**Developing an API with PHP**

Constructing an API with PHP involves creating endpoints that enable clients to interact with your server via HTTP requests.

Establishing a Database using MySQL or another suitable database to store and manage data.

Creating Endpoints developing PHP scripts to handle various HTTP methods (GET, POST, PUT, DELETE).

Routing implementing routing mechanisms to direct URLs to specific functions or scripts.

Handling Requests and Responses utilizing PHP's superglobals (`$\_GET`, `$\_POST`, etc.) to process incoming requests and format JSON responses.

Security Measures incorporating authentication, authorization, and input validation to secure API.

<?php

        // Connect to the MySQL database

        // mysqli\_connect(server, username, password, database)

        $con = mysqli\_connect("localhost", "root" , "", "API\_DATA");

        // Initialize an empty array to store the response

        $response = array();

        // Variable to keep track of array index

        $i = 0;

        // Check if the connection was successful

        if($con){

            // SQL query to select all data from the "data" table

            $sql = "SELECT \* FROM data";

            // Execute the SQL query and store the result

            $result = mysqli\_query($con, $sql);

            // Check if the query execution was successful

            if($result) {

                // Set the content type of the response to JSON

                header("Content-Type: application/json");

                // Fetch data from the result set row by row

                while($row = mysqli\_fetch\_assoc($result)){

                    // Assign the fetched data to the response array

                    $response[$i]['id'] = $row['id'];

                    $response[$i]['name'] = $row['name'];

                    $response[$i]['age'] = $row['age'];

                    $response[$i]['email'] = $row['email'];

                    // Increment the index for the next iteration

                    $i++;

                }

                // Encode the response array into JSON format and print it

                echo json\_encode($response, JSON\_PRETTY\_PRINT);

            }

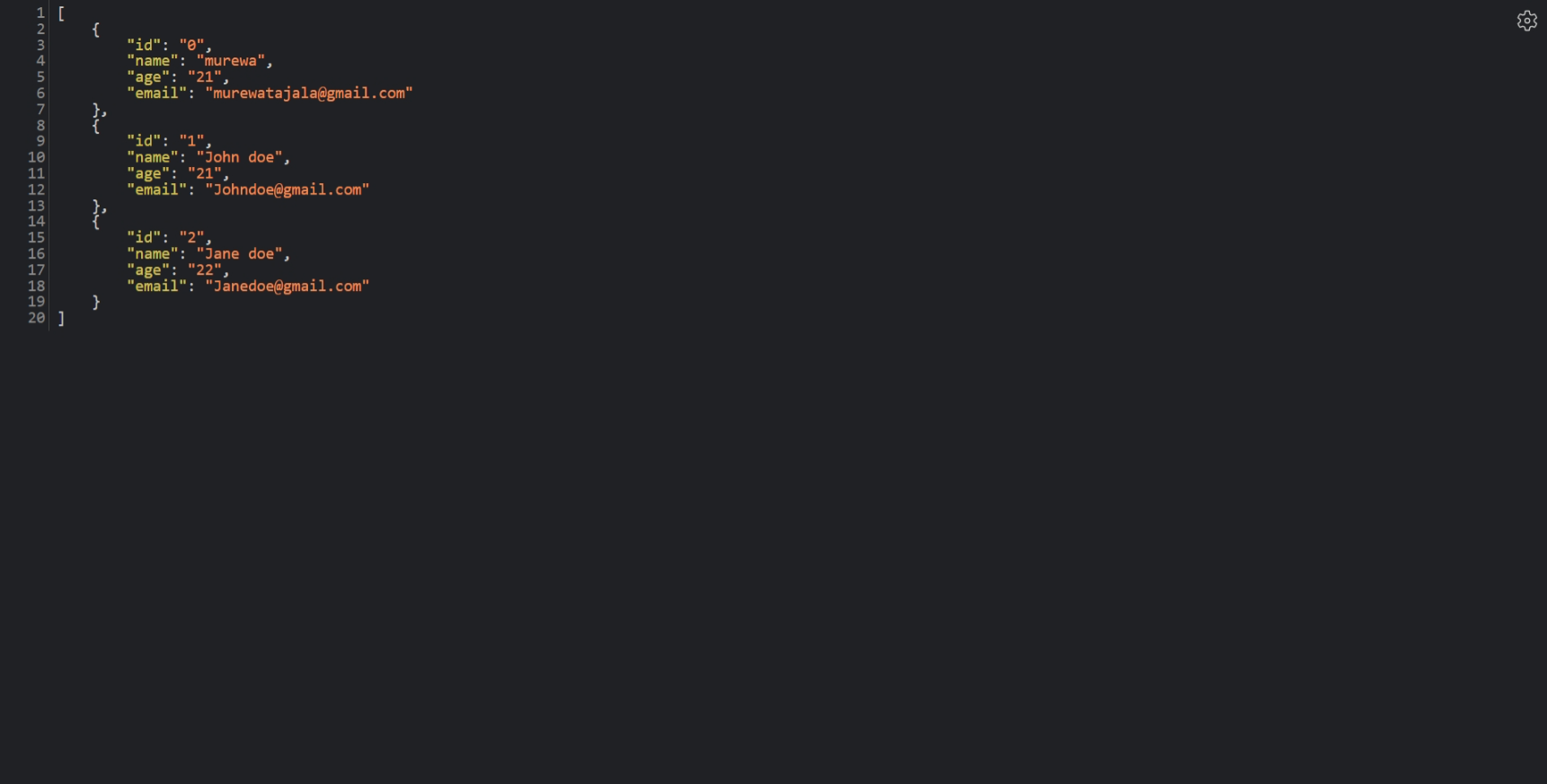
        } else {

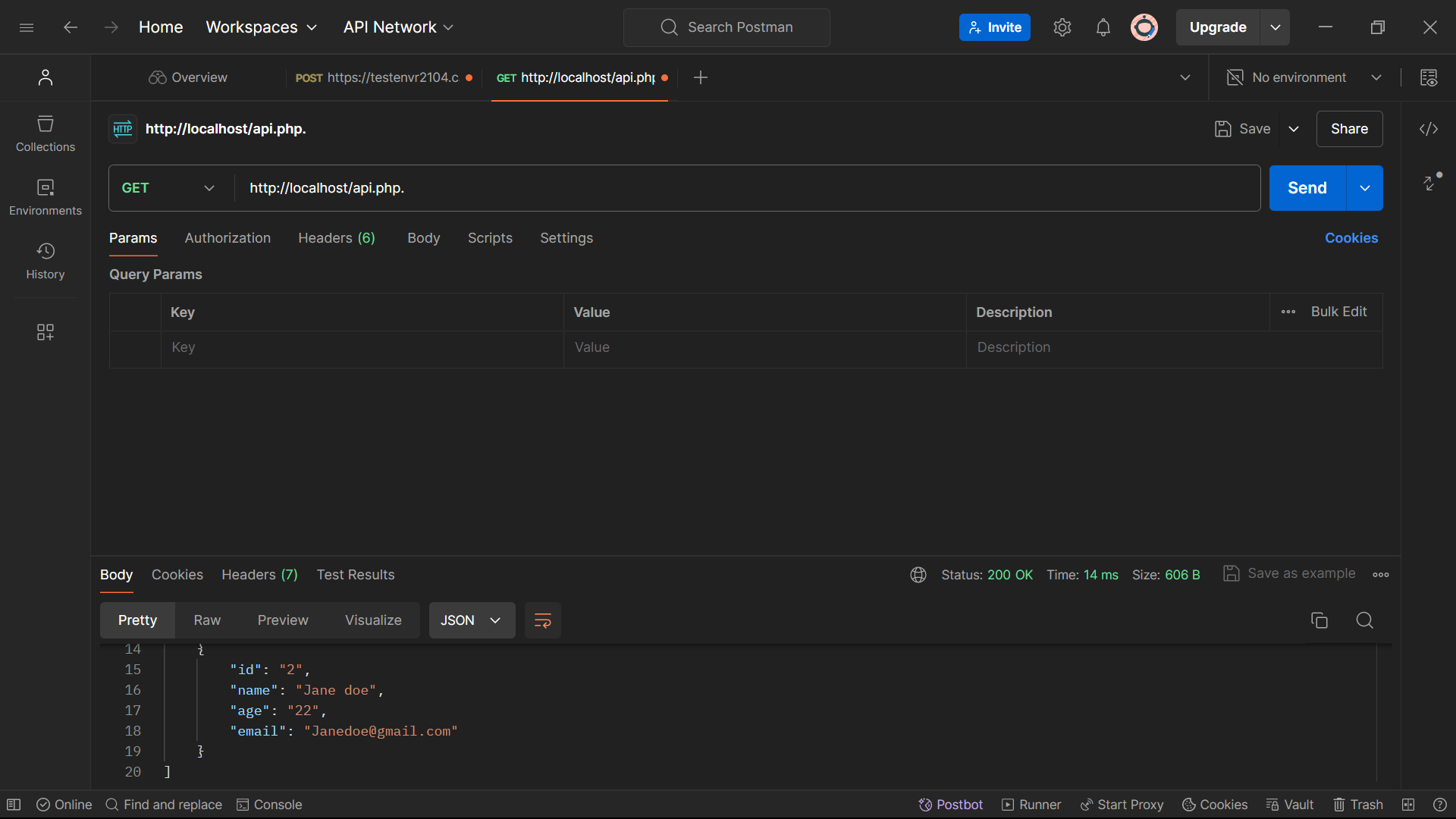
            // If the connection failed, print an error message

            echo "Connection failed";

        }

?>





phpMyAdmin

phpMyAdmin is an open-source, web-based application written in PHP. Its primary function is to facilitate the administration of MySQL and MariaDB databases. Key functionalities include:

- Database Management: Create, modify, and delete databases and their structures.

- SQL Query Execution: Execute SQL queries directly through a user-friendly interface.

- User and Privilege Management: Manage database users and their permissions.

- Data Import and Export: Transfer data between different database systems.

- Maintenance Tasks: Perform routine maintenance operations on databases.

Creating Database Tables

When creating tables within a database, A framework for storing data is defined. Using SQL's `CREATE TABLE` statement:

CREATE TABLE users (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    username VARCHAR(50) NOT NULL,

    email VARCHAR(100) NOT NULL,

    created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

In this example, a users table is established with columns for unique `id`, `username`, `email`, and a timestamp indicating creation time.

Rows and Columns in Database

- Columns: Each column represents a distinct attribute of the stored data, such as `name`, `email`, or `age`. Each column possesses specific data types (e.g., INT, VARCHAR, DATE) and constraints (e.g., NOT NULL, UNIQUE).

- Rows: Alternatively known as records, rows are individual entries within a table. Each row contains data corresponding to the defined columns. For instance, a row in the `users` table might include a specific user's ID, username, email, and the timestamp of record creation.

SQL Essentials

SQL (Structured Query Language) serves as a universal programming language for managing and manipulating relational databases. Common SQL commands encompass:

- SELECT: Retrieve data from a database.

- INSERT: Add new records to a table.

- UPDATE: Modify existing records.

- DELETE: Remove records from a table.

- CREATE: Establish new databases, tables, or other database elements.

- ALTER: Adjust the structure of an existing table.

- DROP: Delete databases, tables, or other database entities.

An instance of a SELECT query:

SELECT username, email FROM users WHERE id = 1;

This query retrieves the `username` and `email` from the `users` table for the user with an `id` of 1.

An instance of a SELECT query:

To insert a new record into the users table:

INSERT INTO users (username, email, created\_at)

VALUES ('john\_doe', 'john.doe@email.com', NOW());

This query adds a new user with the username 'john\_doe', email 'john.doe@email.com', and sets the created\_at timestamp to the current time.

### UPDATE: Modify Existing Records:

To update an existing record in the users table:

UPDATE usersSET email = 'updated.email@email.com'  
WHERE id = 1;

This query updates the email address for the user with id 1 to 'updated.email@email.com'.

### DELETE: Remove Records from a Table

To delete a record from the users table:

DELETE FROM users

WHERE id = 1;

This query deletes the user with id 1 from the users table.

### CREATE: Establish New Databases, Tables, or Other Database Elements

To create a new database named company:

CREATE DATABASE company;

To create a new table products in the company database:

CREATE TABLE products (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    price DECIMAL(10, 2) NOT NULL

);

This creates a table products with columns id, name, and price.

### ALTER: Adjust the Structure of an Existing Table

To add a new column age to the users table:

ALTER TABLE users

ADD COLUMN age INT;

This command modifies the users table by adding a new column age of type INT.

### DROP: Delete Databases, Tables, or Other Database Entities

To drop a table products from the company database:

USE company;

DROP TABLE products;

This command deletes the products table from the company database.