REST vs GraphQL

SAKTHI PRIYA M
MCA

API

An API or Application Program Interface, is a set of defined rules that explain how computers or applications communicate with one another. APIs sit between an application and the web server, acting as an intermediary layer that processes data transfer between systems.

For example,

When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API.

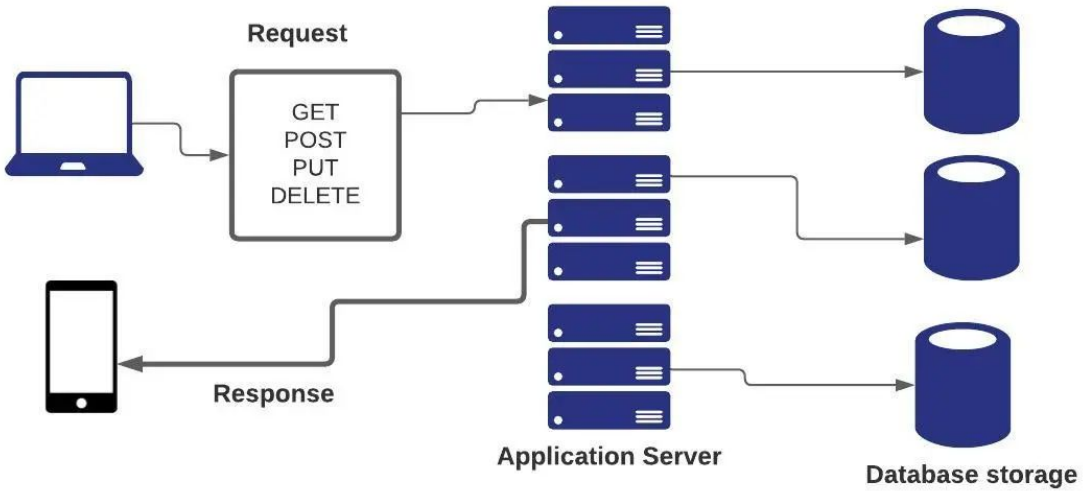
SOAP(Simple Object Access Protocol)

- The API creators used SOAP protocol because it was simple while also supporting XML format which is also really easy to use.
- SOAP relies heavily on XML, and together with schemas, defines a very strongly typed messaging framework.
- Every operation the service provides is explicitly defined, along with the XML structure of the request and response for that operation.
- Drawback - SOAP does provide error-handling features that make troubleshooting quick and easy when there are problems with the API or your network connection

REST(Representational State Transfer)

- REST is an architectural style that emphasizes using resources identified by URI and the transfer of representations to describe and manipulate them.
- GET, POST, PUT and DELETE are the most common REST operations. GET is used to read data from resource URIs. PUT creates a new resource while POST updates a resource. Finally, DELETE destroys a resource.
- Since RESTful APIs use JSON as their primary format, they are also known as “RESTful JSON APIs” or “JSON APIs.”
- The REST (short for representational state transfer) design paradigm centers on a uniform and predefined set of stateless operations that enable users to access and manipulate web data.
- In REST architecture, APIs expose their functionality as resources, which are any type of service, data, or object that a client can access. Every resource comes with its own unique URI (Uniform Resource Identifier) that a client can access by sending a request to the server.

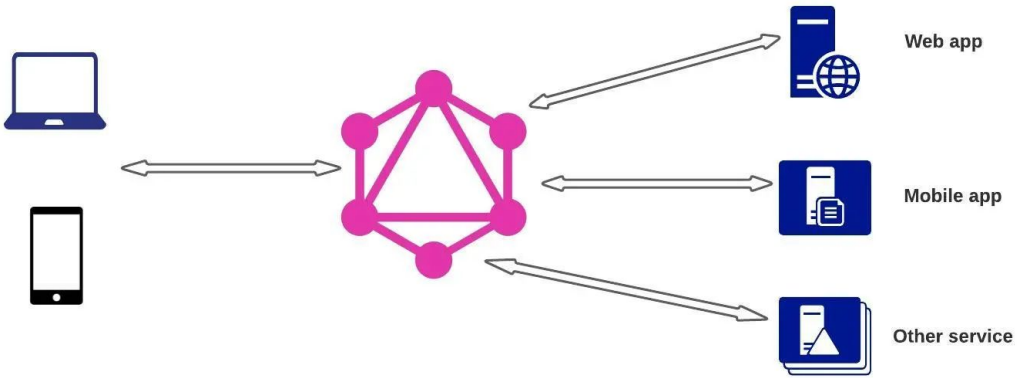
- So, whenever a client calls a RESTful API, the server will respond with a *representation* of the state of the queried resource. Many common REST implementations utilize the standard HTTP methods (GET, POST, PUT, DELETE, and PATCH) to call a server.
- Figure shows request and response structure in REST API



GraphQL

- GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.
- The language enables the client to make HTTP requests and get responses.
- In GraphQL, a set of information is seen in the context of a graph, just as the name suggests. Nodes, which are defined using the GraphQL schema system, represent objects. And edges represent the connection between nodes in the graph. This enables clear relationships between queries and increases the connectivity between objects.
- GraphQL allows users to request data from several resources using a single request. Rather than making multiple requests to fetch data, you can use it to make ad-hoc queries to a single endpoint and access all the required data.

- The main advantage of graphQL is that it provides the client with the luxury to specify the exact type of data to be received from the server
- Figure shows how client access data in graphQL



REST vs GraphQL

REST and GraphQL are two API design approaches that fulfill the same function: data transmission via internet protocols such as HTTP. However, how they do so varies significantly. GraphQL is a query language, whereas REST is an architectural pattern.

1. **Popularity**

   Most popular open-source technologies have been community-tested and are likely to be mature and stable. Furthermore, adopting a widely popular technology can help you leverage community support if you experience any issues during implementation.

   When it comes to GraphQL vs. REST popularity, the latter is clearly ahead.

   While GraphQL's usage is increasing rapidly, it's still somewhat new and may need more time to reach the realms of REST.

2. Usability

GraphQL allows you to send a request to your API and get the exact results you require—without unnecessary inclusions. Its queries return predictable results, which highly increases its usability.

On the contrary, REST's behavior usually varies based on the URI and HTTP method used. This makes it unclear for an API consumer to know what to expect when calling a new endpoint.

REST lacks clear and standardized guidelines for versioning

GraphQL follows a clear and straightforward approach to versioning

So, if we compare GraphQL vs. REST API in terms of predictability and versioning, GraphQL's simplicity significantly adds to its usability.

3. GraphQL vs REST Performance

Over-fetching happens when an endpoint delivers more data than is necessary. On the other hand, under-fetching indicates that an endpoint cannot return a significant amount of data.

REST APIs inherently have rigid data structures designed to return the stipulated data whenever they get hit, you may get unnecessary data or be forced to make multiple calls before getting the relevant data

GraphQL, rather than using fixed server-defined endpoints, uses a flexible style that allows you to retrieve what you want in a single API request.After defining the structure of the information you need, the same structure will be returned to you from the server, which avoids under-fetching and over-fetching.

Though GraphQL simplifies the number of requests an average application requires, REST tops it in the robustness of caching capabilities.

4. Security

REST provides several inherent ways to enforce the security of your APIs.

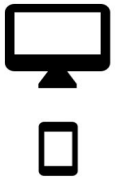
For example, you can ensure REST API security by implementing different API authentication methods, such as via HTTP authentication, where sensitive data is sent in HTTP headers, via JSON Web Tokens (JWT), where sensitive data is sent as JSON data structures

While GraphQL also provides some measures to ensure your APIs' security, they are not as mature as those of REST. For example, although GraphQL assists in integrating data validation, users are left on their own to apply authentication and authorization measures on top.

Data Fetching with REST vs GraphQL

With a REST API, you would typically gather the data by accessing multiple endpoints. In the example, these could be /users/<id> endpoint to fetch the initial user data. Secondly, there's likely to be a /users/<id>/posts endpoint that returns all the posts for a user. The third endpoint will then be the /users/<id>/followers that returns a list of followers per user.

1



HTTP GET

```
{
  “user”: {
    “id”: “er3tg439frjw”
    “name”: “Mary”,
    “address”: { … },
    “birthday”: “July 26, 1982”
  }
}
```

/users/<id>

/users/<id>/posts

/users/<id>/followers

2

HTTP GET

```
{
  “posts”: [{
    “id”: “ncwon3ce89hs”
    “title”: “Learn GraphQL today”,
    “content”: “Lorem ipsum … ”,
    “comments”: [ … ],
  }]
}
```

/users/<id>

/users/<id>/posts

/users/<id>/followers

3

```
{
  “followers”: [{
    “id”: “leo83h2dojsu”
    “name”: “John”,
    “address”: { … },
    “birthday”: “July 26, 1982”
  },
  …]
}
```

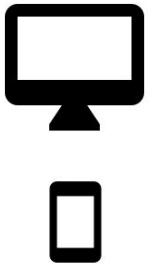
HTTP GET

/users/<id>

/users/<id>/posts

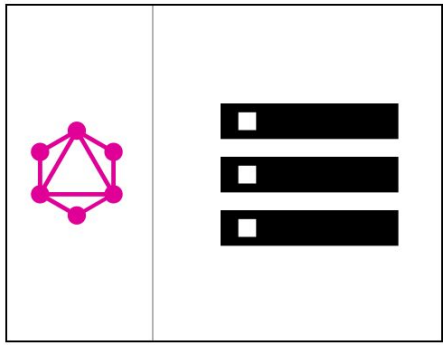
/users/<id>/followers

In GraphQL on the other hand, you’d simply send a single query to the GraphQL server that includes the concrete data requirements. The server then responds with a JSON object where these requirements are fulfilled.



```
query {
  User(id: “er3tg439frjw”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```

HTTP POST



```
{
  “data”: {
    “User”: {
      “name”: “Mary”,
      “posts”: [
        { title: “Learn GraphQL today” }
      ],
      “followers”: [
        { name: “John” },
        { name: “Alice” },
        { name: “Sarah” },
      ]
    }
  }
}
```

Reference

https://blog.api.rakuten.net/graphql-vs-rest/

https://www.howtographql.com/basics/1-graphql-is-the-better-rest/

THANK YOU