

## GraphQL integráció

Főbb tulajdonságok összehasonlítása a RESTAPI-val.

Hivatalos dokumentáció:

<https://graphql.org/learn/>

	REST API	GraphQL
<b>Adatlekérdezés rugalmassága</b>	Fix végpontok és válaszok	Rugalmas lekérdezések, csak a szükséges adatokat kapod meg
<b>Kliens teljesítmény</b>	Többszöri kérések szükségesek különböző erőforrásokhoz	Egyetlen kérésben több adatforrásból is lekérhetők az adatok
<b>Adatok</b>	Több lekérdezésre lehet szükség összetett adatokhoz	Egyetlen kérésben lekérdezhető minden szükséges adat
<b>Hálózati hatékonyság</b>	Többszöri kérés esetén megnő a hálózati forgalom	Csökkenti a felesleges adatátvitelt és lekérdezések számát
<b>Használhatóság</b>	Egyszerű, széles körben ismert	Nehezebben elsajátítható, de hatékonyabb
<b>Gyorsítótárazás (Caching)</b>	HTTP cache és CDN támogatott	Nehezebb megvalósítani, egyedi cache stratégia szükséges
<b>Verziókezelés</b>	API verziózás szükséges (pl. `/v1/users`)	Nincs szükség verziókezelésre, mert a kliens választja ki a szükséges mezőket
<b>Adatstruktúra változás kezelése</b>	Módosítások új végpontok létrehozását igényelhetik	Az új mezők bevezethetők a régiak megtartásával
<b>Biztonság és hozzáférés-kezelés</b>	Beépített HTTP biztonság, szerepkörök	Finomhangolt hozzáférési szabályok szükségesek (pl. mezőszintű jogosultságok)
<b>Támogatott formátumok</b>	Általában JSON (de támogat másokat is)	JSON alapú (szigorúan GraphQL schema szerint)
<b>Alkalmazási körök</b>	Egyszerű CRUD API-khoz ideális	Összetett, dinamikus kliensigényekhez jobb választás

Python virtuális környezet kialakítás után az alábbi sorral telepíthetjük a függőségeket:

```
pip install fastapi strawberry-graphql uvicorn
```

Mintapélda az első használathoz:

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

# User adatmodell
@strawberry.type
class User:
    id: int
```

```
name: str
age: int

# Példa adatbázis
users = [
    User(id=1, name="Noa", age=30),
    User(id=2, name="Anna", age=25),
]

# GraphQL Query osztály
@strawberry.type
class Query:
    @strawberry.field
    def get_users(self) -> list[User]:
        return users

# GraphQL Mutáció osztály (új felhasználó hozzáadása)
@strawberry.type
class Mutation:
    @strawberry.mutation
    def create_user(self, name: str, age: int) -> User:
        new_user = User(id=len(users) + 1, name=name, age=age)
        users.append(new_user)
        return new_user

# GraphQL séma létrehozása
schema = strawberry.Schema(query=Query, mutation=Mutation)

# FastAPI alkalmazás létrehozása
app = FastAPI()

# GraphQL endpoint regisztrálása
graphql_app = GraphQLRouter(schema)
app.include_router(graphql_app, prefix="/graphql")
```

```
uvicorn.exe main:app --reload
```

A GraphQL Playground a következő URL-en elérhető lesz: <http://127.0.0.1:8000/graphql>

From: <https://edu.iit.uni-miskolc.hu/> - Institute of Information Science - University of Miskolc

Permanent link: [https://edu.iit.uni-miskolc.hu/tanszek:oktatas:informacios\\_rendszerek\\_integralasa:graphql\\_integracio?rev=1744043782](https://edu.iit.uni-miskolc.hu/tanszek:oktatas:informacios_rendszerek_integralasa:graphql_integracio?rev=1744043782)

Last update: 2025/04/07 16:36

