Group 10.1 REST and GRAPHQL Basics

By: Yogesh.C, Jaya. S, Arun. K, Ryan. Y

API

What is an API?

Application **P**rogramming **I**nterface

Allows different software systems to communicate

Provides a set of rules and protocols for building and interacting with software applications

API

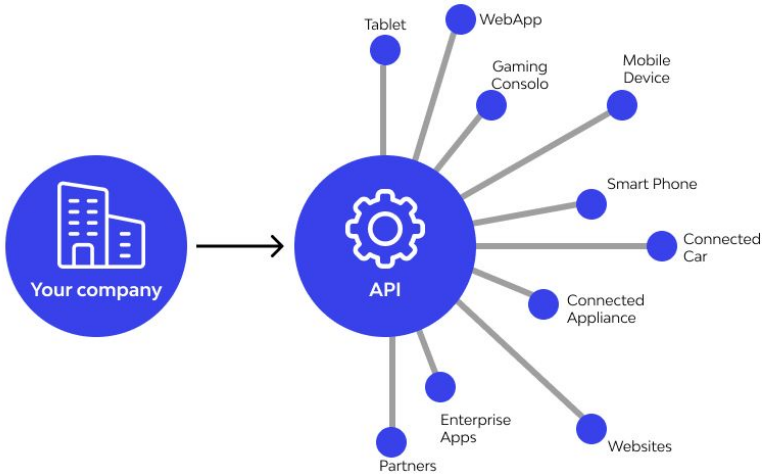
Why use API?

Enable integration between different systems

Facilitate automation of tasks

Provide access to data and services

Enhance functionality without reinventing the wheel



API

Types of API:

REST (Representational State Transfer):

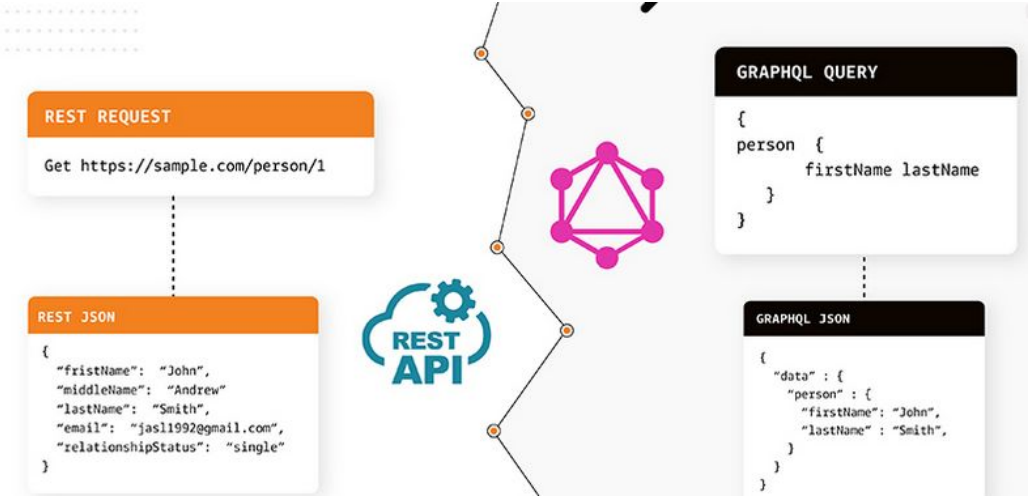
- Uses standard HTTP methods to perform CRUD (Create, Read, Update, Delete)
- Typically returns data in JSON format

SOAP (Simple Object Access Protocol):

- Uses XML for message format
- More rigid and standardized

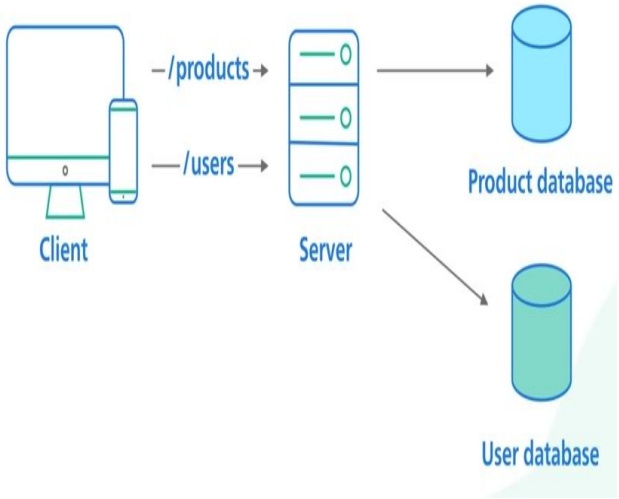
GraphQL:

- Allows clients to request exactly the data they need
- Single endpoint for querying



REST API

Endpoints: URL paths that represent specific resources



Methods

GET: Retrieve data (request data from a specified resource, such as getting a list of users or details about a specific us

POST: Create data (submit new data to a specified resource, such as creating a new user).

PUT: Update data(modify an existing resource, such as updating a user's information.)

DELETE: Remove data(deletes the specified resource, such as removing a user from the database.)

Status Codes

200: OK(The request has succeeded. This is the standard response for successful HTTP requests.)

201: Created(The request has succeeded and a new resource has been created as a result. This is typically the response to a POST request.

400: Bad Request(The server cannot or will not process the request due to a client error (e.g., malformed request syntax, invalid request parameters).

404: Not Found(he server cannot find the requested resource. This can happen if the endpoint is incorrect or the resource does not exist.)

500: Internal Server Error(The server encountered an unexpected condition that prevented it from fulfilling the request. This is a generic error message when no more specific message is suitable.

Benefits and challenges of Rest API

Benefits

- **Scalability->** REST APIs are stateless, meaning each request from a client to the server must contain all the information the server needs to fulfill that request. This statelessness simplifies server design and improves scalability, as servers can quickly allocate resources and handle more requests.
- **Flexibility and Portability**-> These are not tied to a specific protocol or client type. They can be consumed by any client capable of making HTTP requests, making them highly portable across different platforms and environments.
- **Simplicity->** It use standard HTTP methods (GET, POST, PUT, DELETE), which are easy to understand and use. This simplicity reduces the learning curve and facilitates quicker development and integration processes.
- **Performance->** It can be optimized for web performance, allowing for caching of responses. This reduces server load and improves response times, enhancing the overall performance of web applications.

Challenges

- **Over-fetching and Under-fetching->** May return more data than needed or insufficient data due to predefined response structures.
- **Statelessness->** Requires each request to contain all necessary information, potentially increasing payload size and latency.
- **Versioning->** Maintaining multiple versions can be complex and requires careful management to ensure compatibility.
- **Security->** REST APIs are often exposed over the internet, making them susceptible to various security threats like man-in-the-middle attacks, cross-site scripting (XSS), and injection attacks. Implementing robust security measures (like OAuth, HTTPS, and input validation) is essential but can add complexity.
- **Limited Flexibility->**Although REST APIs are flexible, they might not be as adaptable as other approaches (e.g., GraphQL) in terms of allowing clients to specify exactly what data they need. This can lead to inefficiencies in data retrieval and processing.

GRAPHQL API

What is GraphQL?

- A query language for your API
- Developed by Facebook in 2012 and released as an open-source project in 2015

Key Features:

- **Client-Specified Queries:** Clients can request exactly the data they need
- **Single Endpoint:** All queries are sent to one endpoint
- **Strongly Typed:** Schema defines the types of data and relationships

Benefits and Challenges of GraphQL

Benefits:

- Efficient data fetching: Only the data you need, no more, no less
- Flexibility and power: Combine multiple resources in a single request
- Strongly typed schema: Clear contract between client and server

Challenges:

- Complexity in setting up and maintaining the schema
- Learning curve for developers unfamiliar with GraphQL
- Over-fetching and under-fetching risks if not designed properly

Demo

Key points

-API enables seamless integration between systems, platforms, and services

-Saves time and effort by allowing you to use existing components and services

-REST is a widely used architectural style for designing networked applications

-REST utilizes standard HTTP methods to perform CRUD

-GraphQL powerful query language for API

-While REST APIs organized around multiple endpoints, GraphQL API uses a single endpoint.

-GraphQL uses a schema to define types and relationships providing clear contract between client and server