# GraphQL

Mimanshu Maheshwari

December 4, 2024

# Contents

# List of Figures

# Listings

Chapter 1

# Why and What is GraphQL?

## 1.1 Why GraphQL

We all know that REST contributes more towards API development. In recent years, REST has become the dominant style for creating web services that are reusable and manageable. Everything in REST is represented as a resource which in turn has a unique URI to get identified. Also, REST works with a set of HTTP methods/verbs to define the operations being performed on the REST resources. Each one of us, the developers, will accept that REST is a great approach to proceed with API development as

- It leverages the usage of the standard HTTP protocol
- It is stateless
- It supports caching
- It is light-weight
- It supports multiple representations for resources and so on

But, we should agree that REST has its own limitations as well. And, the limitations are

- **Over-fetching**: The client might be interested to get the id and name of a resource, but the REST call will fetch the whole set of fields corresponding to that specific resource.
- **Multiple requests**: in case the client needs multiple data - REST API routes get longer when the data goes increasingly complex.

Because of these complexities, the client devices will slow down when there is limited network bandwidth. One more limitation of REST that needs to be put forth here is, it's important for the client to know the exact location of every single resource - If the API server holds multiple resources named customer, plan and so on, the client is expected to know the absolute URI of all the resources that are available with the server. Do we have anything in hand to overcome these issues?

Yes, we have something called GraphQL to help us with these issues.

## 1.2 What is GraphQL?

GraphQL is a server-side runtime and a query language for the APIs that we write It was developed by Facebook to optimize the REST calls to meet the growing requirements Later, Facebook open-sourced GraphQL as a specification for building APIs

GraphQL makes it possible to organize our data in the form of a graph and uses a powerful query syntax to traverse over the graph to retrieve and modify the data being stored in the graph.

From the definition above, it is evident that GraphQL is not only a specification for how to build APIs but also a specification for how to access the APIs from the client end. GraphQL queries are declarative by nature.

Now let's take a look at the advantages being offered by GraphQL that this course covers with detailed dem

- No over and under fetching – Ask for what is required and get that alone.
- A single request can help to retrieve multiple resources
- Describes well what's possible with a well-structured type system to help the clients understand the API better
- Move faster - in the sense of rapid API development with the help of powerful developer tools.

Resource utilization is much optimized because of the power of asking for only the things that are required and the ability to get multiple resources in a single hit/request. Hence, the client devices can perform better even on very less network bandwidth.

## 1.3 Ask for what do you want

We can send GraphQL queries to the APIs and get exactly what we need - nothing more, nothing less. A GraphQL query will always return a predictable result. Applications that use GraphQL are both fast and stable. Unlike RESTful calls, a GraphQL client call can actually restrict data that needs to be retrieved from the server. The following example can help us understand better. Let us consider an object Employee with the attributes - id, firstName, lastName, and technologyName. And, assume that we have a mobile application that needs to fetch the employees' firstName and id alone. If we design a RESTful endpoint for this requirement- `/api/v1/employees`, it just will end up fetching/retrieving data for every single field of the employee objects. This literally means, data is being over-fetched by the RESTful service. This problem can efficiently be addressed with the power of GraphQL. Using GraphQL, we can create a query similar to the one being shown below.

```
{
  employees {
    id
    firstName
  }
}
```

Listing 1.1: Employee fetch id and name only using GraphQL query

Listing 1.1 is a basic example of usage. Now, this query shall return the values of the `id` and `firstName` fields alone and will ignore other attributes like `lastName` and `technologyName` of the employee object. Observe the response for the above query:

```
  {
 "data": {
   "employees": [
     {
       "id": "E1001",
       "firstName": "John"
     },
     {
       "id": "E1002",
       "firstName": "Jane"
     }
   ]
```

```
        }
}
```
Listing 1.2: Employee fetch id and name only response