CONF42

Building GraphQL microservices using FastAPI

CONF42

Aby M Joseph

Product Engineer

UST | Strollby

Jayalekshmi KS

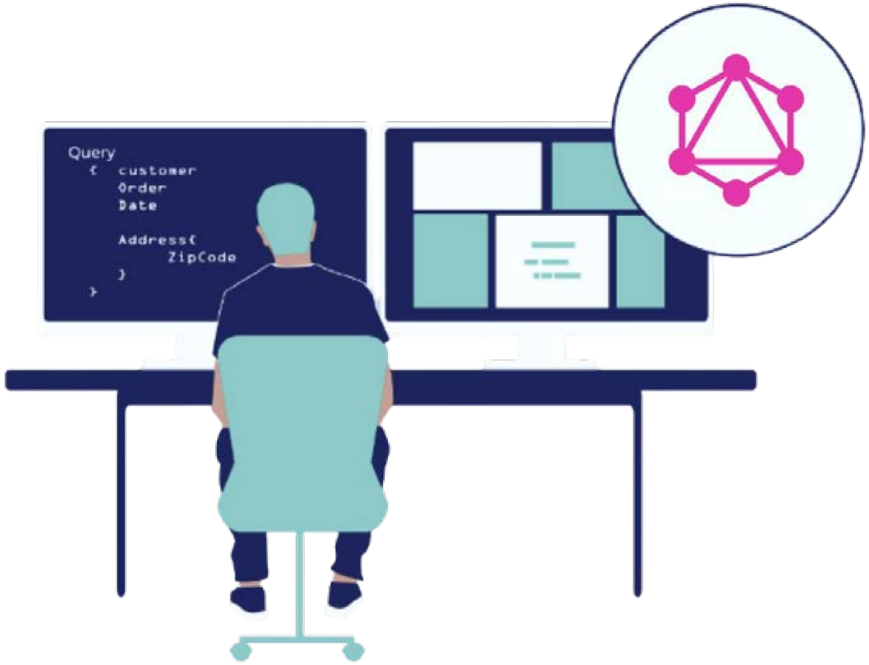
Product Engineer

UST | Strollby

GraphQL

A query language for your API's.

Ask for what you need, get exactly that.



Challenges with REST APIs

OVERFETCHING

UNDERFECTHING

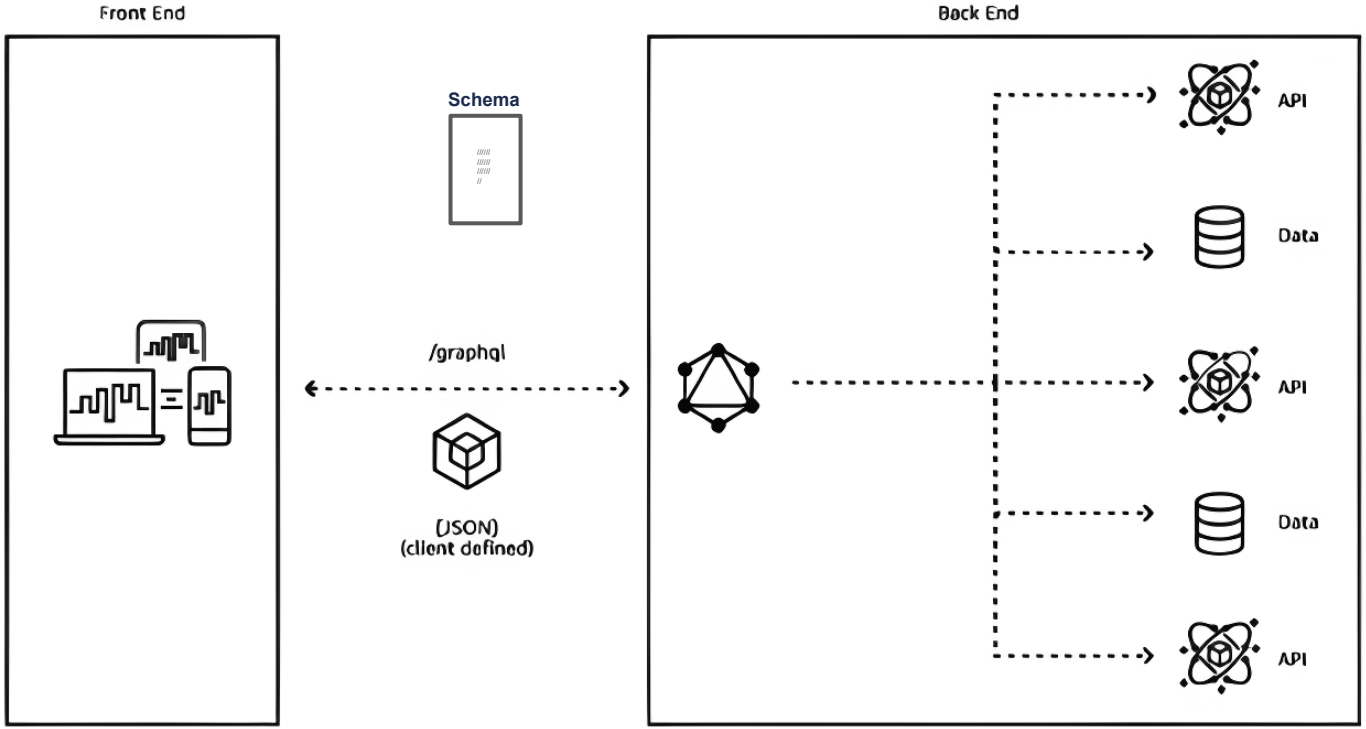
NOT ENOUGH DATA PER REQUEST

Challenges with REST APIs

Frontend

Backend

GraphQL Core



GraphQL Operations

Query

Read data

```
query {
  user(id:"1"){
        first_name
        last_name
      }
}
```

Mutations

write data

```
mutation{
   createPost(post:"P1"){
        id
 }
}
```

Subscription

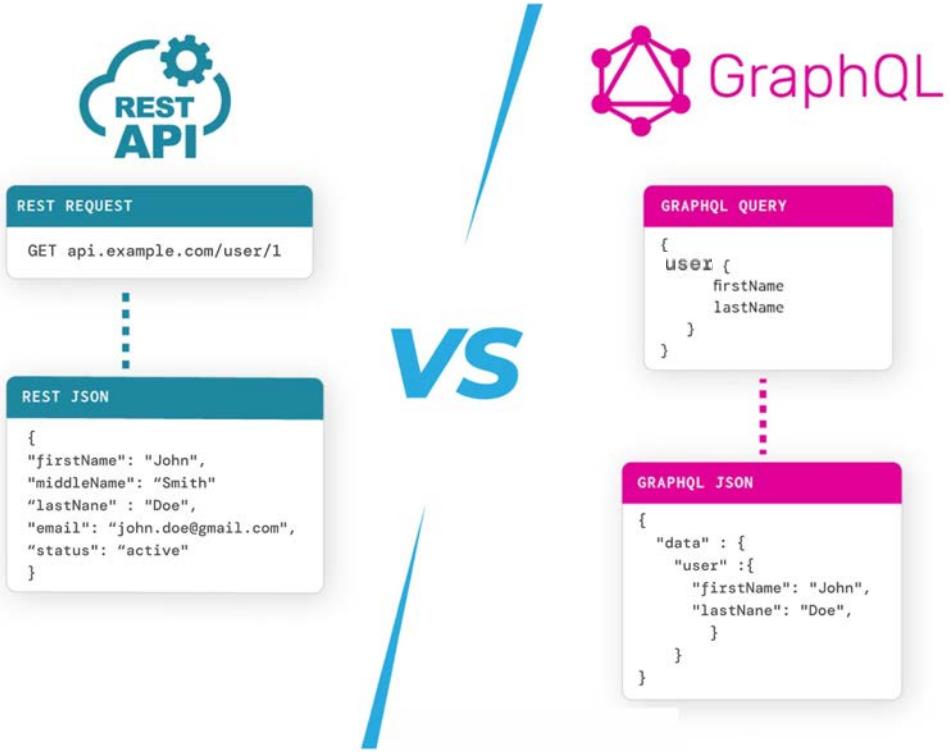
Listen for data

```
subscription{
  onCreate{
    id
  }
}
```

GraphQL Schema

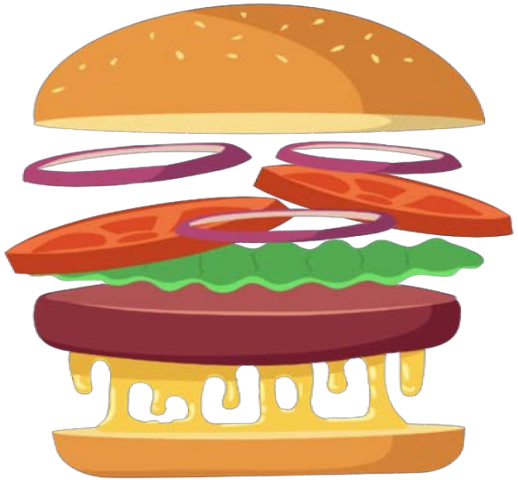
```
# Define the User type with three fields
type User {
  id: ID!
  name: String!
  email: String!
}

# Define the Query type to fetch user data
type Query {
  getUser(id: ID!): User
}

# Define the Mutation type to update a user's name
type Mutation {
  updateUserName(id: ID!, name: String!): User
}
```

Rest vs GraphQL



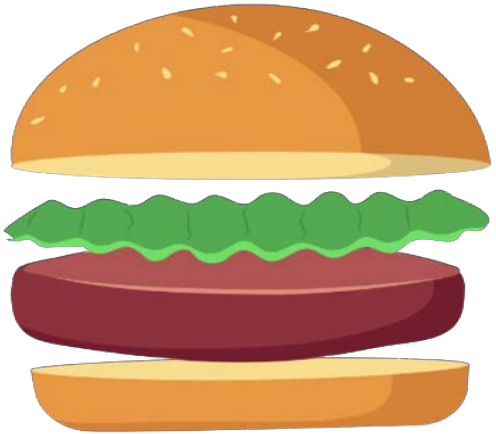
Rest vs GraphQL - Analogy

```
https://api.com/cheeseburger/
```

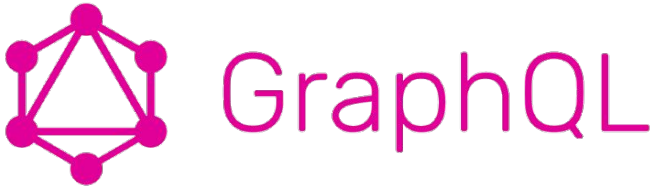


VS

```
query getCheeseBurger ($vegan: Boolean) {
    cheeseburger{
        bun
        lettuce
        Patty
        bun
        cheese @skip(if:$vegan)
        }
}
```



GraphQL Python Libraries



GraphQL Python Libraries

Code First

Graphene

Strawberry

Schema First

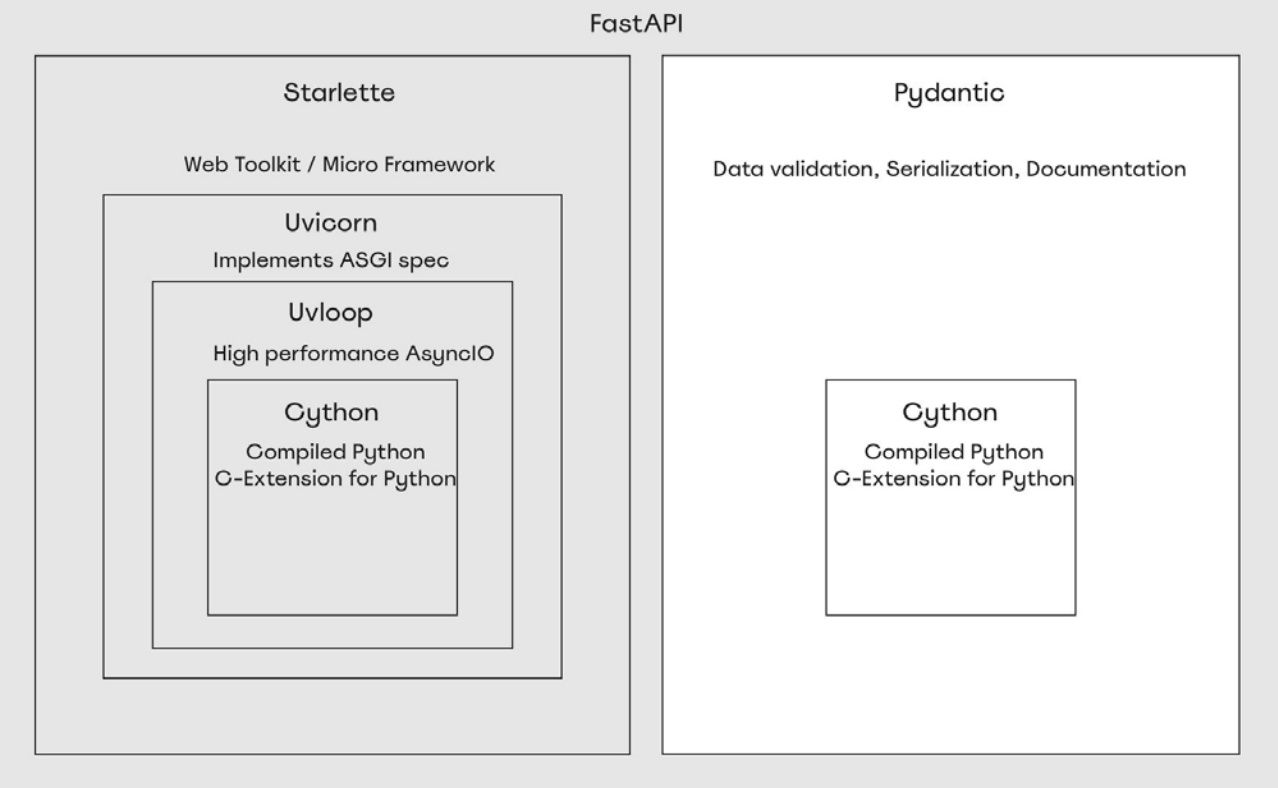
Ariadne

Tartiflette

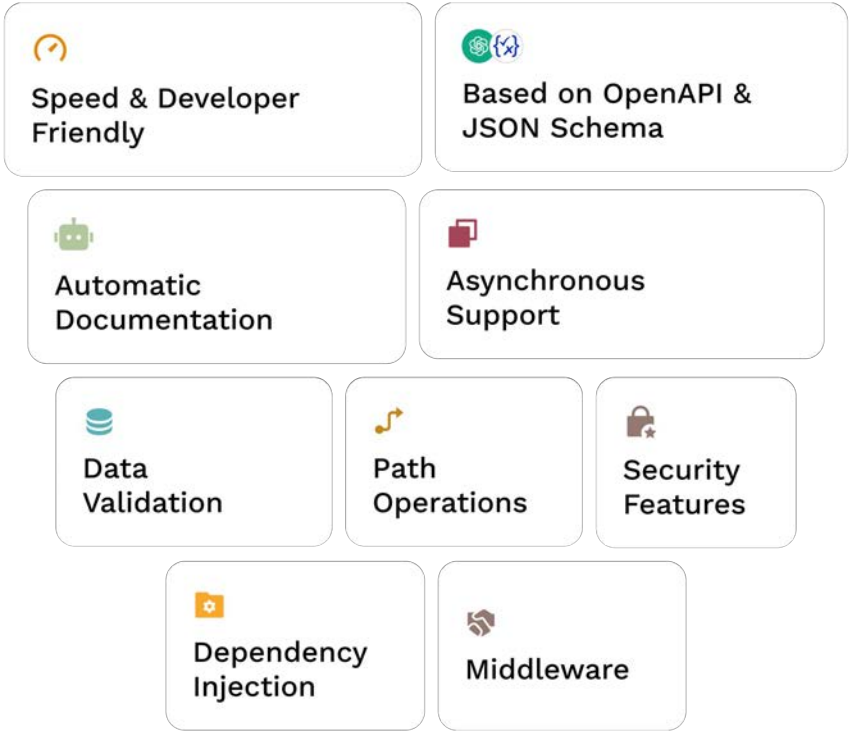
FastAPI

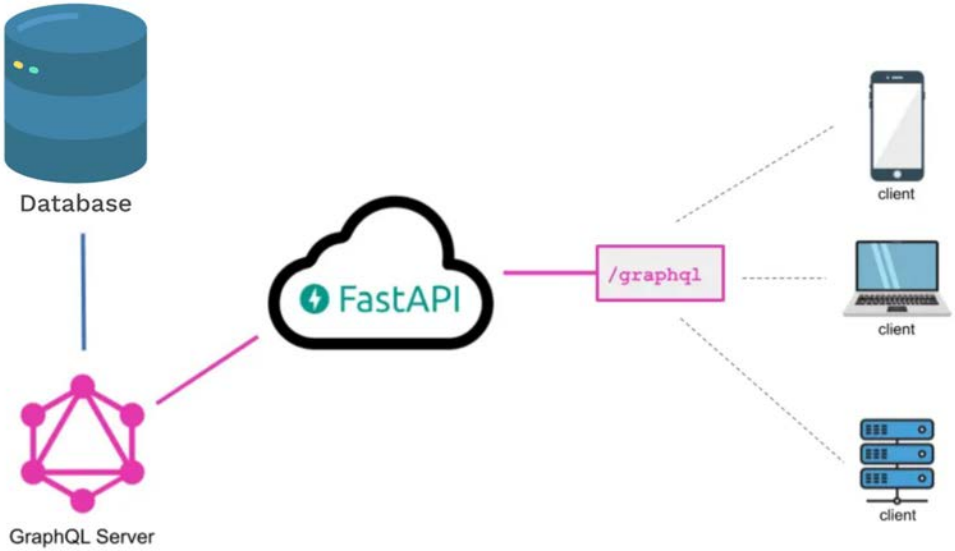
- High-performance Python API framework.
- Built on Starlette for a solid foundation.
- Uses Python type hints for better productivity and readability.

FastAPI - Core Components



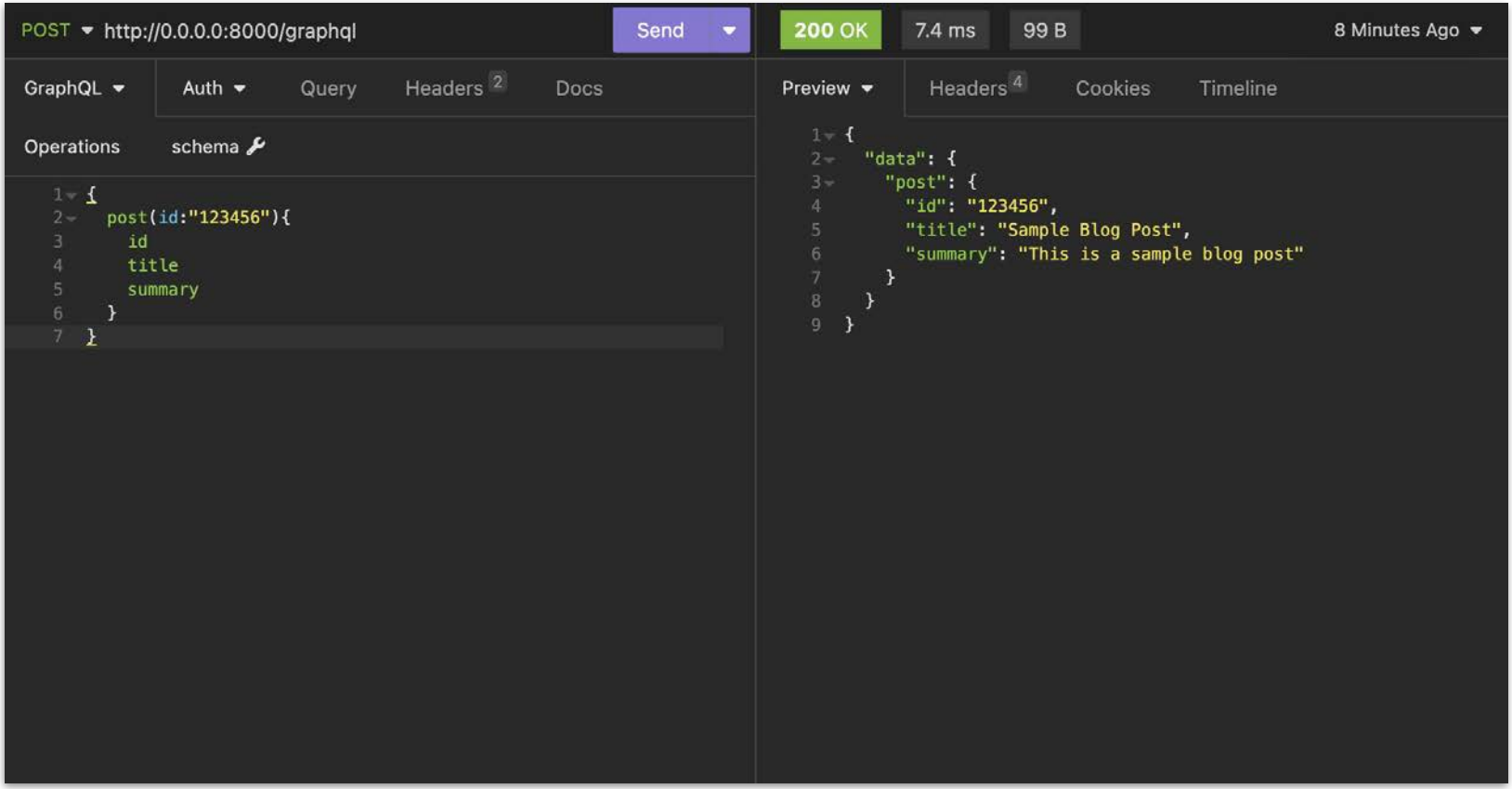
Features


Speed & Developer Friendly
Based on OpenAPI & JSON Schema
Automatic Documentation
Asynchronous Support
Data Validation
Path Operations
Security Features
Dependency Injection
Middleware


Database
FastAPI
/graphql
client
client
client
GraphQL Server

GraphQL with FastAPI

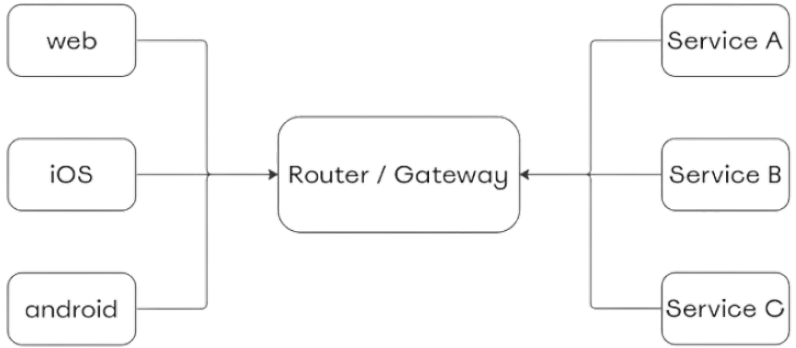
```python
import graphene
import uvicorn
from fastapi import FastAPI
from starlette_graphene3 import GraphQLApp


class PostType(graphene.ObjectType):
    id = graphene.ID(required=True, description="Unique identifier of the post")
    title = graphene.String(required=True, description="Title of the post")
    summary = graphene.String(required=True, description="Summary the post")


class Query(graphene.ObjectType):
    post = graphene.Field(PostType, id=graphene.ID(required=True))

    def resolve_post(self, info, id):
        return PostType(
            id=id,
            title="Sample Blog Post",
            summary="This is a sample blog post"
        )


graphql_schema = graphene.Schema(query=Query)

app = FastAPI()
app.add_route("/graphql", GraphQLApp(graphql_schema))

if __name__ == "__main__":
    uvicorn.run(app, host="0.0.0.0", port=8000)
```


POST http://0.0.0.0:8000/graphql
Send
200 OK
7.4 ms
99 B
8 Minutes Ago
GraphQL
Auth
Query
Headers 2
Docs
Preview
Headers 4
Cookies
Timeline
Operations
schema
{
post(id:"123456"){
id
title
summary
}
}
{
"data": {
"post": {
"id": "123456",
"title": "Sample Blog Post",
"summary": "This is a sample blog post"
}
}
}

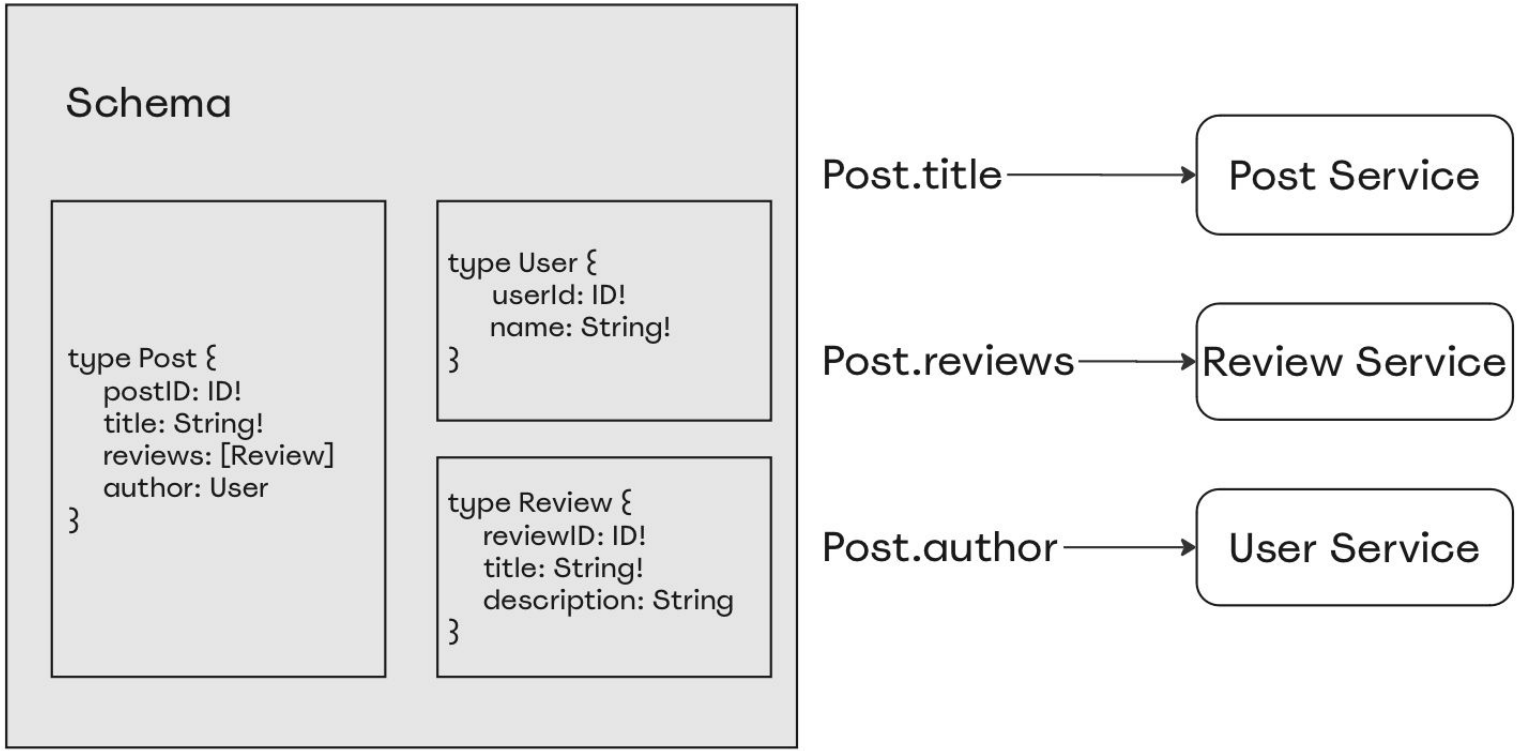
GraphQL Federation

GraphQL Federation combines multiple GraphQL APIs into a single, federated graph.

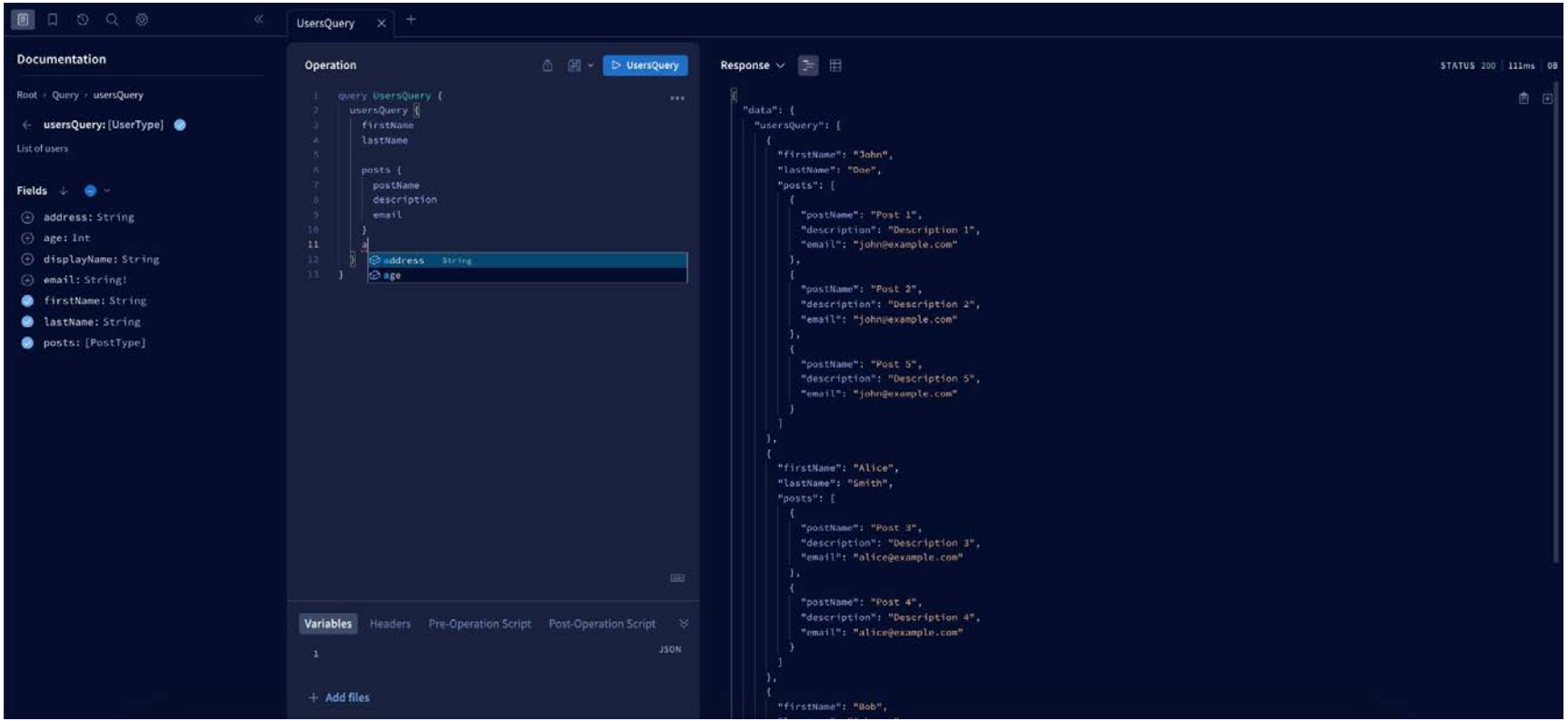
This federated graph enables clients to interact with multiple APIs through a single request.



Example


Schema
type Post {
postID: ID!
title: String!
reviews: [Review]
author: User
}
type User {
userId: ID!
name: String!
}
type Review {
reviewID: ID!
title: String!
description: String
}
Post.title
Post Service
Post.reviews
Review Service
Post.author
User Service

Example


UsersQuery
Documentation
Root › Query › usersQuery
usersQuery: [UserType]
List of users
Fields
address: String
age: Int
displayName: String
email: String!
firstName: String
lastName: String
posts: [PostType]
Operation
UsersQuery
query UsersQuery {
usersQuery {
firstName
lastName
posts {
postName
description
email
}
address String
age
Variables
Headers
Pre-Operation Script
Post-Operation Script
JSON
Add files
Response
STATUS 200 | 111ms | 0B
{
"data": {
"usersQuery": [
{
"firstName": "John",
"lastName": "Doe",
"posts": [
{
"postName": "Post 1",
"description": "Description 1",
"email": "john@example.com"
},
{
"postName": "Post 2",
"description": "Description 2",
"email": "john@example.com"
},
{
"postName": "Post 5",
"description": "Description 5",
"email": "john@example.com"
}
]
},
{
"firstName": "Alice",
"lastName": "Smith",
"posts": [
{
"postName": "Post 3",
"description": "Description 3",
"email": "alice@example.com"
},
{
"postName": "Post 4",
"description": "Description 4",
"email": "alice@example.com"
}
]
},
{
"firstName": "Bob",

CONF42

Thank you

abymjoseph

jayalekshmiks112