

## GraphQL integráció

A GraphQL egy modern API-lekérdezési nyelv és futtatókörnyezet, amelyet a Facebook fejlesztett ki 2012-ben, majd 2015-ben nyílt forráskódúvá tett. A GraphQL célja, hogy rugalmasabb és hatékonyabb adatlekérdezési lehetőséget biztosítson a kliensalkalmazások számára.

A hagyományos REST API-k esetén a kliens különböző endpointokon keresztül kér le adatokat, és gyakran előfordul, hogy több HTTP-kérésre van szükség az összes szükséges információ megszerzéséhez. Emellett a szerver által visszaadott adatszerkezet fix, így a kliens sokszor több adatot kap, mint amire valójában szüksége van.

A GraphQL ezt a problémát úgy oldja meg, hogy a kliens pontosan megadhatja, milyen adatmezőket szeretne lekérdezni. A szerver ennek megfelelően csak a kért adatokat küldi vissza. Ez csökkenti a hálózati forgalmat és egyszerűbbé teszi az összetett adatszerkezetek lekérdezését.

A GraphQL működésének három alapvető eleme van:

- **Schema** - a rendszer adatmodelljének és lekérdezési lehetőségeinek leírása. A schema határozza meg, milyen típusok, mezők és műveletek érhetők el az API-ban.
- **Query** - adatlekérdezésre szolgál. A kliens meghatározza, hogy milyen mezőket szeretne visszakapni.
- **Mutation** - adatmódosító műveletek (például létrehozás, módosítás vagy törlés).

A GraphQL egyik fontos jellemzője, hogy általában **egyetlen endpointot** használ (például `/graphql``). A különböző műveleteket nem az URL-ek, hanem a lekérdezések szerkezete határozza meg.

A GraphQL különösen előnyös olyan rendszerekben, ahol:

- a kliensalkalmazások különböző adatszerkezeteket igényelnek,
- összetett, egymáshoz kapcsolódó adatok lekérdezése szükséges,
- fontos a hálózati forgalom minimalizálása.

Ugyanakkor a GraphQL implementációja általában összetettebb, mint egy hagyományos REST API, és a gyorsítótárazás vagy a jogosultságkezelés megvalósítása is több tervezést igényel.

Az alábbi részben bemutatjuk a GraphQL alapvető jellemzőit, valamint egy egyszerű Python alapú GraphQL API implementációját FastAPI és Strawberry könyvtárak segítségével.

Főbb tulajdonságok összehasonlítása a RESTAPI-val.

Hivatalos dokumentáció:

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

	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 kérdésben lekérdezhető minden szükséges adat
<b>Hálózati hatékonyság</b>	Többszöri kérés esetén megnő a hálózati forgalom	Csökkenti a felesleges adatátvitelt és lekérdezések számát
<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 kialakítása:

```
python -m virtualenv .venv
```

Majd ez 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>

---

## Mintafeladat: Könyvtári rendszer GraphQL-lel

### Feladat leírása

Készítsünk egy egyszerű GraphQL API-t, amely egy könyvtár adatait kezeli. A rendszer tárolja a könyveket és a szerzőket, valamint lehetőséget ad új könyv felvitelére.

### Követelmények

A GraphQL séma tartalmazza az alábbi típusokat:

- **Author:**
  - `id` (Int)
  - `name` (String)
- **Book:**
  - `id` (Int)
  - `title` (String)

- `author` (Author)
- `year` (Int)

## Implementálandó funkciók

- **Lekérdezés:**
  - Az összes könyv lekérdezése (cím, szerző neve, év)
  - Egy szerző könyveinek lekérdezése név alapján
- **Mutáció:**
  - Új könyv hozzáadása a következő adatokkal: cím, szerző ID, év

## Példa lekérdezés

```
query {  
  books {  
    title  
    author {  
      name  
    }  
    year  
  }  
}
```

## Példa mutáció

```
mutation {  
  addBook(title: "1984", authorId: 1, year: 1949) {  
    id  
    title  
    author {  
      name  
    }  
  }  
}
```

## Technikai követelmények

- Használjunk **FastAPI + Strawberry GraphQL** könyvtárakat
- Tárolásra használjunk beépített listákat (pl. `authors`, `books`)
- A GraphQL endpoint legyen elérhető a <http://127.0.0.1:8000/graphql> címen

## Kiegészítő feladat

- Készítsünk egy új mutációt: egy szerző hozzáadása név alapján

- Lekérdezés, amely egy évszám alapján listázza a megjelent könyveket

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](https://edu.iit.uni-miskolc.hu/tanszek:oktatas:informacios_rendszerek_integralasa:graphql_integracio)

Last update: **2026/04/24 11:42**

