

Guide d'Utilisation de GraphQL avec FastAPI

Ammar Djebabla

27 novembre 2025

Résumé

Ce guide présente l'utilisation de GraphQL avec FastAPI, un framework web Python moderne. Il explique l'installation, la configuration et fournit des exemples pratiques de requêtes et de mutations.

1 Introduction

GraphQL est un langage de requête pour les API, permettant aux clients de récupérer exactement les données dont ils ont besoin. FastAPI est un framework Python rapide, moderne et facile à utiliser. L'intégration de GraphQL avec FastAPI offre une API flexible et performante.

2 Installation

2.1 Créer un environnement virtuel

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

Listing 1 – Création d'un environnement virtuel

2.2 Installer FastAPI et GraphQL

```
pip install fastapi
pip install "uvicorn[standard]"
pip install strawberry-graphql
```

Listing 2 – Installation des dépendances

3 Configuration de base

3.1 Création du serveur FastAPI avec GraphQL

```

from fastapi import FastAPI
import strawberry
from strawberry.fastapi import GraphQLRouter

# Définition d'un type GraphQL
@strawberry.type
class User:
    id: int
    name: str
    email: str

# Définition du schéma
@strawberry.type
class Query:
    @strawberry.field
    def hello(self) -> str:
        return "Hello from GraphQL"

    @strawberry.field
    def user(self) -> User:
        return User(id=1, name="Alice", email="alice@example.com")

schema = strawberry.Schema(query=Query)

# Crée l'application FastAPI
app = FastAPI()

# Ajouter le routeur GraphQL
graphql_app = GraphQLRouter(schema)
app.include_router(graphql_app, prefix="/graphql")

```

Listing 3 – Serveur FastAPI avec Strawberry GraphQL

3.2 Lancer le serveur

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

Listing 4 – Lancement du serveur FastAPI

Accéder à l'interface GraphQL : <http://127.0.0.1:8000/graphql>

4 Exemples de requêtes GraphQL

4.1 Requête simple

```
query {
  hello
}
```

Listing 5 – Requête GraphQL pour récupérer un message

4.2 Requête pour récupérer un utilisateur

```
query {
  user {
    id
    name
    email
  }
}
```

Listing 6 – Requête GraphQL pour récupérer un utilisateur

4.3 Mutation GraphQL

```
@strawberry.type
class Mutation:
    @strawberry.mutation
    def create_user(self, name: str, email: str) -> User:
        return User(id=2, name=name, email=email)

schema = strawberry.Schema(query=Query, mutation=Mutation)
```

Listing 7 – Ajout d'une mutation dans FastAPI

Exemple de mutation :

```
mutation {
  createUser(name: "Bob", email: "bob@example.com") {
    id
    name
    email
  }
}
```

Listing 8 – Créer un nouvel utilisateur

5 Meilleures pratiques

- Utiliser des types explicites pour toutes les données.
- Valider les données côté serveur pour éviter les erreurs.
- Limiter la profondeur des requêtes pour prévenir les attaques de type $N+1$.
- Utiliser les résolveurs pour séparer la logique métier.

6 Conclusion

GraphQL avec FastAPI permet de construire des API flexibles, performantes et faciles à maintenir. Grâce à Strawberry, l'intégration est rapide et le code reste clair.