256:d9cd50a1c190a2bf990f4fd6f270a0421d8ae341c5e2f1cf22f
-> exporting attestation manifest sha256:db6284238431ca29ee2751ce23f5512196c74a
-> exporting manifest list sha256:80fb4451f4161bedaba940d84781da062e3e0513a4f0045
-> naming to docker.io/library/fastapi-tp:latest
-> unpacking to docker.io/library/fastapi-tp:latest

View build details: docker-desktop://dashboards/build/docker-run-docker-linux/vn7n
(vnm) PS C:\Users\Vane\Desktop> fastapi-tp docker run -d -p 8000:8000 --name fastap
0812bf2a12af15b6f0c1d758e0757a35f3542cbf7cbd29707da7fadcc01800e5
```

- Luego de resolver problemas de nombre (tenía otro contenedor con el mismo nombre) y de puerto (estaba siendo usado por Portainer) queda el contenedor corriendo

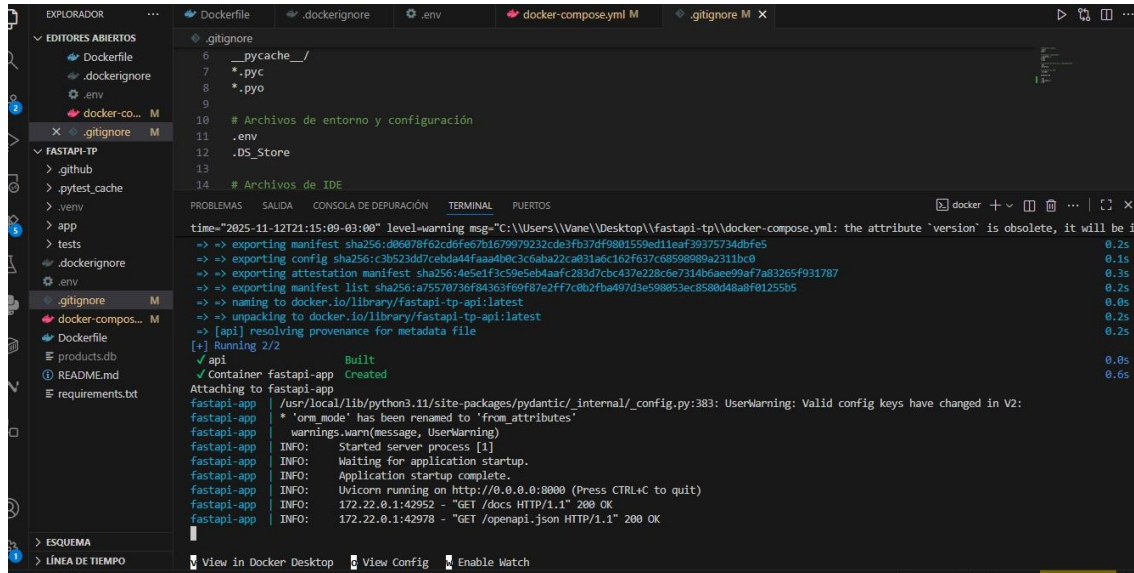
```
(.venv) PS C:\Users\Vane\Desktop>fastapi-tp> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
98685d956978	fastapi-tp:latest	"uvicorn app.main:ap..."	47 seconds ago	Up 46 seconds	0.0.0.0:8000->8000/tcp	fastapi-app



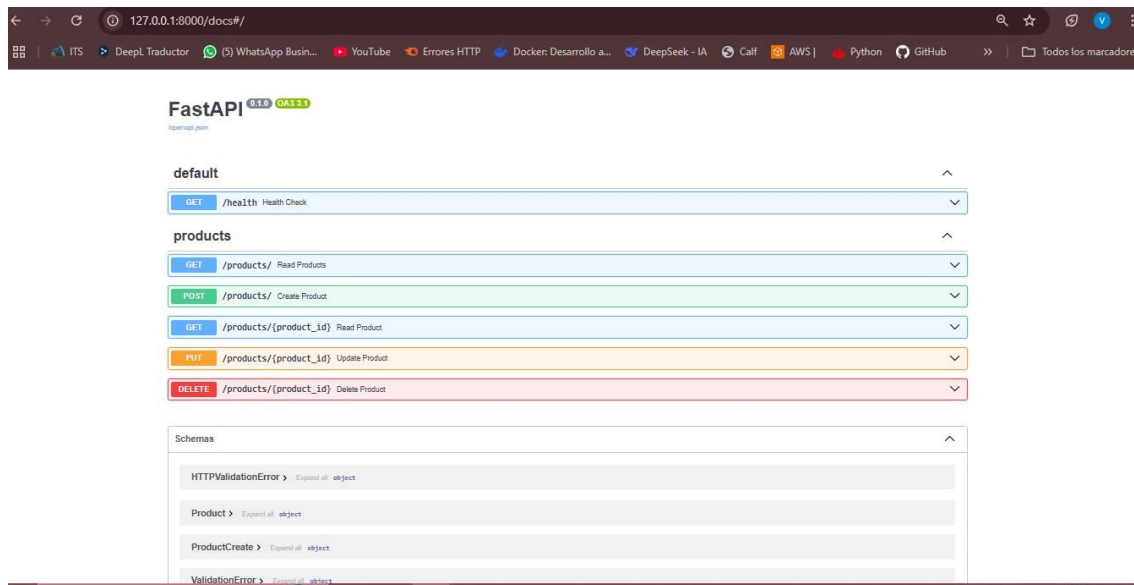
# Trabajo Final Obligatorio

- Dockercompose.yml



The screenshot shows a VS Code editor with a project named 'FASTAPI-TP'. The file explorer on the left shows files like 'Dockerfile', '.dockerignore', '.env', 'docker-compose.yml', and '.gitignore'. The main editor displays the content of '.gitignore', which includes entries for '\_\_pycache\_\_/', '\*.pyc', '\*.pyo', and various IDE and project-specific files. The terminal at the bottom shows the output of running 'docker-compose up', including warnings about the 'version' attribute in the Docker Compose file and the successful creation and startup of the 'fastapi-app' container. The container logs show a warning about 'orm\_mode' being renamed and the application starting successfully on port 8000.

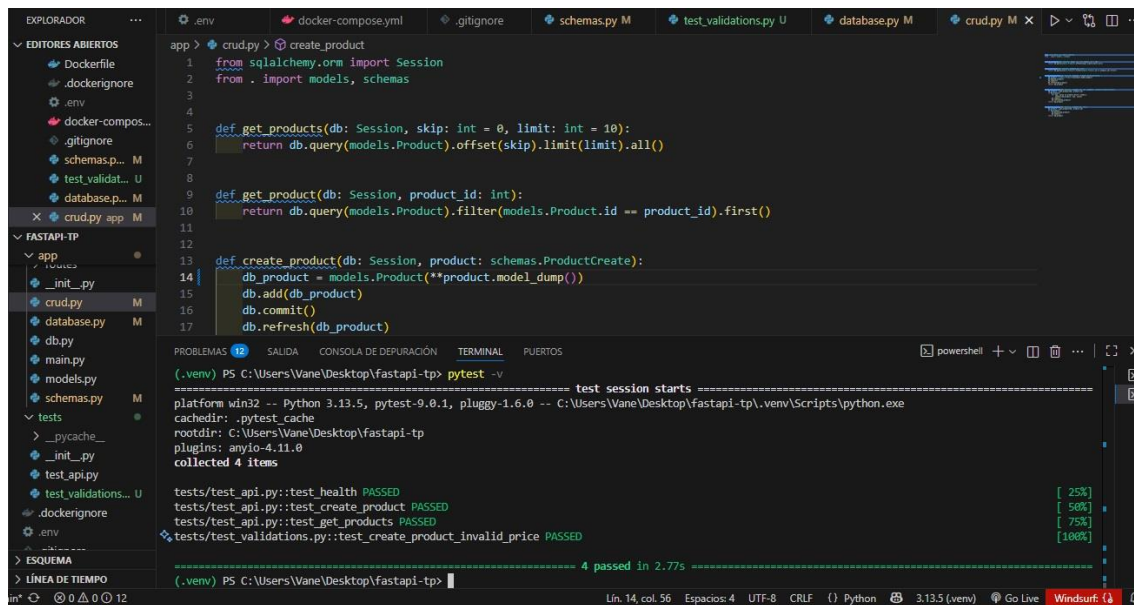
- <http://127.0.0.1:8000/docs> funcionando





# Trabajo Final Obligatorio

- Agrego validaciones, manejo de errores claros y pruebas automáticas con pytest:  
Validaciones en schemas.py, Creo un nuevo archivo de test:  
tests/test\_validations.py. Luego hice modificaciones en app/database.py y en  
app/crud.py por warnings que se veían a la hora de ejecutar los tests:



```
app > crud.py > create_product
1 from sqlalchemy.orm import Session
2 from . import models, schemas
3
4
5 def get_products(db: Session, skip: int = 0, limit: int = 10):
6     return db.query(models.Product).offset(skip).limit(limit).all()
7
8
9 def get_product(db: Session, product_id: int):
10    return db.query(models.Product).filter(models.Product.id == product_id).first()
11
12
13 def create_product(db: Session, product: schemas.ProductCreate):
14    db_product = models.Product(**product.model_dump())
15    db.add(db_product)
16    db.commit()
17    db.refresh(db_product)
```

```
PROBLEMAS 12 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp> pytest -v
===== test session starts =====
platform win32 -- Python 3.13.5, pytest-9.0.1, pluggy-1.6.0 -- C:\Users\Vane\Desktop\fastapi-tp\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Vane\Desktop\fastapi-tp
plugins: anyio-4.11.0
collected 4 items

tests/test_api.py::test_health PASSED [ 25%]
tests/test_api.py::test_create_product PASSED [ 50%]
tests/test_api.py::test_get_products PASSED [ 75%]
tests/test_validations.py::test_create_product_invalid_price PASSED [100%]

===== 4 passed in 2.77s =====
(.venv) PS C:\Users\Vane\Desktop\fastapi-tp>
```

## Conclusión

El desarrollo de esta API sirvió para integrar conceptos fundamentales de FastAPI, manejo de base de datos con SQLAlchemy, validaciones con Pydantic y ejecución de pruebas automáticas con pytest.

Además, se trabajó con contenedores Docker para asegurar un entorno de ejecución consistente y portable, y se incorporó un flujo de Integración Continua con GitHub Actions.

El resultado final es una API funcional, testeada, contenedorizada y lista para desplegarse.

El repositorio incluye todo el código fuente, el archivo docker-compose.yml, el Dockerfile, los tests, y este documento con las capturas de los pasos más importantes del proceso.

## Integración Continua:

Los tests automáticos se ejecutan en cada commit mediante GitHub Actions.

Link: <https://github.com/mevaktecno-vane/fastapi-tp/actions>