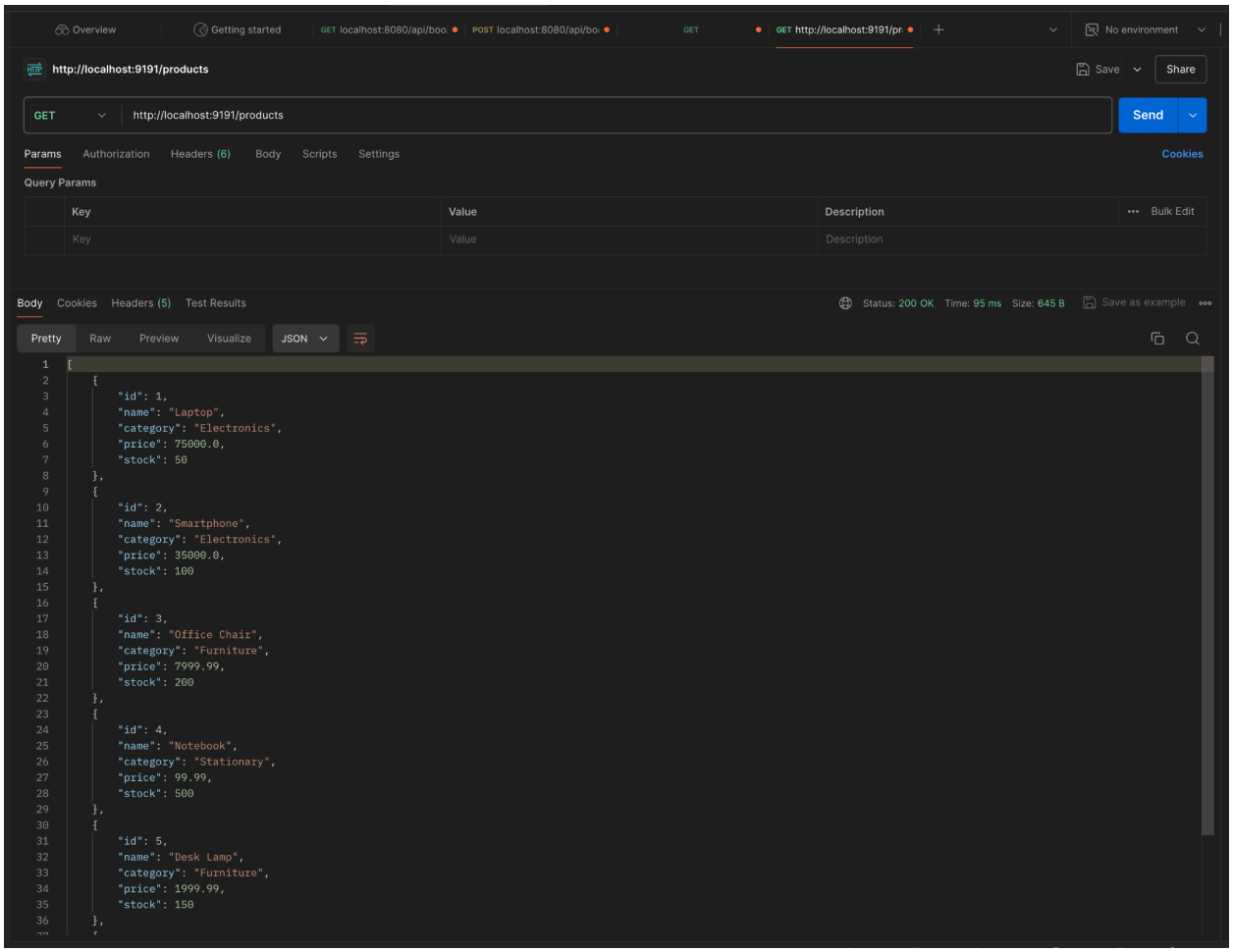
- GraphQL is a query language for your API
- Released by Meta
- GraphQL is a more efficient & powerful way to ask for specific data from an API.
- It allows clients to request exactly data they need and nothing more.
-

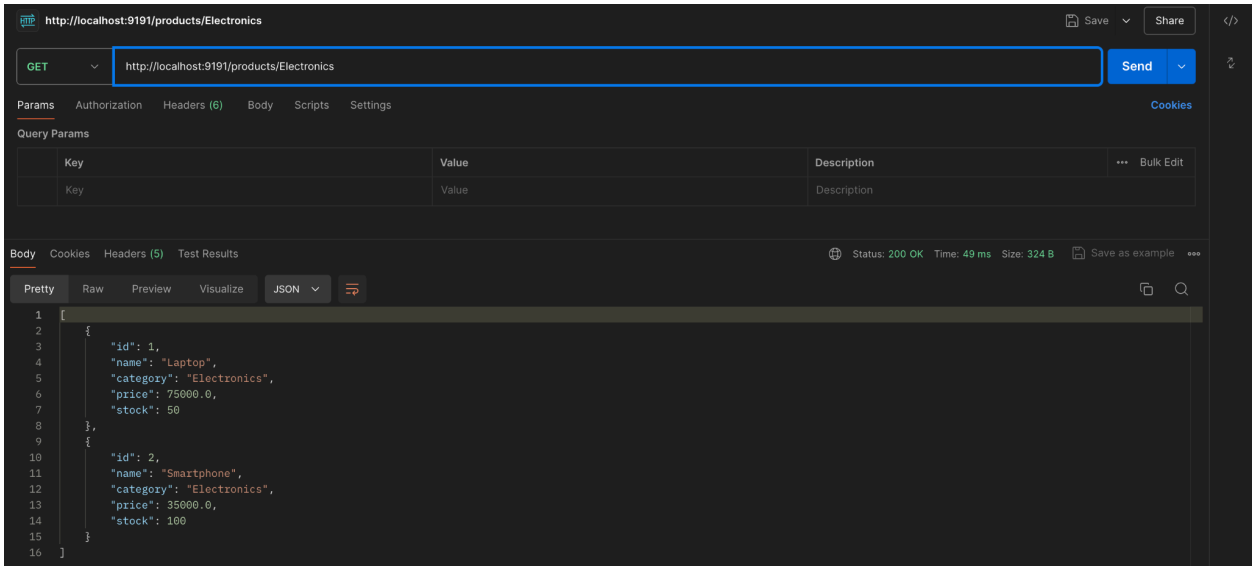
Real time use case

- Inventory service who expose an API with fields
    - ID
    - Name
    - Category
    - Supplier
    - Rating
    - Price
    - Stock
- We also have different client who want specific data from inventory service
    - Catalog-service
    - Sales team
    - Warehouse team
- Eg catalog-service wants only name and price field. Sales team want category, rating and stock information. Warehouse service wants name, category, supplier, stock information
- Different client different set of data to process work, but inventory service gives complete data to all teams which is not required
- Two solution
    - Ask inventory service to provide 3 different REST endpoints with desired field. Each endpoint for catalog, sales, warehouse team. Build response in such a way so that client can get only ID and name
        - Each and every client they can create their own DTO to map the response getting from inventory service
        - Catalog service creates DTO with name and ID
        - This is bad practice as client increase, inventory service creates more endpoints. A new client need different kind of response so we define different kind of response. This results in ***TIGHT COUPLING*** between client and server
        - If catalog now needs id and name, in future it adds stock and price, we have to create new DTO
    - GraphQL
        - Being a client you need to specify what field you need as a response in form of query
        - Then graphql will extract those field for you
        - For future as well define queries and graphql will handle those queries
- Using Spring boot 3

Code

- Problem with RESTApi
- Convert RESTapi to graphQL api
- In a resources folder there is graphql folder you can define request and response
- 

- We don’t want this, if the inventory service will expose all fields, but the client is not interested in all responses. Catalog services only need name and price. But current API service is giving all field as response
- So we convert to graphql query so if I need category and price I will get that value. ***WE will perform this usecase***

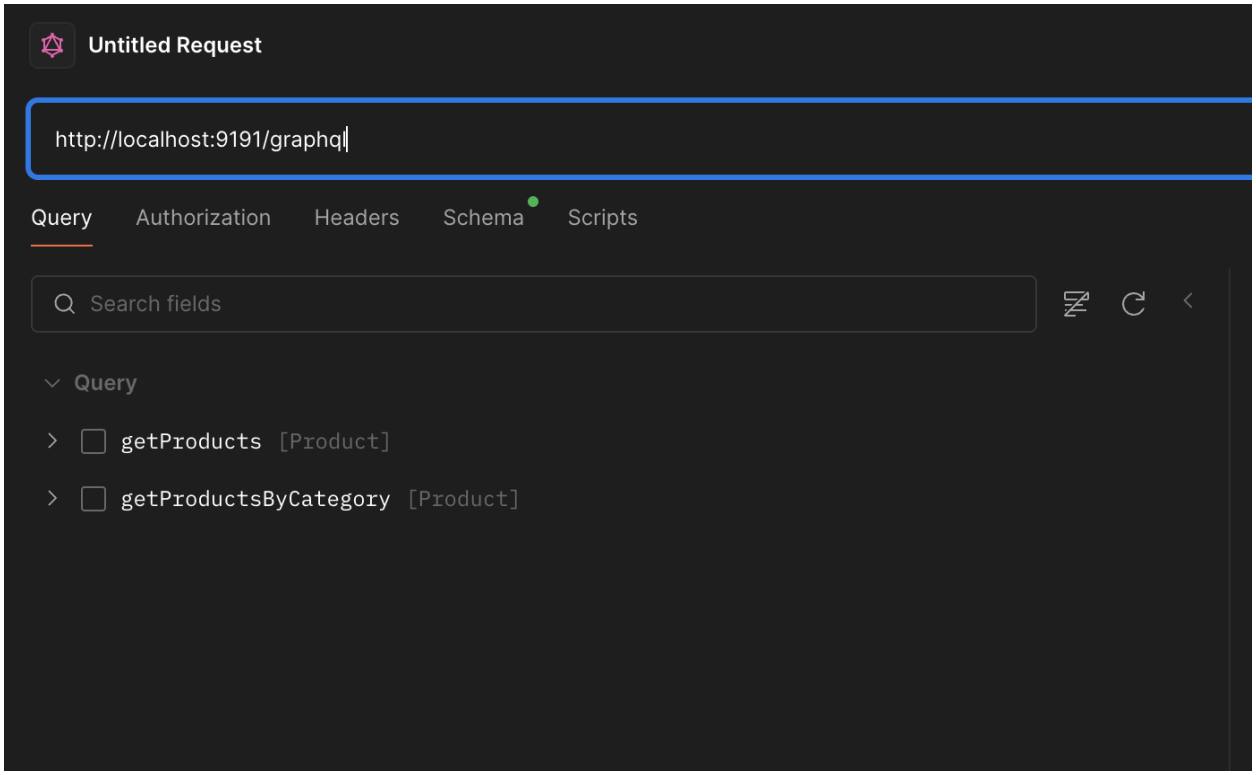
- 

- We are getting all fields, that's usual response we get from REST API
- ***IN OUR USE CASE WE DON'T WANT IT***

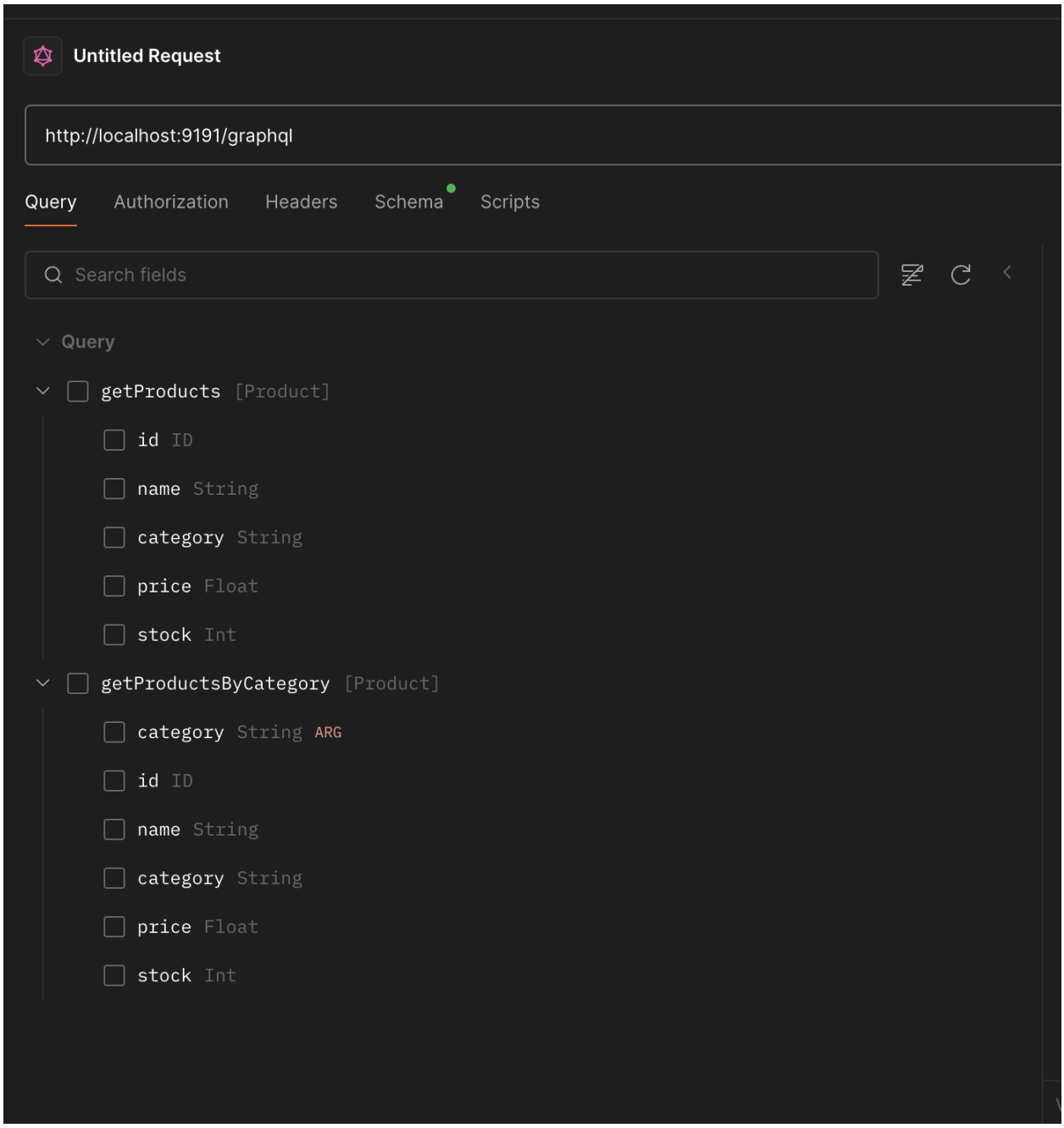
HOW TO CONVERT REST TO GraphQL

- Define a grpahql schemas for you field
- Then play with annotations
- You need to define Query Mapping instead of getmapping.
- Define @controller instead of restcontroller
- If we are not specifying URL how can I access it. This is not rest api. So we want to access it using query
- We want to define fields defined in product.java file as a type in schema.graphql so that you give and category of type
- ***Define method name as a query***

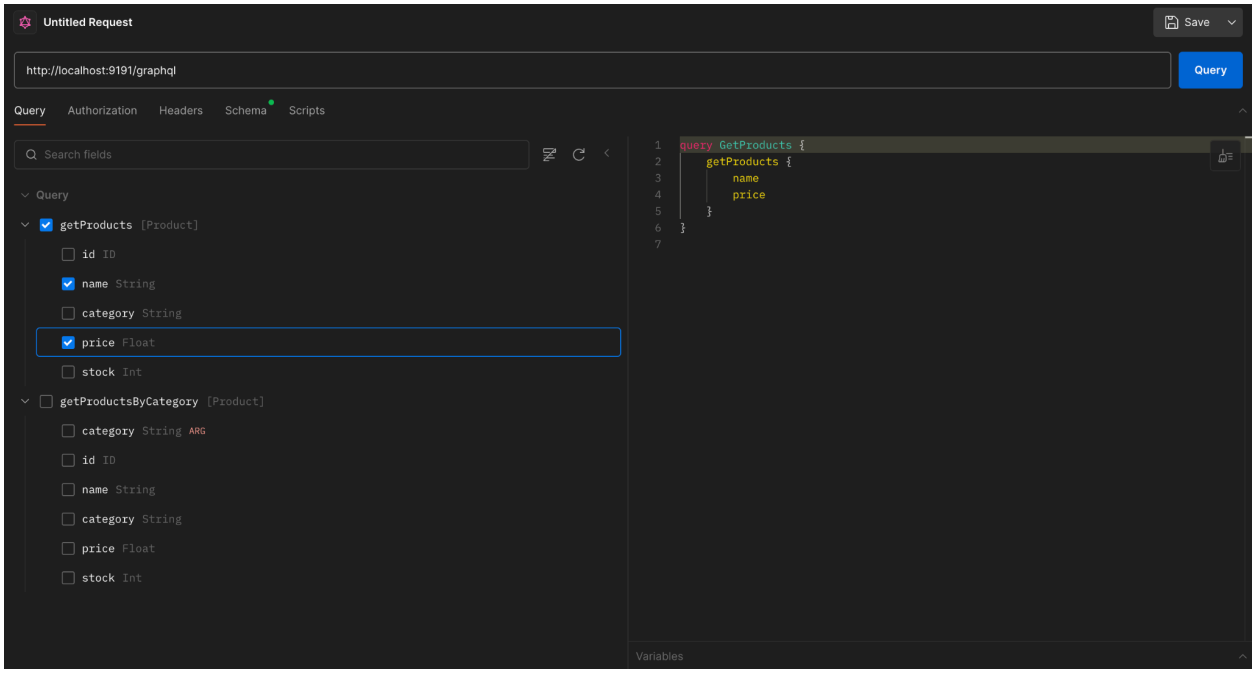
- File extension should be grpahqls

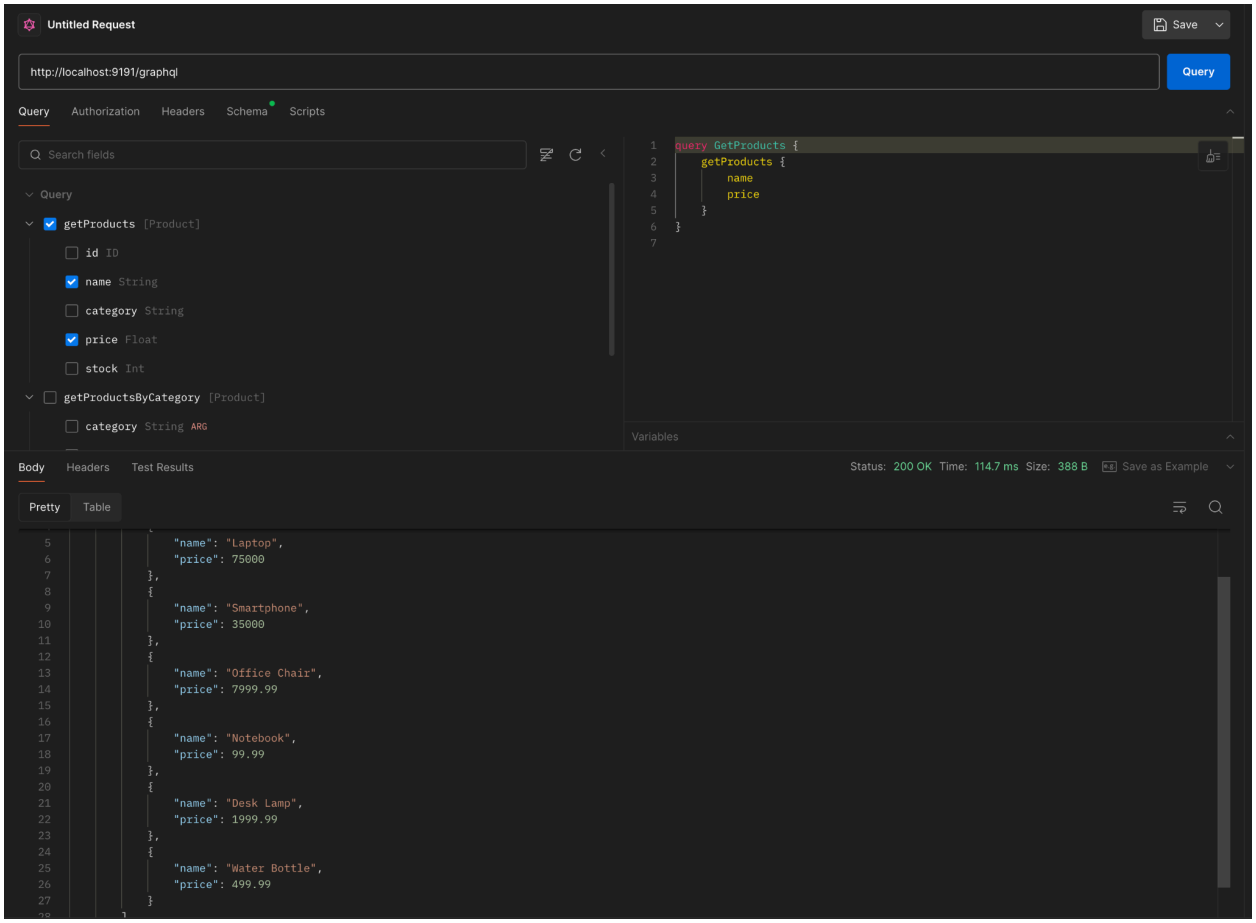


- Before entering it suggests the endpoints

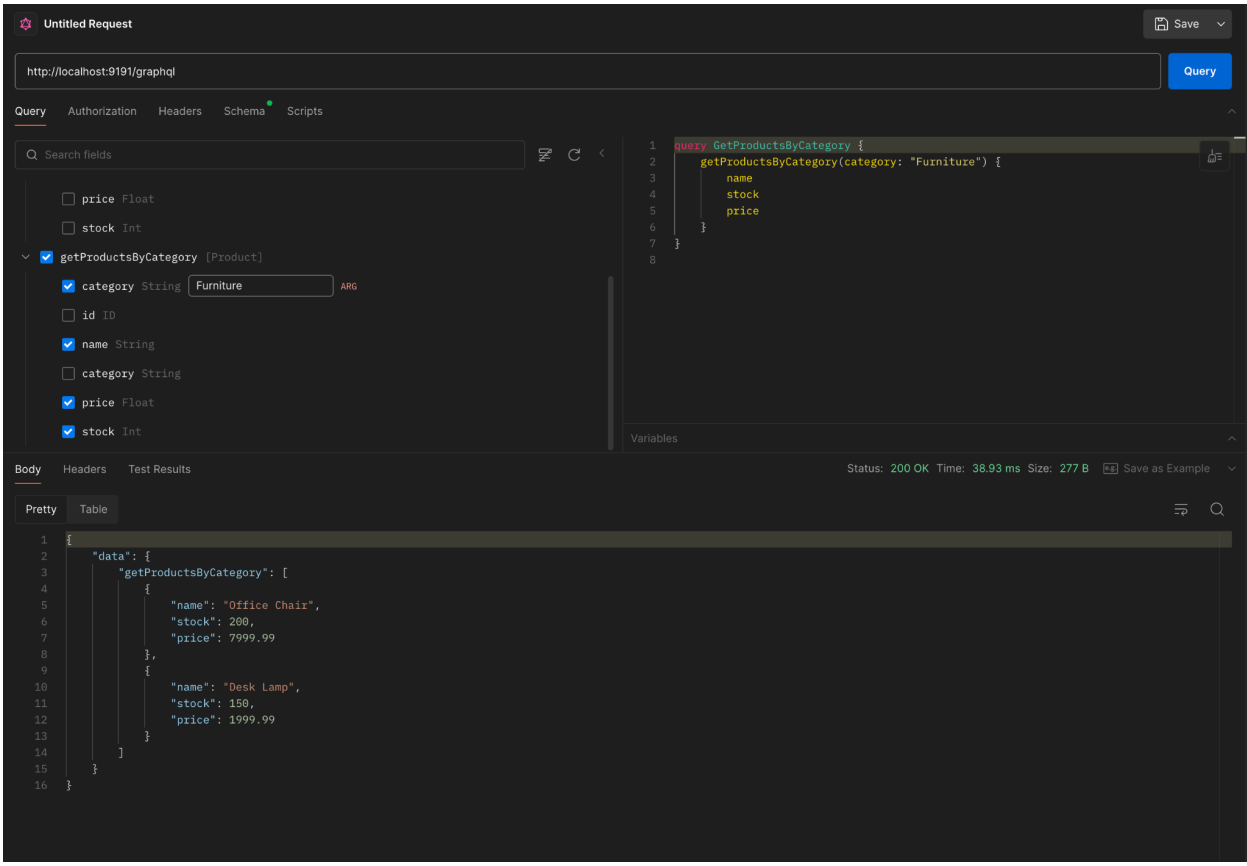
- 
Untitled Request
http://localhost:9191/graphql
Query
Authorization
Headers
Schema
Scripts
Search fields
Query
getProducts [Product]
id ID
name String
category String
price Float
stock Int
getProductsByCategory [Product]
category String ARG
id ID
name String
category String
price Float
stock Int

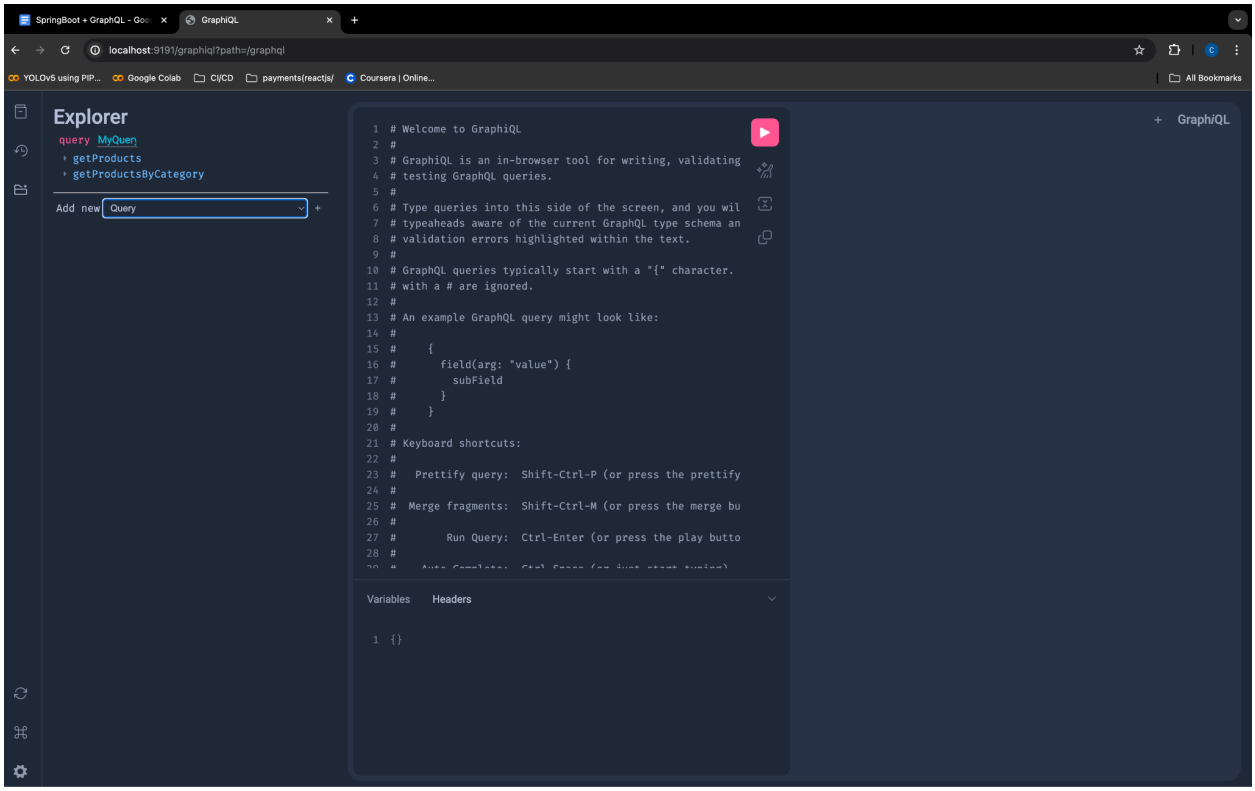
- We get options to select fields we want

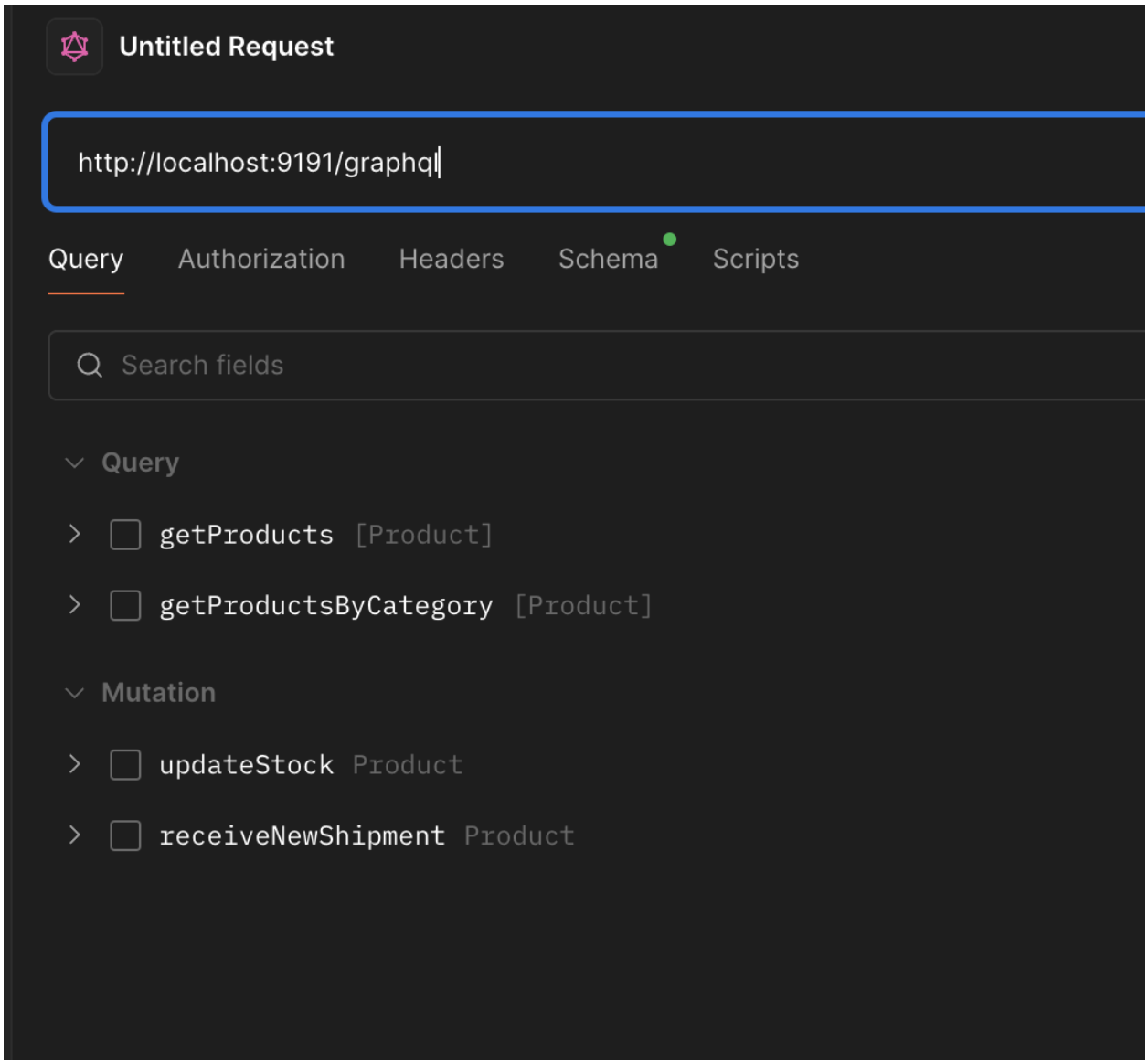




-

- 

- Similar to swagger dashboard for REST API, we can have graphql dashboard
- 

- Graphql dashboard
- We did GET operation using graphql, but we can also do post
- To do create and update use mutation
- For create update and delete use Mutation Mapping annotation
- 

-

Untitled Request

Save

http://localhost:9191/graphql

Query

Query Authorization Headers Schema Scripts

Search fields

getProductsByCategory [Product]

Mutation

updateStock Product

receiveNewShipment Product

id ID 5 ARG

quantity Int 200 ARG

id ID

name String

category String

price Float

stock Int

```
mutation ReceiveNewShipment {
    receiveNewShipment(id: "5", quantity: 200) {
        stock
        name
        category
        price
    }
}
```

Variables

Body Headers Test Results

Status: 200 OK Time: 26.17 ms Size: 243 B Save as Example

Pretty Table

```
{
    "data": {
        "receiveNewShipment": {
            "stock": 350,
            "name": "Desk Lamp",
            "category": "Furniture",
            "price": 1999.99
        }
    }
}
```

Untitled Request

http://localhost:9191/graphql

Query Authorization Headers Schema Scripts

Search fields

getProductsByCategory [Product]

Mutation

updateStock Product

id ID 1 ARG

stock Int 500 ARG

id ID

name String

category String

price Float

stock Int

receiveNewShipment Product

```
mutation UpdateStock {
    updateStock(id: "1", stock: 500) {
        stock
        id
    }
}
```

Variables

Body Headers Test Results

Status: 200 OK Time: 100.96

Pretty Table

```
{
    "data": {
        "updateStock": {
            "stock": 500,
            "id": "1"
        }
    }
}
```

100% 34:1

Result Grid Filter Rows: Search Edit:

id	category	name	price	stock
1	Electronics	Laptop	75000	500
2	Electronics	Smartphone	35000	100
3	Furniture	Office Chair	7999.99	200
4	Stationary	Notebook	99.99	500
5	Furniture	Desk Lamp	1999.99	150
6	Accessories	Water Bottle	499.99	300
NULL	NULL	NULL	NULL	NULL

100% 34:1

Result Grid Filter Rows: Search Edit:

id	category	name	price	stock
1	Electronics	Laptop	75000	500
2	Electronics	Smartphone	35000	100
3	Furniture	Office Chair	7999.99	200
4	Stationary	Notebook	99.99	500
5	Furniture	Desk Lamp	1999.99	150
6	Accessories	Water Bottle	499.99	300
NULL	NULL	NULL	NULL	NULL

- 100% 34:1

  Result Grid Filter Rows: Search Edit:

id	category	name	price	stock
1	Electronics	Laptop	75000	500
2	Electronics	Smartphone	35000	100
3	Furniture	Office Chair	7999.99	200
4	Stationary	Notebook	99.99	500
5	Furniture	Desk Lamp	1999.99	350
6	Accessories	Water Bottle	499.99	300
NULL	NULL	NULL	NULL	NULL

- AIM : crud operations using graphql and spring boot
- Server and client are loosely coupled with each other
- This will always be recommended if you multiple client and they need different dataset as response
- Rather than creating separate DTO or endpoint it is good to use graphQL