Chapter 2 - APIs

1. Anatomy of APIs

An **API (Application Programming Interface)** is a set of rules that allows different software applications to communicate with each other. APIs define the methods and data formats applications can use to request and exchange information. The key components of an API include:

- **Endpoints**: Specific URLs where API requests are sent.
- **Requests**: Calls made to the API using HTTP methods (GET, POST, PUT, DELETE, etc.).
- **Responses**: Data sent back by the API, often in JSON or XML format.
- **Headers**: Contain metadata about the request or response (e.g., authentication tokens, content type).
- **Authentication**: Mechanisms like API keys, OAuth, or JWT to secure access.
- **Rate Limiting**: Restricts the number of requests a client can make within a time period.

2. Web Services

Web services are software applications that use **standardized web protocols** to communicate over a network. They enable interoperability between different applications and platforms. Common types include:

- **SOAP (Simple Object Access Protocol)**
- **REST (Representational State Transfer)**
- **GraphQL**

3. HTTP (Hypertext Transfer Protocol)

HTTP is the foundation of communication for APIs on the web. It defines how requests and responses are structured. Important aspects of HTTP include:

- **Methods**:
  - `GET` : Retrieve data from a server.
  - `POST` : Send data to the server to create a resource.
  - `PUT` : Update an existing resource.
  - `DELETE` : Remove a resource.
- **Status Codes**:
  - `200 OK` : Success.

- `201 Created` : Resource successfully created.
- `400 Bad Request` : Invalid request.
- `401 Unauthorized` : Authentication required.
- `404 Not Found` : Resource not found.
- `500 Internal Server Error` : Server-side error.

4. XML (Extensible Markup Language)

XML is a structured, hierarchical data format used for exchanging information between applications. It was commonly used in **SOAP-based APIs**. Example:

```xml
<user>
    <id>1</id>
    <name>John Doe</name>
</user>
```

5. JSON (JavaScript Object Notation)

JSON is a lightweight data format used in **RESTful APIs**. It is more human-readable and efficient compared to XML. Example:

```json
{
    "id": 1,
    "name": "John Doe"
}
```

6. SOAP (Simple Object Access Protocol)

SOAP is a protocol that allows applications to communicate over the internet using XML-based messages. It is more secure and standardized but is considered heavyweight compared to REST.

- Uses XML exclusively for messaging.
- Operates over HTTP, SMTP, TCP, or other transport protocols.
- Includes built-in security and transaction compliance.

7. REST (Representational State Transfer)

REST is an architectural style that provides a lightweight way of exchanging data over HTTP. It is the most commonly used API type today.

- Uses standard HTTP methods (GET, POST, PUT, DELETE).
- Stateless: Each request is independent and contains all necessary information.
- Responses are usually in JSON but can also be in XML or other formats.
- Example RESTful API request (GET):

```
GET /users/1 HTTP/1.1
Host: example.com
Authorization: Bearer token123
```

8. GraphQL

GraphQL is a query language for APIs that allows clients to request exactly the data they need.

- Unlike REST, GraphQL allows fetching multiple resources in a single request.
- Clients specify the structure of the response, reducing over-fetching and under-fetching of data.
- Example query:

```
query {
  user(id: 1) {
    name
    email
  }
}
```

9. Example of an API

A simple **REST API for user management**:

Endpoint: `https://api.example.com/users`

GET Request (Retrieve all users):

```
GET /users HTTP/1.1
Host: api.example.com
```

Response:

```
[
    { "id": 1, "name": "Alice" },
    { "id": 2, "name": "Bob" }
]
```

POST Request (Create a new user):

```
POST /users HTTP/1.1
Host: api.example.com
Content-Type: application/json

{
    "name": "Charlie"
}
```

Response:

```
{
    "id": 3,
    "name": "Charlie"
}
```