TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE

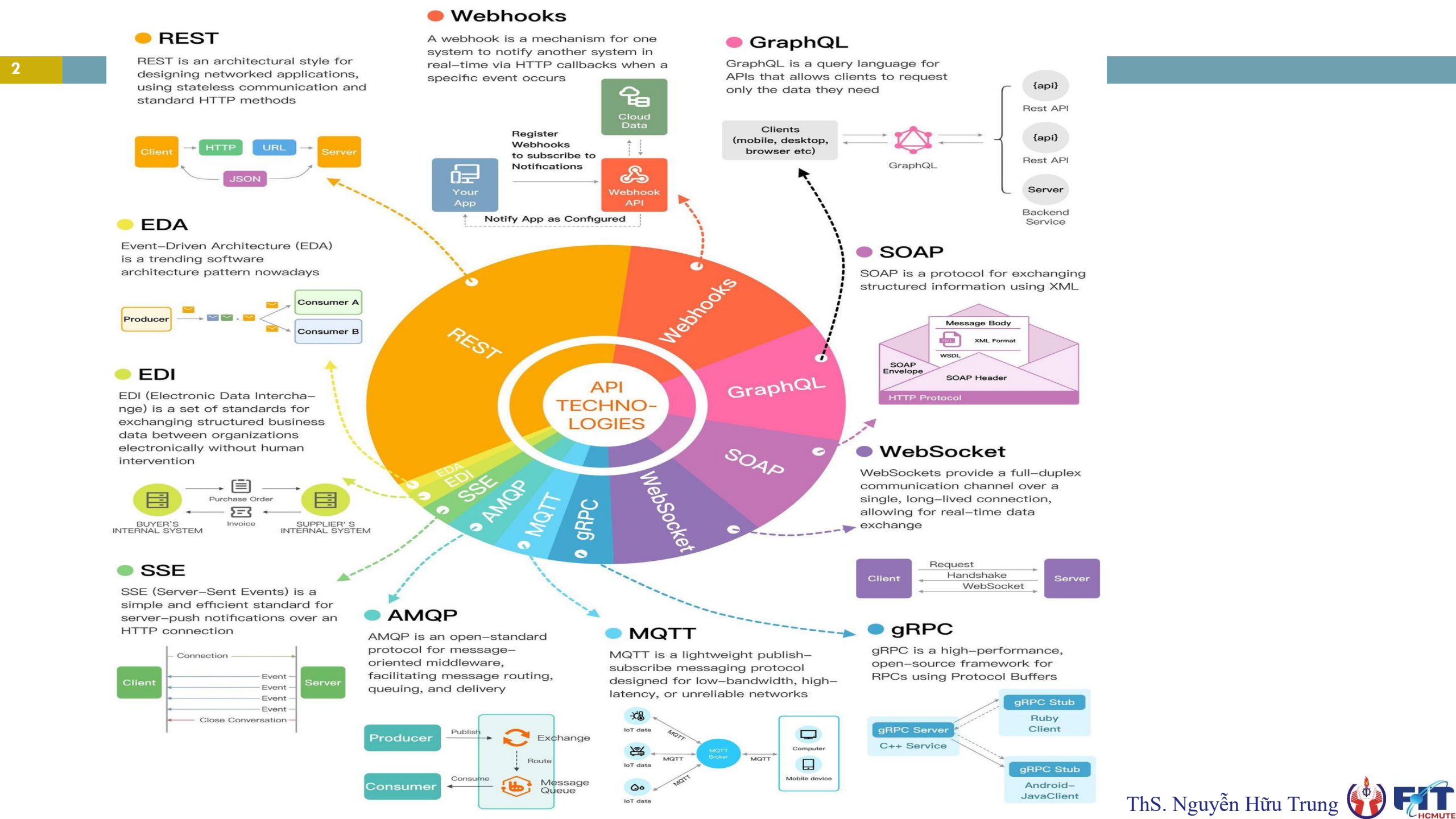
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM

LẬP TRÌNH WEB (WEBPR330479)

GraphQL & RestFul API trong Spring boot

THS. NGUYỄN HỮU TRUNG

PR33 WEB 0479


REST
REST is an architectural style for designing networked applications, using stateless communication and standard HTTP methods
Client
HTTP
URL
Server
JSON
Webhooks
A webhook is a mechanism for one system to notify another system in real–time via HTTP callbacks when a specific event occurs
Cloud Data
Register Webhooks to subscribe to Notifications
Your App
Webhook API
Notify App as Configured
GraphQL
GraphQL is a query language for APIs that allows clients to request only the data they need
Clients (mobile, desktop, browser etc)
GraphQL
{api}
Rest API
{api}
Rest API
Server
Backend Service
EDA
Event–Driven Architecture (EDA) is a trending software architecture pattern nowadays
Producer
Consumer A
Consumer B
SOAP
SOAP is a protocol for exchanging structured information using XML
Message Body
XML Format
WSDL
SOAP Envelope
SOAP Header
HTTP Protocol
EDI
EDI (Electronic Data Intercha–nge) is a set of standards for exchanging structured business data between organizations electronically without human intervention
Purchase Order
Invoice
BUYER'S INTERNAL SYSTEM
SUPPLIER' S INTERNAL SYSTEM
API TECHNO–LOGIES
REST
Webhooks
GraphQL
SOAP
WebSocket
gRPC
MQTT
AMQP
SSE
EDA
EDI
WebSocket
WebSockets provide a full–duplex communication channel over a single, long–lived connection, allowing for real–time data exchange
Client
Request
Handshake
WebSocket
Server
SSE
SSE (Server–Sent Events) is a simple and efficient standard for server–push notifications over an HTTP connection
Client
Connection
Event
Event
Event
Event
Close Conversation
Server
AMQP
AMQP is an open–standard protocol for message–oriented middleware, facilitating message routing, queuing, and delivery
Producer
Publish
Exchange
Route
Consumer
Consume
Message Queue
MQTT
MQTT is a lightweight publish–subscribe messaging protocol designed for low–bandwidth, high–latency, or unreliable networks
IoT data
IoT data
IoT data
MQTT
MQTT
MQTT
MQTT
MQTT Broker
Computer
Mobile device
gRPC
gRPC is a high–performance, open–source framework for RPCs using Protocol Buffers
gRPC Server
C++ Service
gRPC Stub
Ruby Client
gRPC Stub
Android–JavaClient

Giới thiệu GraphQL?

GraphQL

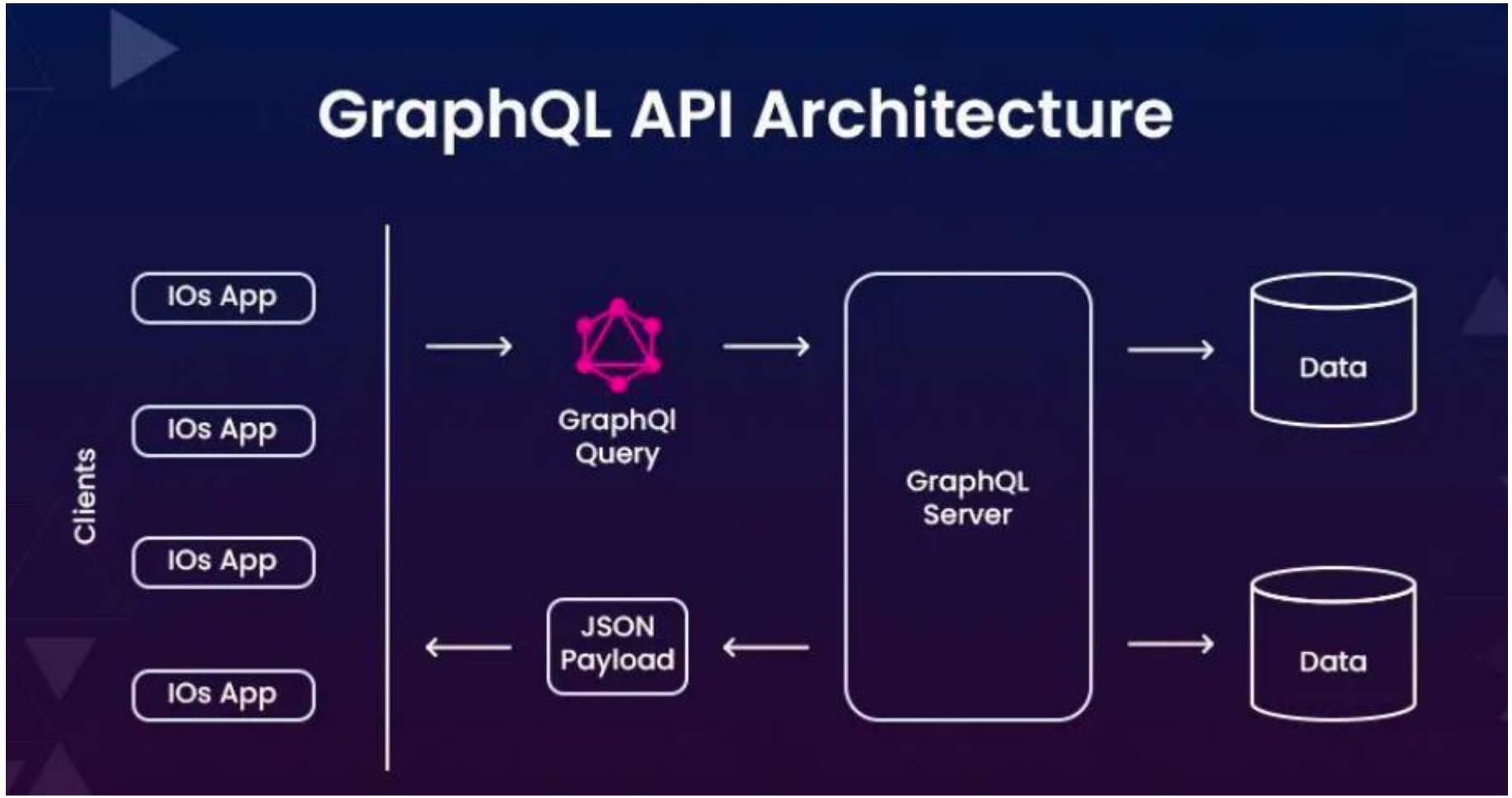
- Là ngôn ngữ thao tác và truy vấn dữ liệu nguồn mở cho API
- Cung cấp cách thức dễ dàng để request chính xác và vừa đủ.
- Được Facebook phát triển vào năm 2012.

Đặc trưng của GraphQL

- Cho phép client xác định chính xác những dữ liệu gì họ cần
- GraphQL làm cho việc tổng hợp dữ liệu từ nhiều nguồn dễ dàng hơn
- Sử dụng một type system để khai báo dữ liệu.

https://graphql.org/

Giới thiệu GraphQL?

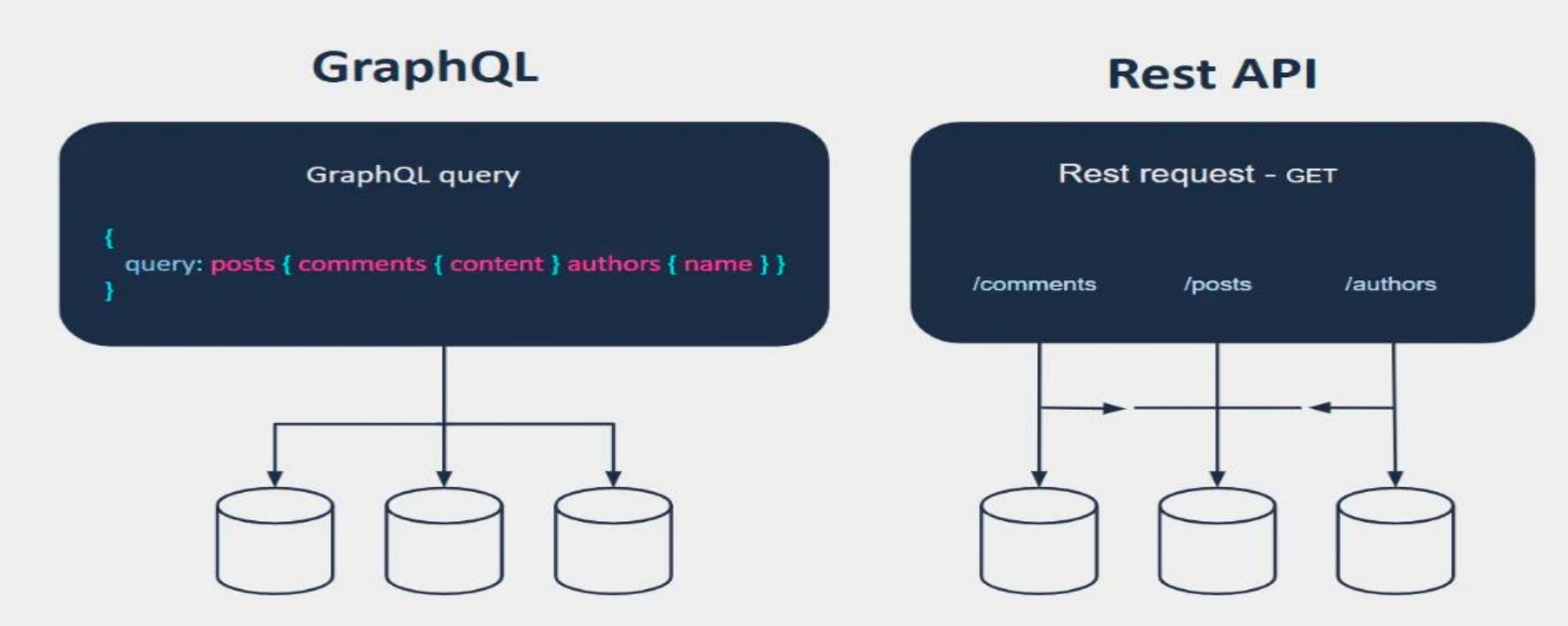


Giới thiệu GraphQL?

- **Trả về chính xác những gì bạn gửi request**
  - Khi gửi 1 request GraphQL đến API, thì sẽ nhận được chính xác những gì được yêu cầu trong request, không hơn không kém.
  - Các truy vấn GraphQL luôn trả về kết quả có thể dự đoán được.
  - Các ứng dụng sử dụng GraphQL rất nhanh và ổn định vì GraphQL kiểm soát dữ liệu mà nó nhận được chứ không phải máy chủ.

Giới thiệu GraphQL?

- **Nhận nhiều dữ liệu trong một request duy nhất**
  - Các câu query GraphQL không chỉ có thể truy xuất các thuộc tính của một dữ liệu mà còn làm việc trơn tru với các đối tượng khác.
  - GraphQL có thể lấy tất cả dữ liệu mà ứng dụng của bạn cần trong một request duy nhất.
  - Các ứng dụng sử dụng GraphQL có tốc độ xử lý rất nhanh ngay cả trên các kết nối chậm (di động, IoTs).



Giới thiệu GraphQL?

- **Mô tả những gì có thể với một type system**
  - API GraphQL được tổ chức theo **type** và **fields**, truy cập toàn bộ dữ liệu của bạn từ một **endpoint** duy nhất.
  - GraphQL sử dụng các **types** để đảm bảo ứng dụng của bạn chỉ request những gì có thể, ngoài ra GraphQL còn giải thích các lỗi 1 cách rất dễ hiểu.
  - Ứng dụng của bạn có thể sử dụng các **types** để tránh viết code phân tích cú pháp thủ công.

application.properties | product.graphqls | account.graphqls

```
type Account {
    id: ID!
    email: String!
    fullName: String!
    images: String
    address: Int
    password: String
    products: [Product]
}

type Query {
    accounts: [Account]
    accountById(id:ID!): Account
    accountsByFullName(fullName: String!): [Account]
}

type Mutation {
    createAccount(account: AccountInput): Account
    updateAccount(account: AccountInput): Account
    deleteAccountById(id: ID!): Account
}
```

Những điểm tương đồng giữa GraphQL và REST là gì?

- REST và GraphQL cho phép bạn tạo, sửa đổi, cập nhật và xóa dữ liệu trên một ứng dụng, dịch vụ hoặc mô-đun riêng biệt thông qua API.
- Một máy khách gửi yêu cầu API đến một điểm cuối hoặc nhiều điểm cuối trên một máy chủ
- Máy chủ cung cấp phản hồi có chứa dữ liệu, trạng thái dữ liệu hoặc mã lỗi
- **Kiến trúc**: đều không có trạng thái, đều sử dụng mô hình máy khách – máy chủ.
- **Thiết kế dựa trên tài nguyên**: Ví dụ: hãy xem xét một API mạng xã hội nơi người dùng tạo và quản lý bài đăng. Trong API dựa trên tài nguyên, bài đăng sẽ là một tài nguyên. Nó có mã định danh duy nhất riêng, ví dụ: */posts/1234*. Đồng thời, nó có một tập hợp các thao tác, như ***GET*** để truy xuất bài đăng trong REST hoặc ***query*** (truy vấn) để truy xuất bài đăng trong GraphQL.
- **Trao đổi dữ liệu**: đều hỗ trợ các định dạng dữ liệu JSON, đều hỗ trợ khả năng ghi vào bộ nhớ đệm.
- **Tính trung lập về ngôn ngữ và cơ sở dữ liệu:** Tính trung lập về ngôn ngữ và cơ sở dữ liệu**.**

GraphQL và REST khác nhau ở điểm nào?

	REST	GraphQL
Đó là gì?	REST là một tập hợp các quy tắc định hình quá trình trao đổi dữ liệu có cấu trúc giữa một máy khách và một máy chủ.	GraphQL là một ngôn ngữ truy vấn, kiểu kiến trúc và tập hợp các công cụ để tạo và thao tác với các API.
Trường hợp sử dụng phù hợp nhất	REST phù hợp với các nguồn dữ liệu đơn giản trong đó tài nguyên được xác định rõ ràng.	GraphQL phù hợp với các nguồn dữ liệu lớn, phức tạp và liên quan đến nhau.
Truy cập dữ liệu	REST có nhiều điểm cuối dưới dạng URL để xác định tài nguyên.	GraphQL có một điểm cuối URL duy nhất.
Dữ liệu trả về	REST trả về dữ liệu theo một cấu trúc cố định do máy chủ xác định.	GraphQL trả về dữ liệu theo một cấu trúc linh hoạt do máy khách xác định.
Cách dữ liệu được cấu trúc và xác định	Dữ liệu REST định kiểu yếu. Vì vậy, máy khách phải quyết định cách diễn giải dữ liệu đã định dạng khi dữ liệu được trả về.	Dữ liệu GraphQL định kiểu mạnh. Vì vậy, máy khách nhận dữ liệu theo định dạng đã xác định trước và cả hai bên cùng hiểu.
Kiểm tra lỗi	Với REST, khách hàng phải kiểm tra xem dữ liệu trả về có hợp lệ hay không. 200, 500, 401	Với GraphQL, các yêu cầu không hợp lệ thường bị cấu trúc lược đồ từ chối. Điều này làm xuất hiện một thông báo lỗi được tự động tạo. 200, error

GraphQL cố gắng khắc phục những hạn chế nào của REST?

- **Trao đổi dữ liệu có cấu trúc cố định**: API REST đòi hỏi các yêu cầu máy khách phải tuân theo một cấu trúc cố định để nhận tài nguyên. Cấu trúc cứng nhắc này rất dễ sử dụng, nhưng không phải lúc nào cũng là phương tiện hiệu quả nhất để trao đổi chính xác dữ liệu cần.
- **Lấy thừa dữ liệu và lấy thiếu dữ liệu**: API REST luôn trả về một tập dữ liệu toàn bộ. Ví dụ: từ một đối tượng *person* (người) trong API REST, sẽ nhận được tên, ngày sinh, địa chỉ và số điện thoại của người đó. Thì sẽ nhận được tất cả dữ liệu này ngay cả khi chỉ cần lấy số điện thoại. Tương tự, nếu muốn biết số điện thoại của một người và lần mua gần đây nhất, bạn sẽ cần nhiều yêu cầu API REST. URL */person* sẽ trả về số điện thoại và URL */purchase* sẽ trả về lịch sử mua hàng.

Thời điểm nên sử dụng GraphQL hay REST

- Có thể sử dụng các API GraphQL và REST thay thế cho nhau. Tuy nhiên, trong một số trường hợp sẽ có một API phù hợp hơn API kia. Ví dụ: GraphQL có thể là lựa chọn tốt hơn nếu bạn có những cân nhắc sau:
  - Bạn có băng thông giới hạn và bạn muốn giảm thiểu số lượng yêu cầu và phản hồi
  - Bạn có nhiều nguồn dữ liệu và bạn muốn kết hợp chúng tại một điểm cuối
  - Bạn có các yêu cầu máy khách có sự khác biệt đáng kể và bạn muốn có các phản hồi rất khác nhau

GraphQL và Rest trong Spring boot

REST

```java
@RestController
@RequestMapping("api")
public class ProductRestApiController {
    @Autowired
    private ProductService productService;
    @GetMapping("/product")
    public ResponseEntity<List<Product>> listAll(){
        List<Product> list= productService.findAll();
        if(list.isEmpty()) {
            return new ResponseEntity<List<Product>>(HttpStatus.NO_CONTENT);
        }
        return new ResponseEntity<List<Product>>(list, HttpStatus.OK);
    }
    @GetMapping("/product/{id}")
    public Product find(@PathVariable long id) {
        Optional<Product> pro = productService.findById(id);
        if(pro == null) {
            ResponseEntity.notFound().build();
        }
        return pro.get();
    }
    @PostMapping("/product")
    public Product save(@Valid @RequestBody Product category) {
        return productService.save(category);

    }
    @PutMapping("/product/{id}")
    public ResponseEntity<Product> update(@PathVariable Long id,
                                          @Valid @RequestBody Product proForm) {
        Optional<Product> category = productService.findById(id);
        if(category == null) {
            return ResponseEntity.notFound().build();
        }
       Product updated = productService.save(proForm);
        return ResponseEntity.ok(updated);
    }
```

GRAPHQL

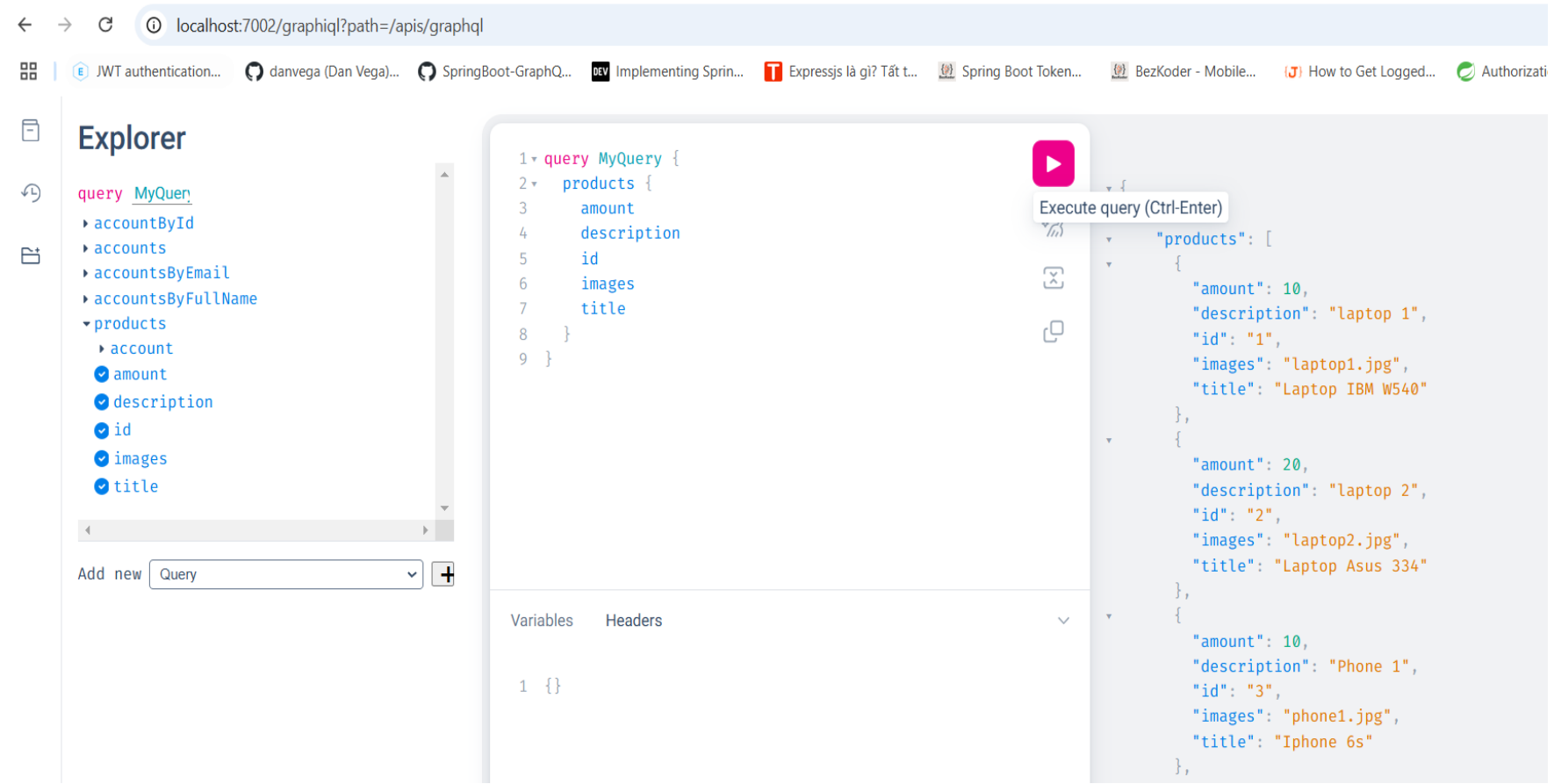
```java
@Controller
public class ProductController {
    @Autowired
    private ProductService productService;

    @QueryMapping
    public List<Product> products(){
        return productService.findAll();
    }
    @QueryMapping
    public Optional<Product> productById(@Argument Long id){
        return productService.findById(id);
    }

    @MutationMapping
    public Product createProduct(@Argument ProductModel product) {
        return productService.createProduct(product);
    }
}
```

localhost:7002/api/product

JWT authentication... danvega (Dan Veg

Pretty-print


```
[
  {
    "id": 1,
    "title": "Laptop IBM W540",
    "description": "laptop 1",
    "images": "laptop1.jpg",
    "amount": 10
  },
  {
    "id": 2,
    "title": "Laptop Asus 334",
    "description": "laptop 2",
    "images": "laptop2.jpg",
    "amount": 20
  },
  {
    "id": 3,
    "title": "Iphone 6s",
    "description": "Phone 1",
    "images": "phone1.jpg",
    "amount": 10
  },
  {
    "id": 4,
    "title": "Samsung s8",
    "description": "Phone 2",
    "images": "phone2.jpg",
    "amount": 10
  },
  {
    "id": 5,
    "title": "Máy giặt LG 610",
    "description": "Máy giặt",
    "images": null,
    "amount": 1
  }
]
```

localhost:7002/api/product/1

JWT authentication... danvega (Dan Vega

Pretty-print


```
{
  "id": 1,
  "title": "Laptop IBM W540",
  "description": "laptop 1",
  "images": "laptop1.jpg",
  "amount": 10
}
```


localhost:7002/graphiql?path=/apis/graphql

JWT authentication... danvega (Dan Vega)... SpringBoot-GraphQ... Implementing Sprin... Expressjs là gì? Tất t... Spring Boot Token... BezKoder - Mobile... How to Get Logged... Authorization Mig

Explorer

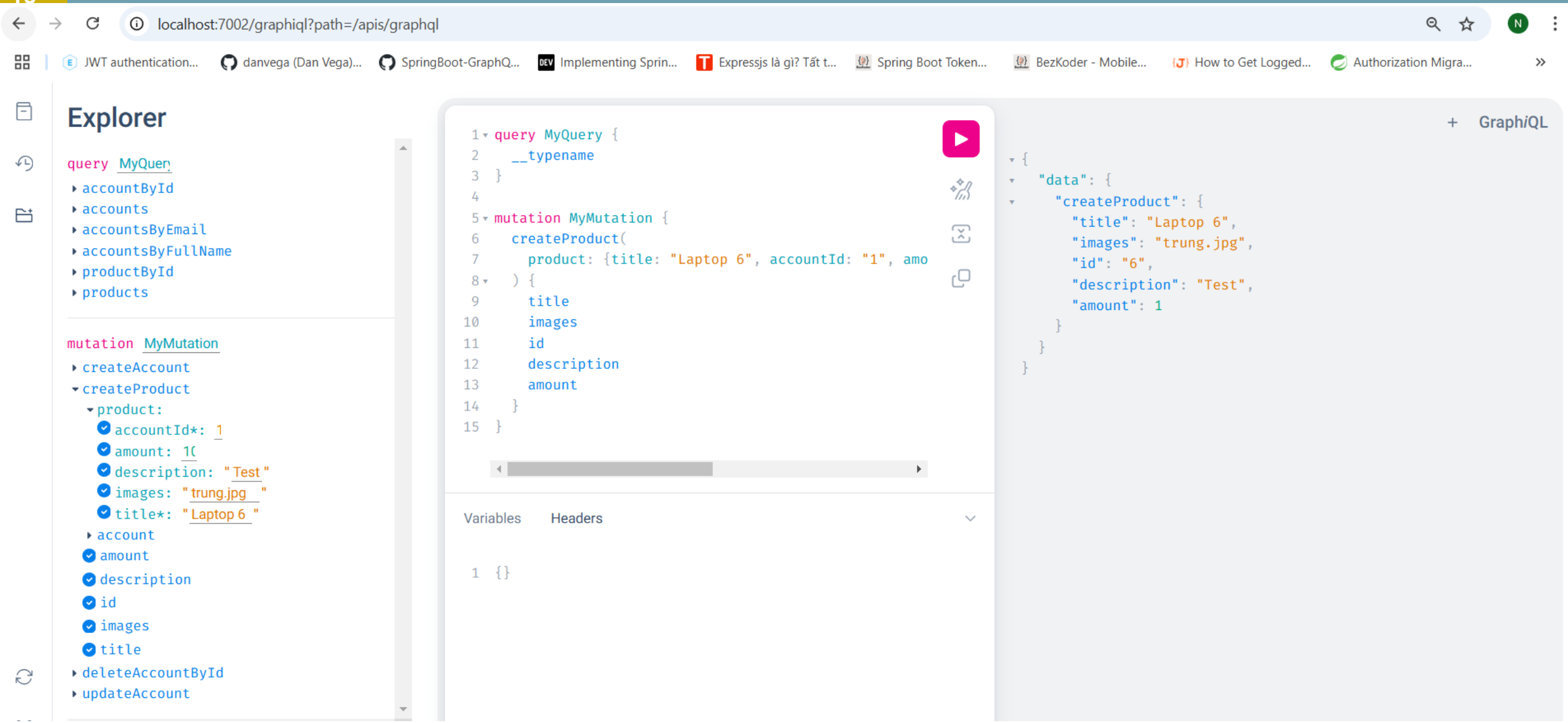
query MyQuery

- accountById
- accounts
- accountsByEmail
- accountsByFullName
- productById
  - id*: 1
  - account
    - address
    - email
    - fullName
    - id
    - images
    - password
    - products

```
query MyQuery {
  productById(id: "1") {
    amount
    description
    images
    title
    account {
      fullName
      email
      images
    }
  }
}
```


```
{
  "data": {
    "productById": {
      "amount": 10,
      "description": "laptop 1",
      "images": "laptop1.jpg",
      "title": "Laptop IBM W540",
      "account": {
        "fullName": "Nguyễn Hữu Trung",
        "email": "trungnh@hcmute.edu.vn",
        "images": "trung.jpg"
      }
    }
  }
}
```

Tạo product


localhost:7002/graphiql?path=/apis/graphql
JWT authentication...
danvega (Dan Vega)...
SpringBoot-GraphQ...
Implementing Sprin...
Expressjs là gì? Tất t...
Spring Boot Token...
BezKoder - Mobile...
How to Get Logged...
Authorization Migra...
Explorer
query MyQuery
accountById
accounts
accountsByEmail
accountsByFullName
productById
products
mutation MyMutation
createAccount
createProduct
product:
accountId*: 1
amount: 10
description: "Test"
images: "trung.jpg"
title*: "Laptop 6"
account
amount
description
id
images
title
deleteAccountById
updateAccount
1 query MyQuery {
2 __typename
3 }
4
5 mutation MyMutation {
6 createProduct(
7 product: {title: "Laptop 6", accountId: "1", amo
8) {
9 title
10 images
11 id
12 description
13 amount
14 }
15 }
Variables
Headers
1 {}
GraphiQL
{
"data": {
"createProduct": {
"title": "Laptop 6",
"images": "trung.jpg",
"id": "6",
"description": "Test",
"amount": 1
}
}
}