FastAPI

CONTENTS:

API Basics

An API (Application Programming Interface) allows two computer systems to interact with each other. For example, if we create an automation that generates a report and sends it by email, the sending of that email is not done manually, the script itself will do it. To do this, Python (or the language we use), must ask Gmail to send that email, with that report attached to certain people. The way to do it is through an API, in this case the Gmail API.

- **Features of FastAPI:**

  - FastAPI is a python **microframework.**
  - **Fast to code**: Increase the speed to develop features by about 200% to 300%.
  - **Fewer bugs**: Reduce about 40% of human (developer) induced errors.
  - **Easy**: Designed to be easy to use and learn. Less time reading docs.
  - Automatic data model documentation with Json schema (as OpenAPI itself is based on JSON Schema).
  - Designed around these standards, after a meticulous study. Instead of an afterthought layer on top.
  - This also allows using automatic **client code generation** in many languages.
  - Open API for API creation, including declarations of path operations, parameters, body requests, security, etc.
  - **FastAPI** is fully compatible with (and based on) pydantic. So, any additional Pydantic code you have, will also work.
  - Including external libraries also based on Pydantic, as ORMs, ODMs for databases.
  - This also means that in many cases you can pass the same object you get from a request **directly to the database**, as everything is validated automatically.
  - Custom Exceptions and customer Exception handlers.
  - Uses models as if it is was in Django.

How to create an API in Python with FastAPI

FastAPI is a way of creating APIs in Python that came out at the end of 2018. It is very fast, although it can only be used with **Python 3.6+** (in my opinion this should not be a problem, but it is important).

To use it, you must install two libraries: fastapi and uvicorn.

Installation:

Requirements:

Python 3.6+

#To install FastAPI

pip install fastapi

pip install sqlalchemy

pip install pymysql

pip install mysqlclient

pip install "uvicorn[standard]"

#To install contrib. with mongodbsupport

Pip installfastapi_contrib[mongo]

#To install contrib. with json support

Pip **install fastapi_contrib[ujson**]

HTTP transfer protocol : it is the main way of communicating information on the web. There are different methods, each of them used for different issues:

- **GET**: this method allows obtaining information from the database or from a process.
- **POST**: allows you to send information, either to add information to a database or to pass the input of a machine learning model, for example.
- **PUT:** update information. It is generally used to manage information in the database.

- **DELETE**: this method is used to delete information from the database.
- **Url**: is the address where we can find our API. Basically this URL will consist of three parts:
  - **Protocol**: like any address, it can be http:// or https://
  - **Domain**: the host on which it is hosted, which goes from the protocol to the end of .com, or whatever ending the url has. On my website, for example, the domain is anderfernandez.com.
  - **Endpoint** : like a website has several pages (/ blog), (/ legal), the same API can include multiple points and each one does different things. When creating our API in Python we will indicate the endpoints, so we must make sure that each enpoint is representative of what the API behind it does.

Virtual environment: to create a virtual environment

Pip –m venv myenv

Project\fastapi\myenv\Scripts\activate.

Example:

User table contain a fields as follows,

employee id(pk)	Name	contact	email(unique)	department

1.Read all the records from database[GET method]:

```
from fastapi import FastAPI
app= FastAPI()
@user.get("/")
 async def read_data("/"):
```

```
    return conn.execute(users.select()).fetchall()
```

2.Read a particular data from database[GET method]:

```
@user.get("/employee_id")
async def read_data(employee_id:int):
    return conn.execute(users.select(). where(users.c.employee_id==employee_id)).fetchall()
```

3.Insert new data into table[POST method]:

```
@user.post("/")
async def create_data(user:User):
    conn.execute(users.insert().values(
        Name=user.Name,
        contact=user.contact,
        email=user.email,
        department=user.department
    ))
    return conn.execute(users.select()).fetchall()
```

4. Update any data from table[PUT method]:

```
@user.put("/employee_id"):
async def update_data(employee_id:int, user:User):
    conn.execute(users.update().values(
        Name=user.Name,
        contact=user.contact,
        email=user.email,
        department=user.department
    ). where(users.c.employee_id==employee_id))
 return conn.execute(users.select()).fetchall()
```

Run it

Run the server with:

uvicorn main:app --reload

The command uvicorn main:app --reload

- main: the file main.py (the Python "module").
- app: the object created inside of main.py with the line app = FastAPI().
- --reload: make the server restart after code changes. Only do this for development.

Check it : Interactive API docs

Open your browser at http://127.0.0.1:8000/

http://127.0.0.1:8000/docs#/default/read_data__get

http://127.0.0.1:8000/docs#/default/read_data__employee_id__get

http://127.0.0.1:8000/docs#/default/create_data__post

http://127.0.0.1:8000/docs#/default/update_data_employee_id_put

Go to browser to check the API excecution.

FastAPI packages:

- Fastapi_contrib.auth package
- Fast api_contrib.auth.middlewares module
- Fast api_contrib.auth serializers module
- Fast api_contrib.common.middlewares module
- Fast api_contrib.common.utils module.