Guide d'installation et premiers pas avec FastAPI

Djebabla Ammar

5 novembre 2025

1 Introduction

FastAPI est un framework web moderne, rapide et intuitif pour construire des API avec Python 3.7+ basé sur les annotations de types. Ce guide vous accompagne dans :

— L'installation et la configuration de FastAPI.
— La création d'une première API simple.
— Quelques exemples pratiques avec requêtes et réponses JSON.

2 Installation de FastAPI et Uvicorn

2.1 1. Créer un environnement virtuel

Ouvrez votre terminal (ou PowerShell sous Windows) et tapez :

```
python -m venv venv
source venv/bin/activate   # Sur Linux ou Mac
venv\Scripts\activate      # Sur Windows
```

2.2 2. Installer FastAPI et le serveur Uvicorn

```
pip install fastapi uvicorn
```

Pour vérifier l'installation :

```
pip list
```

Vous devriez voir `fastapi` et `uvicorn` dans la liste.

3 Création d'une première API

Créez un fichier `main.py` et ajoutez le code suivant :

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
```

```
    return {"message": "Bienvenue sur votre première API FastAPI !"}

@app.get("/hello/{name}")
def say_hello(name: str):
    return {"message": f"Bonjour {name}, ravi de te voir ici !"}

# Démarrage du serveur :
# uvicorn main:app --reload
```

Lancez ensuite le serveur :

```
uvicorn main:app --reload
```

Allez dans votre navigateur à l'adresse :

http://127.0.0.1:8000

3.1 Documentation automatique

FastAPI fournit automatiquement deux interfaces interactives :
— **Swagger UI** : http://127.0.0.1:8000/docs
— **ReDoc** : http://127.0.0.1:8000/redoc

4 Exemple : Requête POST avec modèle Pydantic

Créons un modèle de données pour recevoir des informations depuis le client.

```
from pydantic import BaseModel

class Utilisateur(BaseModel):
    nom: str
    age: int
    email: str

@app.post("/utilisateur/")
def creer_utilisateur(user: Utilisateur):
    return {"message": f"Utilisateur {user.nom} ajouté avec succès."}
```

Pour tester, vous pouvez utiliser Swagger UI et soumettre le JSON suivant :

```
{
  "nom": "Ammar",
  "age": 30,
  "email": "ammar@example.com"
}
```

5 Exemple : Paramètres de requête et valeurs par défaut

```python
@app.get("/produit/")
def lire_produit(nom: str = "Inconnu", prix: float = 0.0):
    return {"produit": nom, "prix": prix}
```

Essayez dans votre navigateur :

http://127.0.0.1:8000/produit?nom=PC&prix=1500

6 Exemple : Simulation d'une base de données simple

```python
fake_db = []

@app.post("/items/")
def create_item(item: dict):
    fake_db.append(item)
    return {"db_size": len(fake_db)}

@app.get("/items/")
def list_items():
    return {"items": fake_db}
```

7 Exemple avancé : API avec dépendance

```python
from fastapi import Depends, HTTPException

def get_token_header(token: str = "12345"):
    if token != "12345":
        raise HTTPException(status_code=400, detail="Token invalide")
    return token

@app.get("/secure-data/", dependencies=[Depends(get_token_header)])
def secure_data():
    return {"message": "Accès autorisé"}
```

8 Conclusion

Vous avez installé FastAPI, exploré les routes GET et POST, et vu comment utiliser les modèles de données avec Pydantic. Prochaine étape : connecter FastAPI à une base de données (SQLite, PostgreSQL ou autre) et gérer l'authentification.

Bon codage et amusez-vous avec FastAPI !