

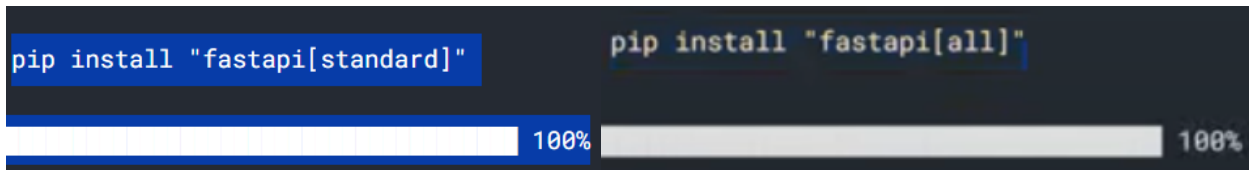
# PROYECTO REST API

## ARQUITECTURA:

BASE DE DATOS	[MYSQL]
BACKEND	[FASTAPI]
FRONTEND	[REACT]

### 1. Instalación de ambiente fastapi

Por línea de comandos es posible utilizar cualquiera de las dos siguientes líneas de código:



Cabe aclarar que en la documentación de fastapi aclara que la primera sentencia es la instrucción que genera la instalación más corta, es decir el número de dependencias es menor, debido a que se han reportado fallos con algunas dependencias. Igual en este instructivo se intentó con la segunda instrucción que genera la instalación con mayor número de dependencias<sup>1</sup>.

**Technical Details**

Before this, `fastapi` would include the standard dependencies, with Uvicorn and the `fastapi-cli`, etc.

And `fastapi-slim` would not include those standard dependencies.

Now `fastapi` doesn't include those standard dependencies unless you install with `pip install "fastapi[standard]"`.

Before, you would install `pip install fastapi`, now you should include the `standard` optional dependencies (unless you want to exclude one of those): `pip install "fastapi[standard]"`.

This change is because having the standard optional dependencies installed by default was being inconvenient to several users, and having to install instead `fastapi-slim` was not being a feasible solution.

**Upgrades**

- Remove `orjson` and `ujson` from default dependencies. PR #11842 by @tiangolo.
- These dependencies are still installed when you install with `pip install "fastapi[all]"`. But they are not included in `pip install fastapi`.

El resultado de la instalación corresponde a la siguiente imagen<sup>2</sup>.

---

<sup>1</sup> La documentación a la cual hago referencia puede consultarse en [https://fastapi.tiangolo.com/release-notes/#inter-nal\\_13](https://fastapi.tiangolo.com/release-notes/#inter-nal_13).

<sup>2</sup> Para mayor precisión o referencia se cuenta con el log de instalación en el archivo incluido llamado «InstallFastapi-Log.txt».

```
Administrador: Node.js command prompt
----- 58.3/58.3 kB 3.0 MB/s eta 0:00:00
Downloading httptools-0.6.4-cp312-cp312-win_amd64.whl (88 kB)
----- 88.6/88.6 kB ? eta 0:00:00
Downloading idna-3.10-py3-none-any.whl (70 kB)
----- 70.4/70.4 kB ? eta 0:00:00
Downloading MarkupSafe-3.0.2-cp312-cp312-win_amd64.whl (15 kB)
Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Downloading sniffio-1.3.1-py3-none-any.whl (10 kB)
Downloading typer-0.13.0-py3-none-any.whl (44 kB)
----- 44.2/44.2 kB 2.3 MB/s eta 0:00:00
Downloading watchfiles-0.24.0-cp312-cp312-win_amd64.whl (277 kB)
----- 277.1/277.1 kB ? eta 0:00:00
Downloading websockets-14.0-cp312-cp312-win_amd64.whl (162 kB)
----- 162.0/162.0 kB 9.5 MB/s eta 0:00:00
Downloading certifi-2024.8.30-py3-none-any.whl (167 kB)
----- 167.3/167.3 kB ? eta 0:00:00
Downloading rich-13.9.4-py3-none-any.whl (242 kB)
----- 242.4/242.4 kB ? eta 0:00:00
Downloading shellingham-1.5.4-py2.py3-none-any.whl (9.8 kB)
Downloading markdown_it_py-3.0.0-py3-none-any.whl (67 kB)
----- 67.3/67.3 kB 5.2 MB/s eta 0:00:00
Downloading pygments-2.18.0-py3-none-any.whl (1.2 MB)
----- 1.2/1.2 MB 37.4 MB/s eta 0:00:00
Downloading mdurl-0.1.2-py3-none-any.whl (10.0 kB)
Installing collected packages: websockets, ujson, typing-extensions, sniffio, shellingham, pyyaml, python-multipart, python-dotenv, pygments, orjson, mdurl, MarkupSafe, itsdangerous, idna, httptools, h11, dnspyt
hon, colorama, certifi, annotated-types, pydantic-core, markdown-it-py, Jinja2, httpcore, email-validator, click, anyio, watchfiles, uvicorn, starlette, rich, pydantic, httpx, typer, pydantic-settings, pydantic-
extra-types, fastapi, fastapi-cli
Successfully installed MarkupSafe-3.0.2 annotated-types-0.7.0 anyio-4.6.2.post1 certifi-2024.8.30 click-8.1.7 colorama-0.4.6 dnspython-2.7.0 email-validator-2.2.0 fastapi-0.115.4 fastapi-cli-0.0.5 h11-0.14.0 htt
pcore-1.0.6 httptools-0.6.4 httpx-0.27.2 idna-3.10 itsdangerous-2.2.0 Jinja2-3.1.4 markdown-it-py-3.0.0 mdurl-0.1.2 orjson-3.10.11 pydantic-2.9.2 pydantic-core-2.23.4 pydantic-extra-types-2.10.0 pydantic-setting
s-2.6.1 pygments-2.18.0 python-dotenv-1.0.1 python-multipart-0.0.17 pyyaml-6.0.2 rich-13.9.4 shellingham-1.5.4 sniffio-1.3.1 starlette-0.41.2 typer-0.13.0 typing-extensions-4.12.2 ujson-5.10.0 uvicorn-0.32.0 wat
chfiles-0.24.0 websockets-14.0

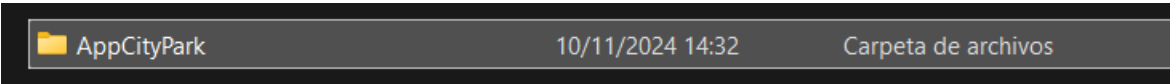
[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Windows\System32>python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\python312\lib\site-packages (24.0)
Collecting pip
  Downloading pip-24.3.1-py3-none-any.whl.metadata (3.7 kB)
    Downloading pip-24.3.1-py3-none-any.whl (1.8 MB)
      ----- 1.8/1.8 MB 14.5 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.0
    Uninstalling pip-24.0:
      Successfully uninstalled pip-24.0
Successfully installed pip-24.3.1

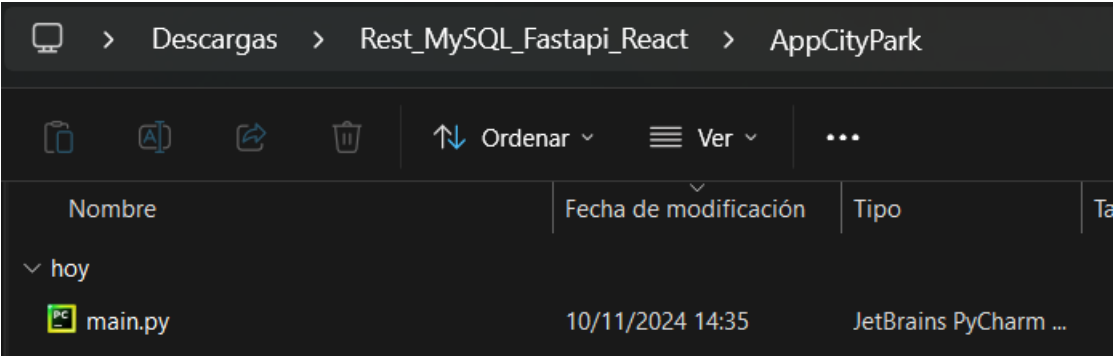
C:\Windows\System32>
```

2. Creación de la parte back del proyecto es decir, el proyecto tipo rest en fastAPI.

Se crea una carpeta para el proyecto.



Luego se crea un archivo con extensión «.py» dentro de la carpeta creada.



En una pantalla de línea de comandos, se realizan las acciones necesarias para ubicarse en la carpeta del proyecto y se ejecuta el comando «fastapi dev main.py»

```
Administrador: Node.js command prompt - fastapi dev main.py
C:\Users>cd usuario
C:\Users\Usuario>cd downloads
C:\Users\Usuario\Downloads>cd rest_mysql_fastapi_react
C:\Users\Usuario\Downloads\Rest_MySQL_Fastapi_React>cd appcitypark
C:\Users\Usuario\Downloads\Rest_MySQL_Fastapi_React\AppCityPark>fastapi dev main.py
INFO: Using path main.py
INFO: Resolved absolute path C:\Users\Usuario\Downloads\Rest_MySQL_Fastapi_React\AppCityPark\main.py
INFO: Searching for package file structure from directories with _init_.py files
INFO: Importing from C:\Users\Usuario\Downloads\Rest_MySQL_Fastapi_React\AppCityPark

Python module file
main.py

INFO: Importing module main
INFO: Found importable FastAPI app

Importable FastAPI app
from main import app

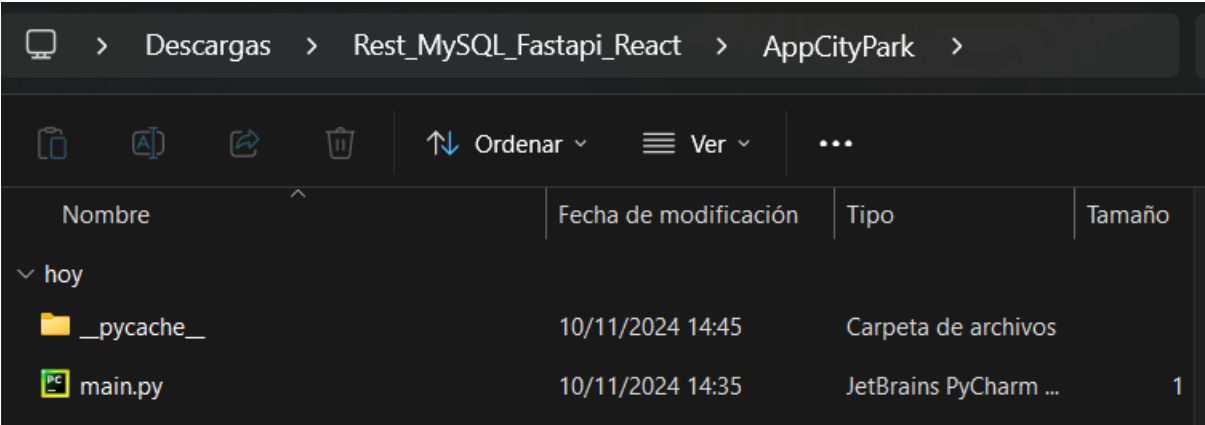
INFO: Using import string main:app

FastAPI CLI - Development mode
Serving at: http://127.0.0.1:8000
API docs: http://127.0.0.1:8000/docs
Running in development mode, for production use:
fastapi run

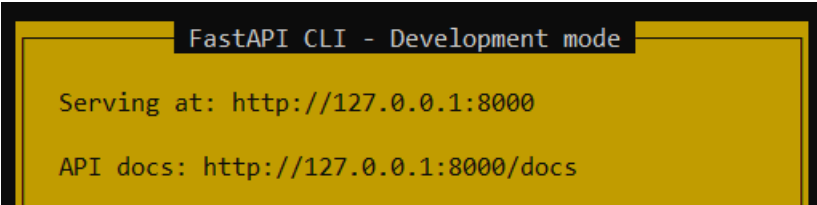
+ [32mINFO+ [0m: Will watch for changes in these directories: ['C:\Users\Usuario\Downloads\Rest_MySQL_Fastapi_React\AppCityPark']
+ [32mINFO+ [0m: Uvicorn running on +[1mhttp://127.0.0.1:8000+ [0m (Press CTRL+C to quit)
+ [32mINFO+ [0m: Started reload process [+ [36m+ [1m20360+ [0m] using [+ [36m+ [1mMatchFiles+ [0m]
+ [32mINFO+ [0m: Started server process [+ [36m+ [1m8900+ [0m]
+ [32mINFO+ [0m: Waiting for application startup.
+ [32mINFO+ [0m: Application startup complete.
```

La ejecución del mencionado comando ha generado 2 acciones:

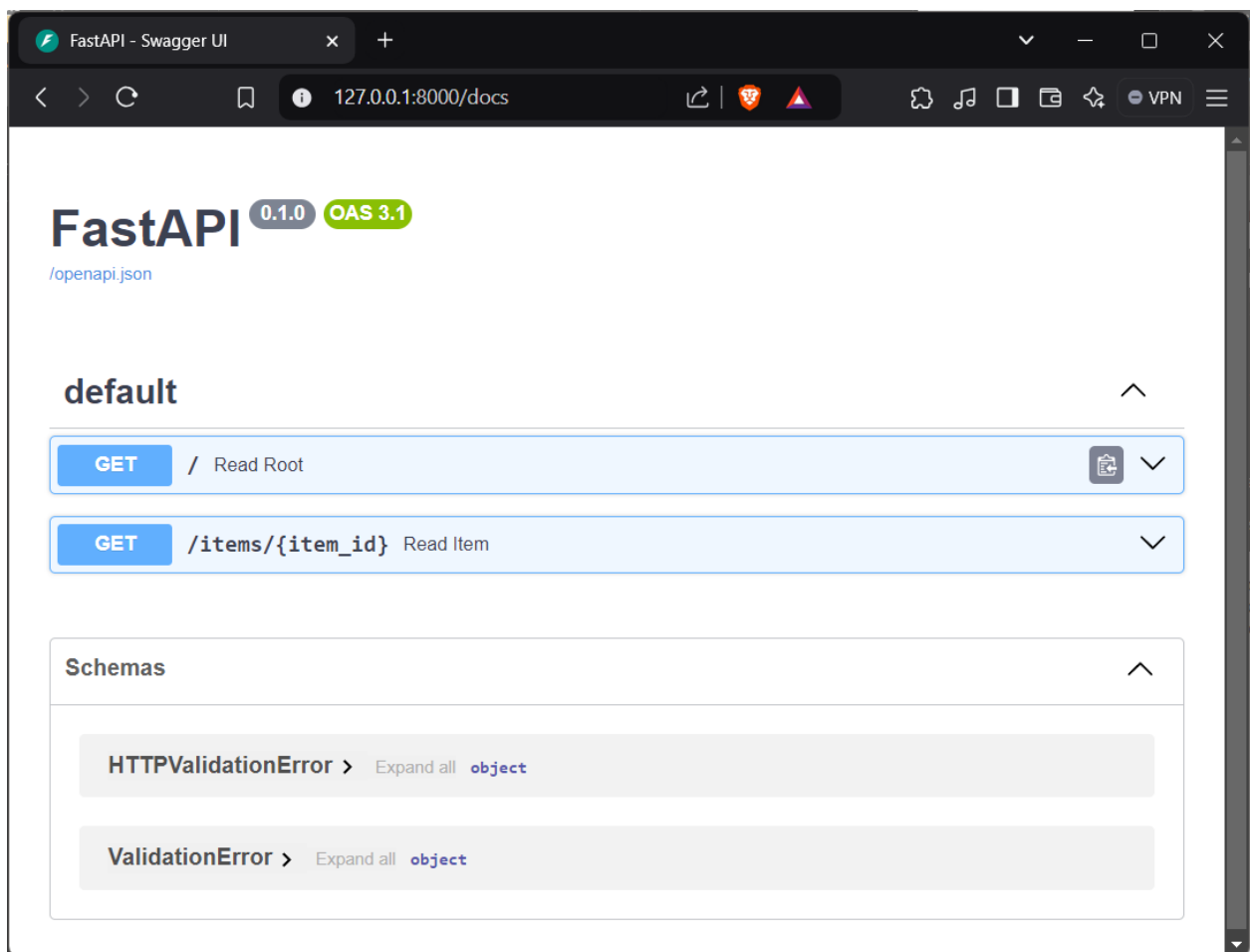
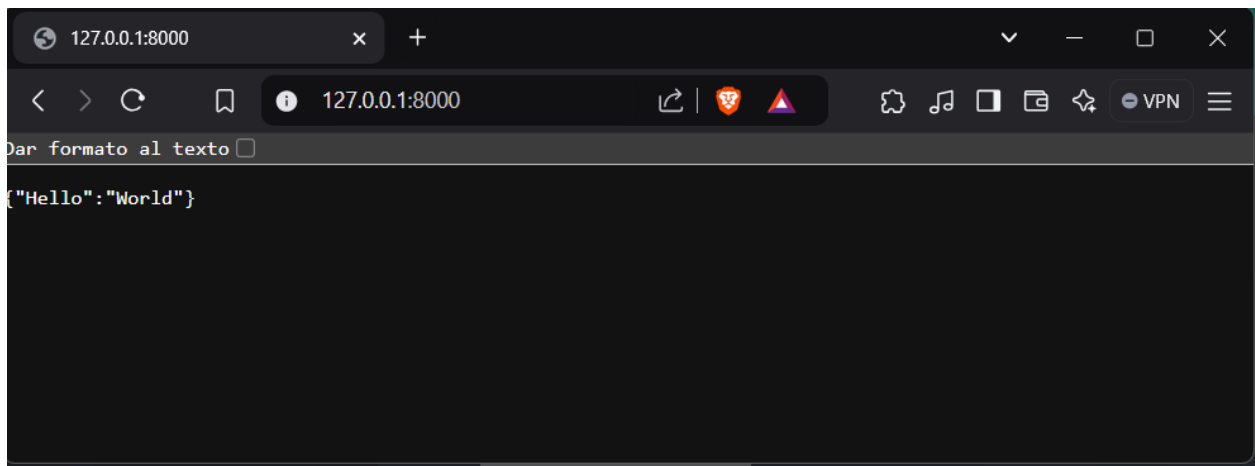
En primer lugar ha creado una carpeta con las dependencias necesarar para simular un servidor web.



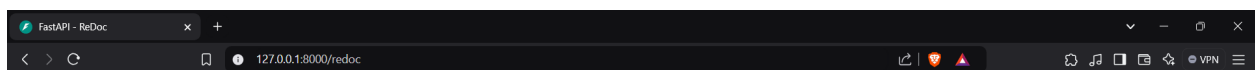
En segundo lugar ha configurado dos url's generales para exponer la aplicación backend (las cuales nos indica en la línea de comandos), y con las acciones que se realizarán más adelante, se tendrán otras url's que estarán expuestas para recibir peticiones desde cualquier aplicación frontend que cuente con la posibilidad de conocerlas.

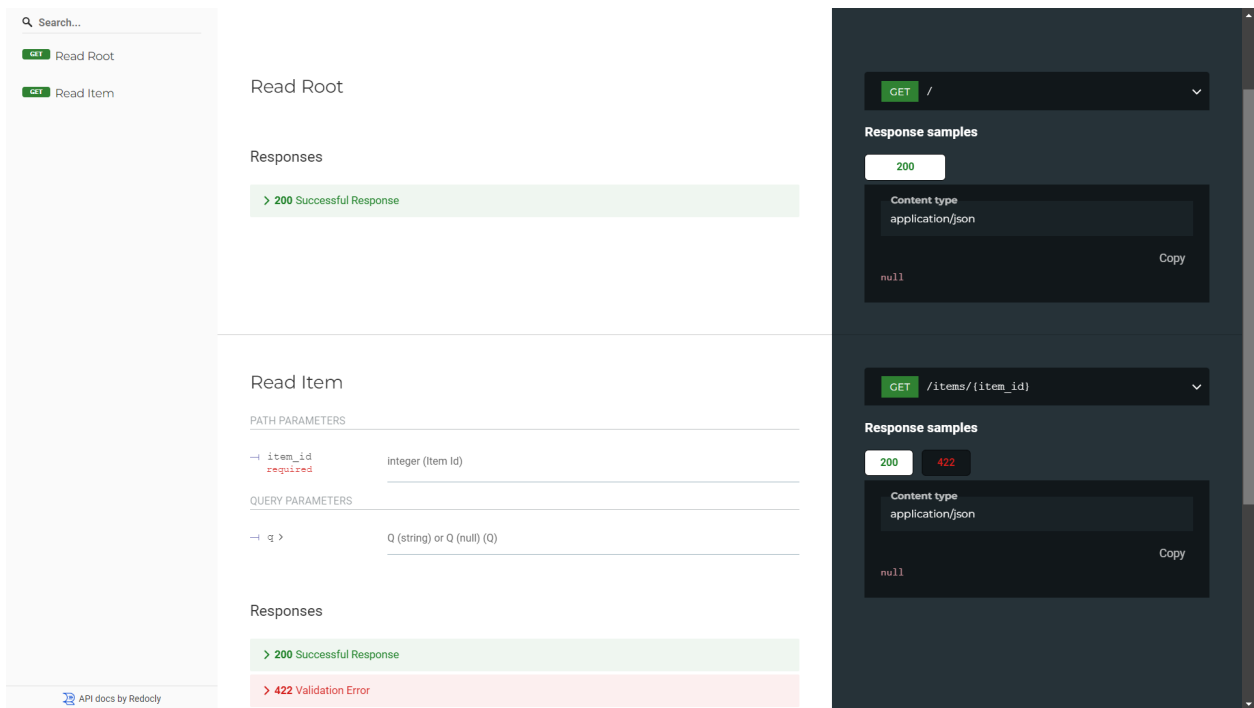


Al escribir estas url's en un navegador se observará:



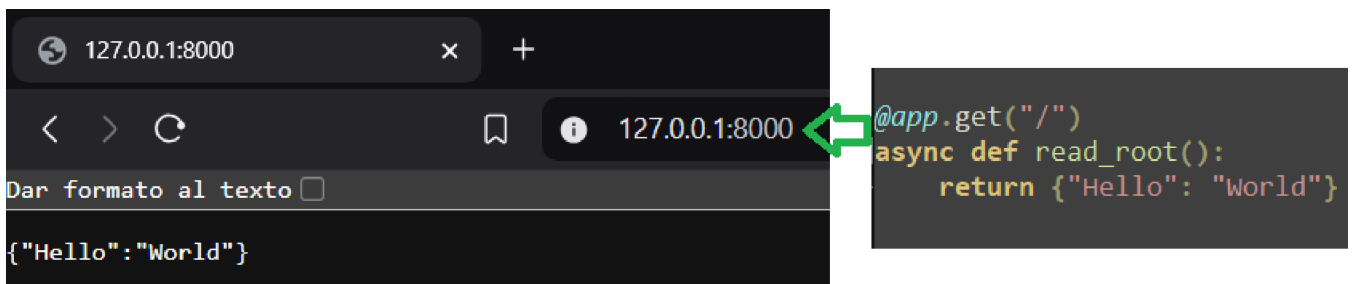
Es importante adicionar que en la documentación de fastapi, se documenta una tercera alternativa que no es referenciada en el proceso anterior pero se relaciona en este documento:



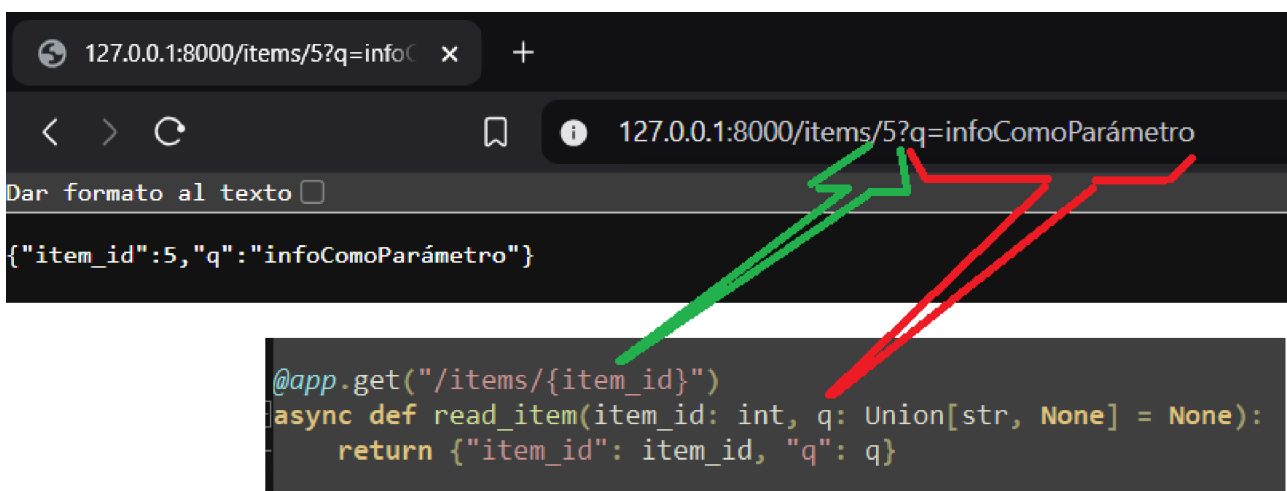


Para este punto se ha creado un entorno backend genérico que permite realizar peticiones de tipo GET, POST, PUT y DELETE, y por ahora la documentación de fastapi ha brindado el código necesario para tener 2 peticiones de tipo GET.

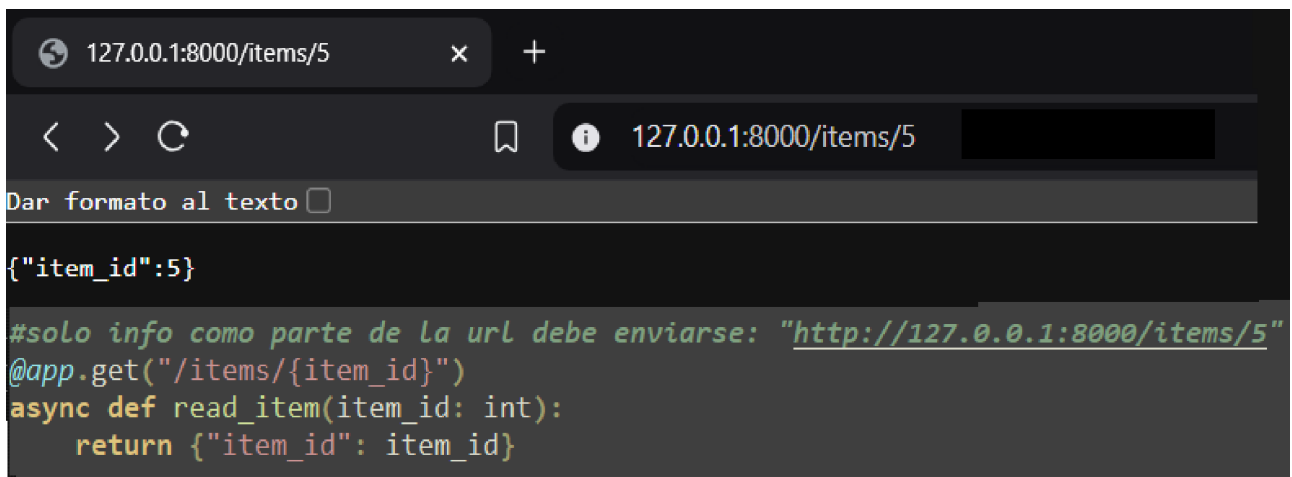
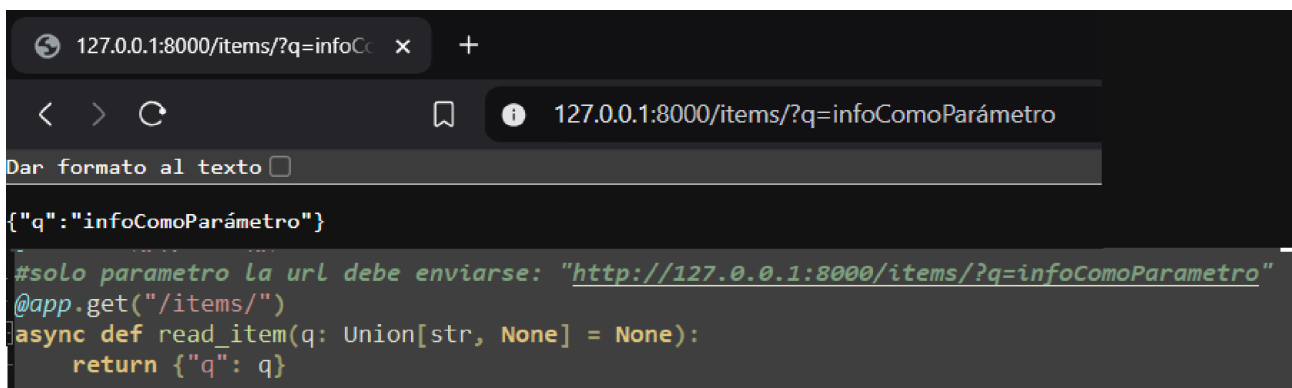
Una, genérica. Nótese que no se adicionó nada en la URL.



Otra, utilizando 2 formas de entregar información a la app, en primer lugar, enviar información inculuida como parte de la url, la segunda, enviar información como parámetro adicionado a la url (la app entenderá que es un parámetro cuando está después de un signo de interrogación).

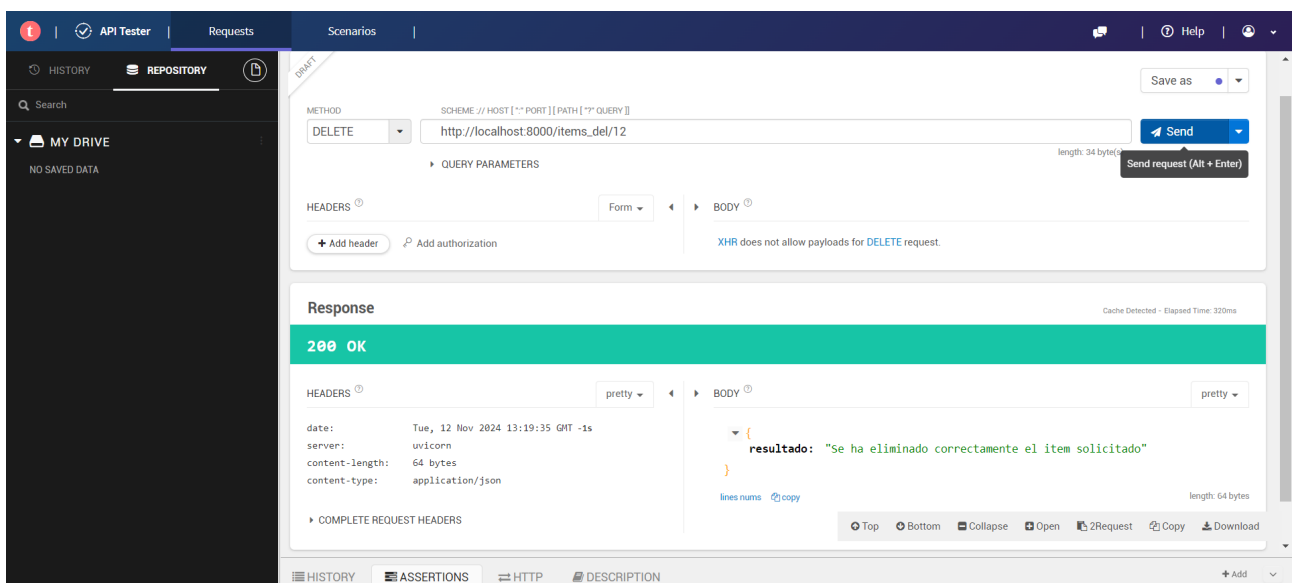


Por lo tanto si se realizan métodos GET que realicen solo una de las dos cosas, el resultado es el siguiente (comparar las imágenes siguientes para observar las diferencias):

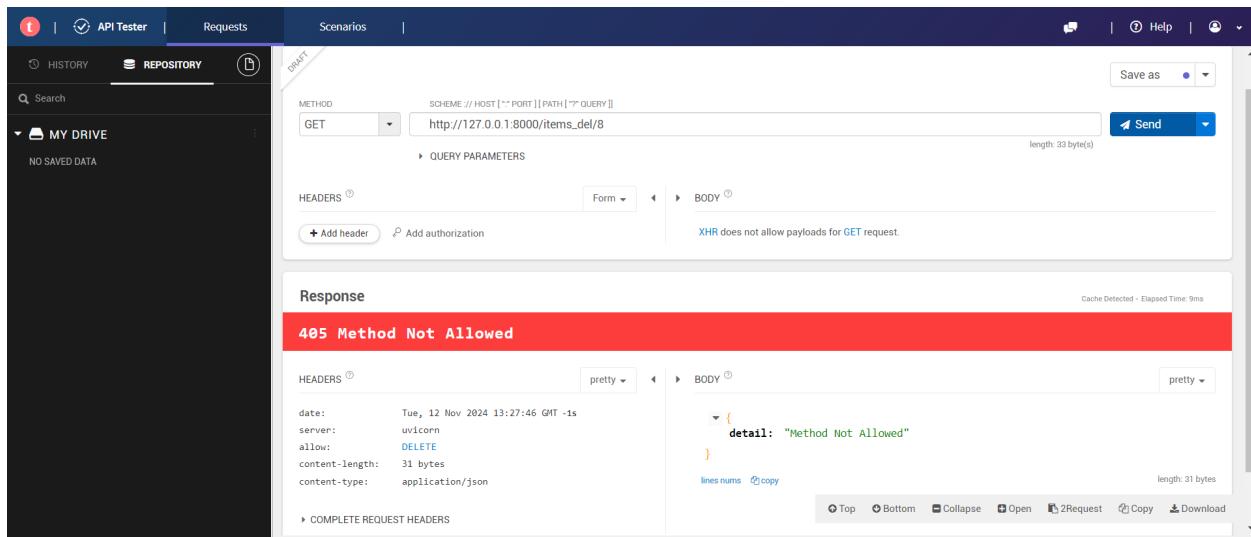
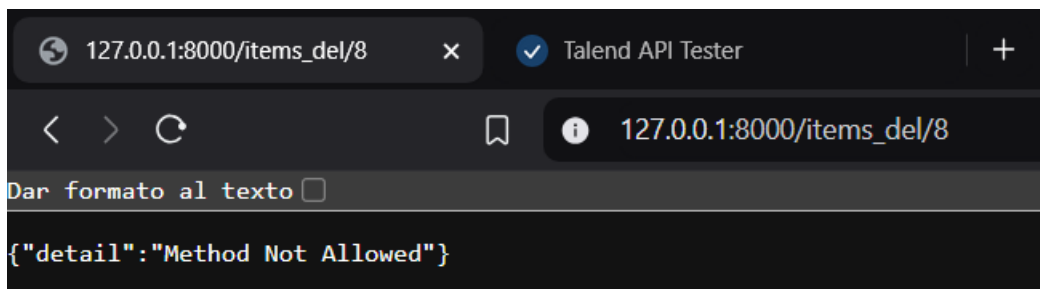


3. Se completarán los métodos de petición http que atenderá nuestra API, a saber, POST, PUT y DELETE

Al igual que el método GET, el método DELETE trabaja suficientemente bien sólo con recibir información en la URL. Por lo tanto un ejemplo básico para observar si realmente se está atendiendo una petición de tipo DELETE se observa en la siguiente imagen.



Es importante aclarar que la url del navegador por defecto siempre responde peticiones de tipo GET, por lo tanto utilizar la url del método DELETE directamente en el navegador generará error, como se observa en las siguientes imágenes:



Esto debido a que el navegador considera que se intenta realizar una petición get. Por esa razón debe utilizarse una herramienta que nos permita indicar que tipo de petición HTTP deseamos hacer. La utilizada anteriormente es externa a la brindada por FASTAPI con el fin de mostrar que efectivamente funciona ante cualquier «Peticionario». Pero es importante recordar que también es válido utilizar la herramienta que nos brinda FASTAPI, cuyo resultado también se incluye.

FastAPI 0.1.0 OAS 3.1  
/openapi.json

### default

- GET / Read Root
- GET /items/ Read Param Item
- GET /items/{item\_id} Read Both Paramtypes Item
- DELETE /items\_del/{item\_id} Delete By Id**

**Parameters** Cancel

Name	Description
item_id <span>required</span>	
integer (path)	22

Execute Clear

**Responses**

Para salir de la pantalla completa, mantén pulsado Esc

**Curl**

```
curl -X 'DELETE' \
'http://127.0.0.1:8000/items_del/22' \
-H 'accept: application/json'
```

**Request URL**

```
http://127.0.0.1:8000/items_del/22
```

**Server response**

Code	Details
200	<p><b>Response body</b></p> <pre>{   "resultado": "Se ha eliminado correctamente el item solicitado" }</pre> <span>Download</span>

**Response headers**

```
content-length: 64
content-type: application/json
date: Tue, 12 Nov 2024 13:29:27 GMT
server: uvicorn
```

**Responses**

Code	Description	Links
200	Successful Response	No links

Media type: application/json

Controls Accept header.

Los dos métodos HTTP restantes son POST y PUT. Estos tienen también algo en común y es que deben utilizar un «colaborador» que les realizará el trabajo de transportar los bloques de información que necesitan entregar. A este colabora-



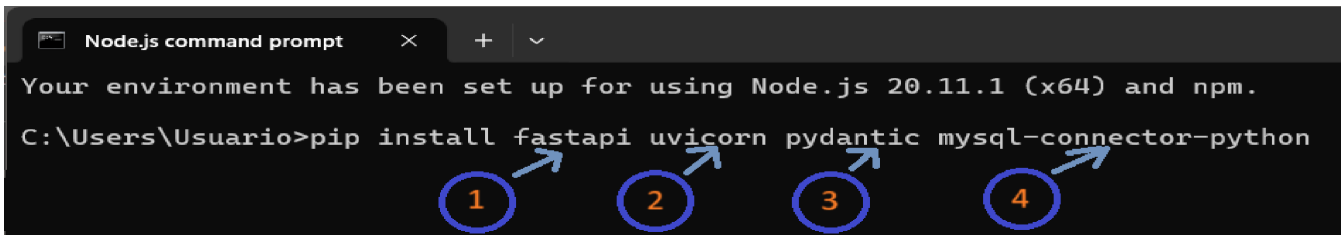
rodor se le denomina BODY, más exactamente BODY REQUEST, para evitar confusiones con el BODY RESPONSE quien es utilizado por los 4 métodos para recibir la respuesta a sus solicitudes. Ahora, para observarles toda su capacidad es necesario previamente establecer un repositorio de información, quien será el otro extremo de la comunicación y con quién realizará este intercambio de bloques de datos. Por lo tanto la creación de los mencionados métodos se observará después de completado el punto siguiente.

#### 4. Conexión de un repositorio de datos.

Para este ejercicio el repositorio es una base de datos relacional y más concretamente construida en el motor de datos MYSQL.

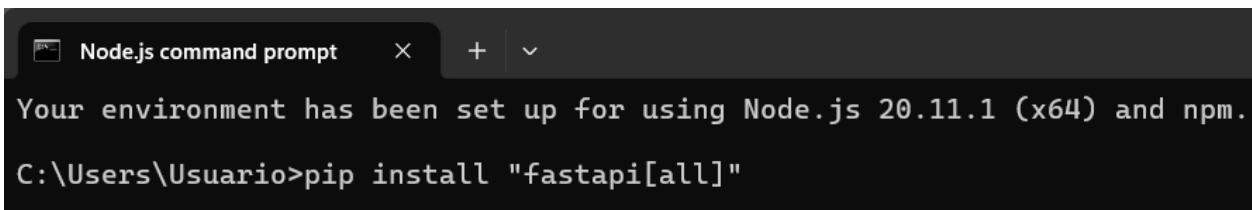
El primer paso es crear la lógica al programa para que pueda comunicarse con su servidor de datos MYSQL es instalar los complementos o plugins que fastAPI requiere para contar con las capacidades necesarias para ello se requiere utilizar la instrucción.

Los 4 complementos que se deben tener son:



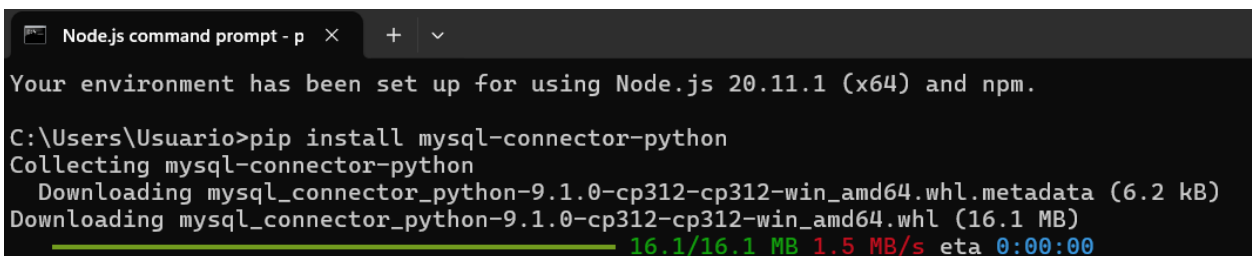
```
Node.js command prompt x + v
Your environment has been set up for using Node.js 20.11.1 (x64) and npm.
C:\Users\Usuario>pip install fastapi uvicorn pydantic mysql-connector-python
```

Cuando se ejecutó la instrucción:

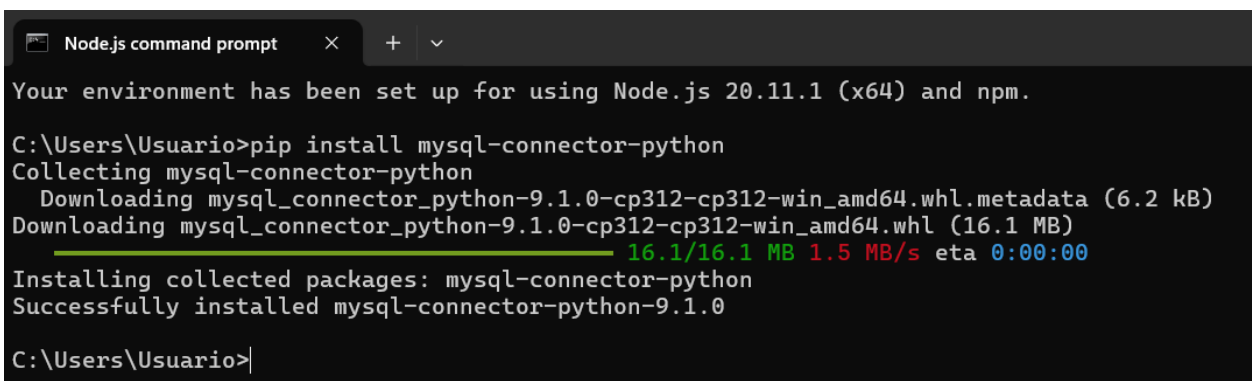


```
Node.js command prompt x + v
Your environment has been set up for using Node.js 20.11.1 (x64) and npm.
C:\Users\Usuario>pip install "fastapi[all]"
```

Se instalaron los primeros 3, ahora solo es necesario realizar la instalación del cuarto.

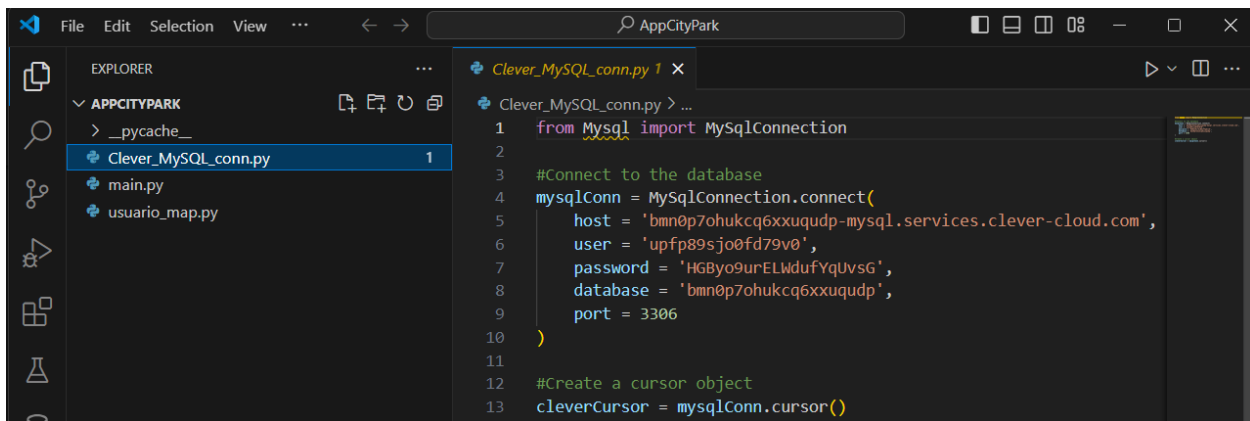


```
Node.js command prompt - p x + v
Your environment has been set up for using Node.js 20.11.1 (x64) and npm.
C:\Users\Usuario>pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.1.0-cp312-cp312-win_amd64.whl.metadata (6.2 kB)
  Downloading mysql_connector_python-9.1.0-cp312-cp312-win_amd64.whl (16.1 MB)
  16.1/16.1 MB 1.5 MB/s eta 0:00:00
```



```
Node.js command prompt x + v
Your environment has been set up for using Node.js 20.11.1 (x64) and npm.
C:\Users\Usuario>pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-9.1.0-cp312-cp312-win_amd64.whl.metadata (6.2 kB)
  Downloading mysql_connector_python-9.1.0-cp312-cp312-win_amd64.whl (16.1 MB)
  16.1/16.1 MB 1.5 MB/s eta 0:00:00
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-9.1.0
C:\Users\Usuario>
```

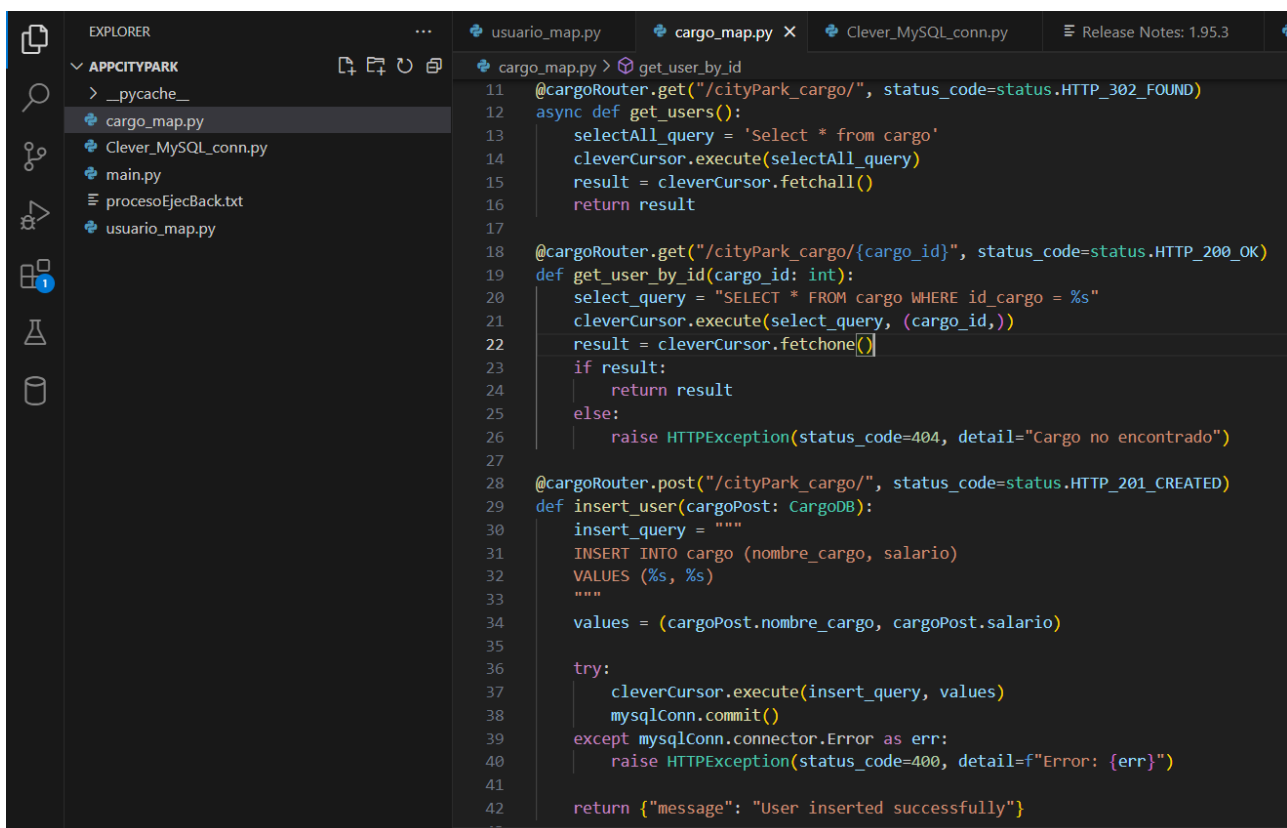
Ahora se crea un archivo de conexión dentro del proyecto:



The screenshot shows the Visual Studio Code editor with a project named 'AppCityPark'. The Explorer panel on the left shows the file structure: `__pycache__`, `Clever_MySQL_conn.py` (selected), `main.py`, and `usuario_map.py`. The main editor displays the code for `Clever_MySQL_conn.py`:

```
1 from MySQL import MySqlConnection
2
3 #Connect to the database
4 mysqlConn = MySqlConnection.connect(
5     host = 'bmn0p7ohukcq6xxuqudp-mysql.services.clever-cloud.com',
6     user = 'upfp89sjo0fd79v0',
7     password = 'HGBYo9urELWdufYqUVsG',
8     database = 'bmn0p7ohukcq6xxuqudp',
9     port = 3306
10 )
11
12 #Create a cursor object
13 cleverCursor = mysqlConn.cursor()
```

Luego se crea el primer archivo de todos los que se deben crear para hacer referencia a cada una de las tablas que hacen parte del proyecto, puntualmente creamos el archivo «cargo» y en la imagen le daremos protagonismo al método POST.



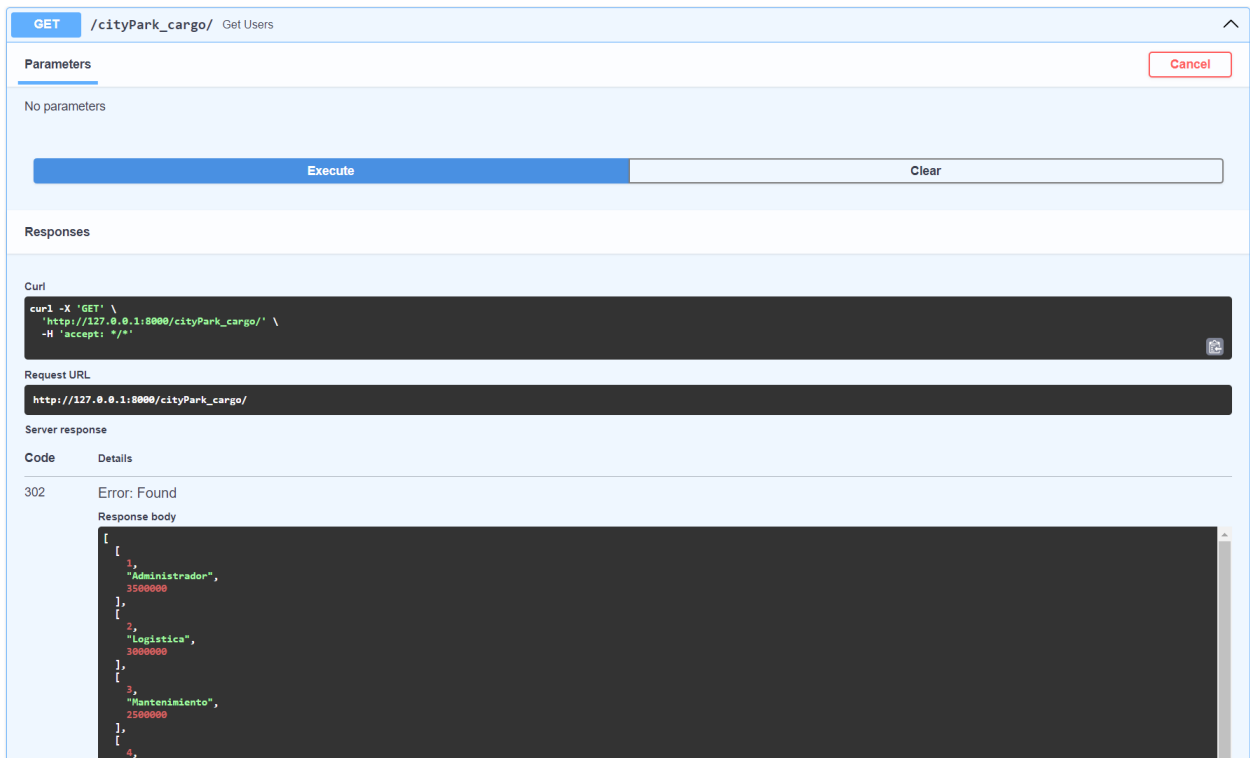
The screenshot shows the Visual Studio Code editor with the `cargo_map.py` file open. The Explorer panel shows the file structure: `__pycache__`, `cargo_map.py` (selected), `Clever_MySQL_conn.py`, `main.py`, `procesoEjecBack.txt`, and `usuario_map.py`. The main editor displays the code for `cargo_map.py`:

```
11 @cargoRouter.get("/cityPark_cargo/", status_code=status.HTTP_302_FOUND)
12 async def get_users():
13     selectAll_query = 'select * from cargo'
14     cleverCursor.execute(selectAll_query)
15     result = cleverCursor.fetchall()
16     return result
17
18 @cargoRouter.get("/cityPark_cargo/{cargo_id}", status_code=status.HTTP_200_OK)
19 def get_user_by_id(cargo_id: int):
20     select_query = "SELECT * FROM cargo WHERE id_cargo = %s"
21     cleverCursor.execute(select_query, (cargo_id,))
22     result = cleverCursor.fetchone()
23     if result:
24         return result
25     else:
26         raise HTTPException(status_code=404, detail="Cargo no encontrado")
27
28 @cargoRouter.post("/cityPark_cargo/", status_code=status.HTTP_201_CREATED)
29 def insert_user(cargoPost: CargoDB):
30     insert_query = """
31     INSERT INTO cargo (nombre_cargo, salario)
32     VALUES (%s, %s)
33     """
34     values = (cargoPost.nombre_cargo, cargoPost.salario)
35
36     try:
37         cleverCursor.execute(insert_query, values)
38         mysqlConn.commit()
39     except mysqlConn.connector.Error as err:
40         raise HTTPException(status_code=400, detail=f"Error: {err}")
41
42     return {"message": "User inserted successfully"}
```

Se realiza el consumo de la url parametizada en el backend para observar la información de los cargos:



Al observarlo desde la herramienta brindada por fastAPI se observa:



Ahora se confirmará que el método «POST» funcione correctamente. Para ello observamos en primer lugar la base de datos, luego realizamos un proceso de inserción desde «TALEND» (recordar que es lo mismo que si lo realizáramos desde la herramienta brindada por fastAPI desarrollada en «SWAGGER»), para finalmente volver a consultar nuevamente la base de datos y confirmar que efectivamente se creó el nuevo registro:

Base de datos previa al proceso

carga x <audisofttest> Script-9 <audisofttest> Script-5				
Propiedades Datos Diagrama ER				
carga Enter a SQL expression to filter results (use Ctrl+Space)				
Grilla	id_carga	nombre_carga	salario	
1	1	Administrador	3.500.000	
2	2	Logistica	3.000.000	
3	3	Mantenimiento	2.500.000	
4	4	Operacion	2.000.000	
5	5	Publicidad	4.000.000	

Proceso de creación de un nuevo cargo

a. El método «POST» dentro de la herramienta brindada por fastAPI muestra como debe ser el JSON que espera el proyecto para que pueda procesarlo. Se creará uno idéntico para realizar la prueba desde «TALEND»

POST /cityPark\_crea\_cargo/ Insert User

Parameters

No parameters

Request body

application/json

```
{
  "nombre_cargo": "string",
  "salario": 0
}
```

Execute

Clear

Responses

Curl

curl -X 'POST' \
'http://127.0.0.1:8000/cityPark\_crea\_cargo/' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
 "nombre\_cargo": "string",
 "salario": 0
}'

Request URL

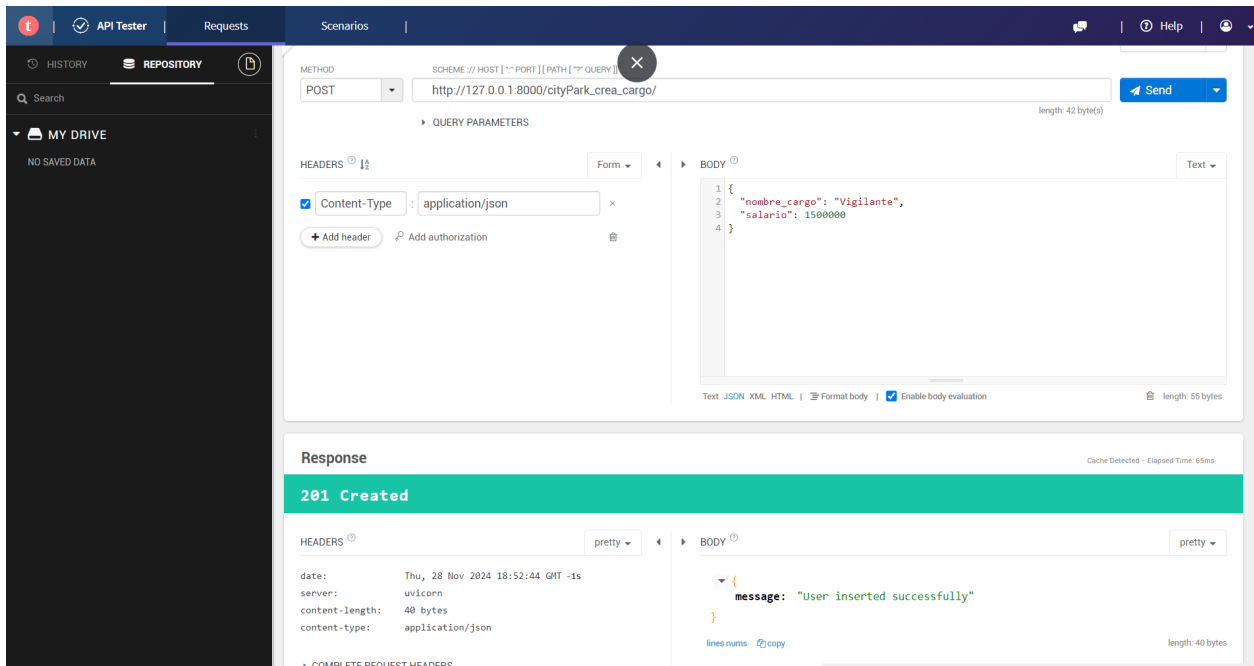
http://127.0.0.1:8000/cityPark\_crea\_cargo/

Server response

Code	Details
201	<div><div>Response body</div><div><pre>{   "message": "User inserted successfully" }</pre></div><div><div>Response headers</div><div>content-length: 40 content-type: application/json date: Thu, 28 Nov 2024 18:51:03 GMT server: uvicorn</div></div></div>

Responsees

Code	Description	Links
201	<div>Successful Response</div> <div><div>Media type</div><div>application/json</div><div>Controls Accept header.</div><div>Example Value   Schema</div><div>"string"</div></div> <div>No links</div>	No links
422	<div>Validation Error</div> <div><div>Media type</div><div>application/json</div><div>Example Value   Schema</div><div><pre>{   "detail": [     {       "loc": [         "string",         0       ]     }   ] }</pre></div></div> <div>No links</div>	No links



b. Y al consultar la base de datos tenemos:

The screenshot shows the DBeaver 23.3.4 interface. The database is `audisofttest` and the table is `carga`. The query result is as follows:

	id_cargo	nombre_cargo	salario
1	1	Administrador	3.500.000
2	2	Logistica	3.000.000
3	3	Mantenimiento	2.500.000
4	4	Operacion	2.000.000
5	5	Publicidad	4.000.000
6	8	Vigilante	1.500.000

5. Enlace con un proyecto front React.

Se cuenta con un proyecto react fronend, al cual ya se le ha incluido el endpoint que expone la APP para crear registros en Cargo.

Se encuentra adjunto al proyecto back en el repositorio de github.

Formacion > reactJS > Framework_JS > FrontParaFastApi >			
<div> <div> <div></div> <div></div> <div></div> <div></div> </div> <div> <div>Ordenar</div> <div>Ver</div> <div></div> </div> </div>			
Nombre	Fecha de modificación	Tipo	Tamaño
node_modules	28/11/2024 10:17	Carpeta de archivos	
public	28/11/2024 9:53	Carpeta de archivos	
src	28/11/2024 9:53	Carpeta de archivos	
package.json	28/11/2024 10:14	JSON File	1 KB
package-lock.json	28/11/2024 10:14	JSON File	695 KB
primerArranque.log	27/11/2024 23:21	Text Document	100 KB
segundoArranqueOK.log	28/11/2024 11:03	Text Document	0 KB

Al realizar el intento de registrar un nuevo cargo desde este proyect frontend se observa:

## Formulario cargo

Nombre del cargo

Salario

Cargar

```

DevTools - localhost:3000/
DevTools is now available in Spanish!
Always match Chrome's language Switch DevTools to Spanish Don't show again
Elements Console Sources Network Performance Memory Application >>
top Filter Default levels 2 Issues: 2 1 hidden
Download the React DevTools for a better development experience: https://reactjs.org/link/react-devtools
react_refresh:6
  {nombre_cargo: 'Supervisor Vigilancia', salario: '1800000'} FormCargo.jsx:14
el nuevo registro fue un exito... FormCargo.jsx:17
  {nombre_cargo: 'Supervisor Vigilancia', salario: '1800000'}

```

DBeaver 23.3.4 - cargo

Archivo Editar Navegar Buscar Editor SQL Base de Datos Ventana Ayuda

SQL Commit Rollback Auto audisofttest < N/A >

Navegador de Bases de Datos

Ingrese parte del nombre de un objeto aquí

audisofttest - localhost:3306

Databases

citypark

Tables

- acudiente 32K
- atraccion 16K
- atraccion\_condicionuso\_map 48K
- cargo 32K
- condicionuso 16K
- empleado 32K
- estacion 16K

<audisofttest> Script-9 <audisofttest> Script-5 user

Propiedades Datos Diagrama ER

cargo Enter a SQL expression to filter results (use Ctrl+Space)

	id_cargo	nombre_cargo	salario
1	1	Administrador	3.500.000
2	2	Logistica	3.000.000
3	3	Mantenimiento	2.500.000
4	4	Operacion	2.000.000
5	5	Publicidad	4.000.000
6	8	Vigilante	1.500.000
7	9	Supervisor Vigilancia	1.800.000