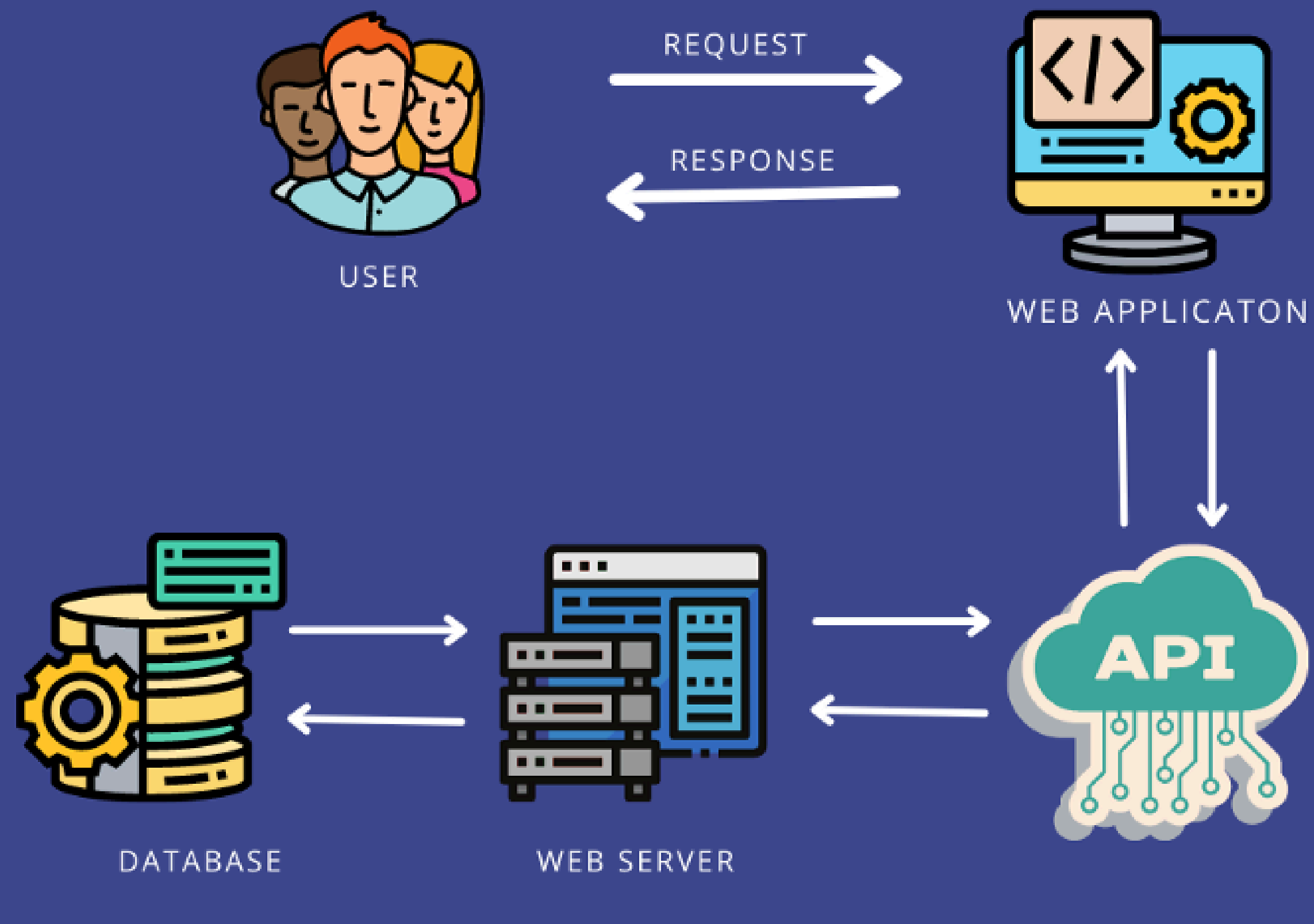
kodemaven-portfolio.vercel.app

RESTful APIs In Web Development



What are REST APIs?

RESTful APIs, a.k.a. Representational State Transfer, are a set of rules and conventions for building and interacting with web services. They use standard HTTP methods to carry out their operations.

Properties of RESTful APIs

- RESTful APIs use standard HTTP methods like GET, POST, PUT, and DELETE.
- They are stateless, meaning each request from a client to server must contain all the information needed to understand and process the request.
- They are known for their simplicity, scalability, and flexibility, making them ideal for web and mobile applications.

Why are RESTful APIs Important

- They easily handle multiple client-server interactions, which make them scalable to use.

- They work across different platforms and languages.

- They are efficient in the use of HTTP protocols which leads to high performance in web applications.

- They have clear structure and separation of concerns for easier maintenance.

Operations in RESTful APIs

RESTful APIs use standard HTTP methods to perform operations on resources. These methods include:

- **Create (POST): Add new data.**
- **Read (GET): Retrieve data.**
- **Update (PUT/PATCH): Modify existing data.**
- **Delete (DELETE): Remove data.**

kodemaven-portfolio.vercel.app

Let's use a restaurant analogy to understand these better!

1. Create (POST): Add new data.

The POST method in RESTful APIs is used to create a new resource on the server.

Think of this as a customer who places an order at a restaurant.

The customer knows what they want and could make a list of requests as he checks through the menu list and can add new items as many times as possible.

kodemaven-portfolio.vercel.app

Customer can place orders for anything on the Menu List (POST request), and it is being added to the customer’s Order List each time.

```
async function createOrder(order) {
  const response = await fetch('https://api.example.com/orders', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(order),
  });
  const data = await response.json();
  console.log('Order created:', data);
}

createOrder({ item: 'Pizza', quantity: 1 });
```

The client sends a POST request to the server with the order details. The server processes the order and responds with the created order data.

2. Read (GET): Retrieve data.

The GET method is one of the fundamental operations in RESTful APIs, used to retrieve data from a server. It is a read-only operation, meaning it does not alter the state of the resource on the server.

The customer has made some orders, but wants to check the items on his Order List. He could check for a particular variety of meal or could see all the items he has on his Menu List depending on his request (query).

kodemaven-portfolio.vercel.app

```
async function getOrders() {
  const response = await fetch('https://api.example.com/orders');
  const data = await response.json();
  console.log('Orders:', data);
}

getOrders();
```

The customer sends a **GET** request to the server to retrieve the list of orders. The server responds with the requested data.

3. Update (PUT/PATCH): Modify existing data.

In the context of RESTful APIs, the **PUT** and **PATCH** methods are used to update existing resources. Although both methods serve the purpose of updating data, they differ in their functionality and typical use cases. Let’s see that:

PUT Method

The PUT method is used to update a resource by replacing the existing data with the new data provided in the request.

PUT does a complete replacement of the entire resource. If certain fields are omitted in the request during update, they are replaced with null or default values in the resource.

kodemaven-portfolio.vercel.app

Original data structure in the resource

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john.doe@example.com",
  "age": 30
}
```

During **PUT** update, all structure fields must be in the request payload, including the non-updated fields.

```
PUT /users/1 HTTP/1.1
Host: api.example.com
Content-Type: application/json

{
  "id": 1,
  "name": "John Smith",
  "email": "john.smith@example.com",
  "age": 31
}
```

PATCH Method

The PATCH method is used to apply partial updates to a resource. Unlike PUT, PATCH only modifies the fields provided in the request payload, leaving the other fields unchanged.

Original data structure in the resource

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john.doe@example.com",
  "age": 30
}
```

During PATCH update, only the modified field is provided in the request payload

```
{
  "name": "John Smith"
}
```

kodemaven-portfolio.vercel.app

For our restaurant analogy, the **PUT** method will be that, when the customer wants to update his Order List, he has to rewrite all his orders (both the new update on a menu and the unchanged menu)

But if customer uses **PATCH** method, he only have to rewrite the particular items he wants to make changes to.

4. Delete (DELETE): Remove data.

The DELETE method as one of the standard HTTP methods used in RESTful APIs, is designed to delete a specified resource from the server. When a client sends a DELETE request, it instructs the server to remove the resource identified by the URL.

DELETE requests usually do not have a request body, the URL used in the DELETE request specifies which resource should be deleted. The resource is typically identified by its unique identifier (ID).

kodemaven-portfolio.vercel.app

For our restaurant analogy, the **DELETE** method will be that the customer cancels an order by the item's index (removed from Order List).

```
async function deleteOrder(orderId) {
  const response = await
fetch(`https://api.example.com/orders/${orderId}`, {
    method: 'DELETE',
  });
  if (response.ok) {
    console.log('Order deleted');
  } else {
    console.log('Failed to delete order');
  }
}
deleteOrder(1);
```

The customer sends a DELETE request to the server to remove an order. The server processes the deletion and confirms it.

RESTful APIs are a powerful tool in modern web and mobile application development, making data interaction seamless and efficient.

kodemaven-portfolio.vercel.app

Stay tuned as we continue our journey into the world of APIs!!!

Hi There! 👋

Thank you for reading through

Did you enjoy this knowledge?

💼 Follow my LinkedIn page for more work-life balancing and Coding tips.

LinkedIn: Oluwakemi Oluwadahunsi

kodemaven-portfolio.vercel.app