

Nama : Zaenal Abidin Syah

Nim : 210411100186

Kelas : Pengembangan Aplikasi Terintegrasi A

Dokumentasi Api

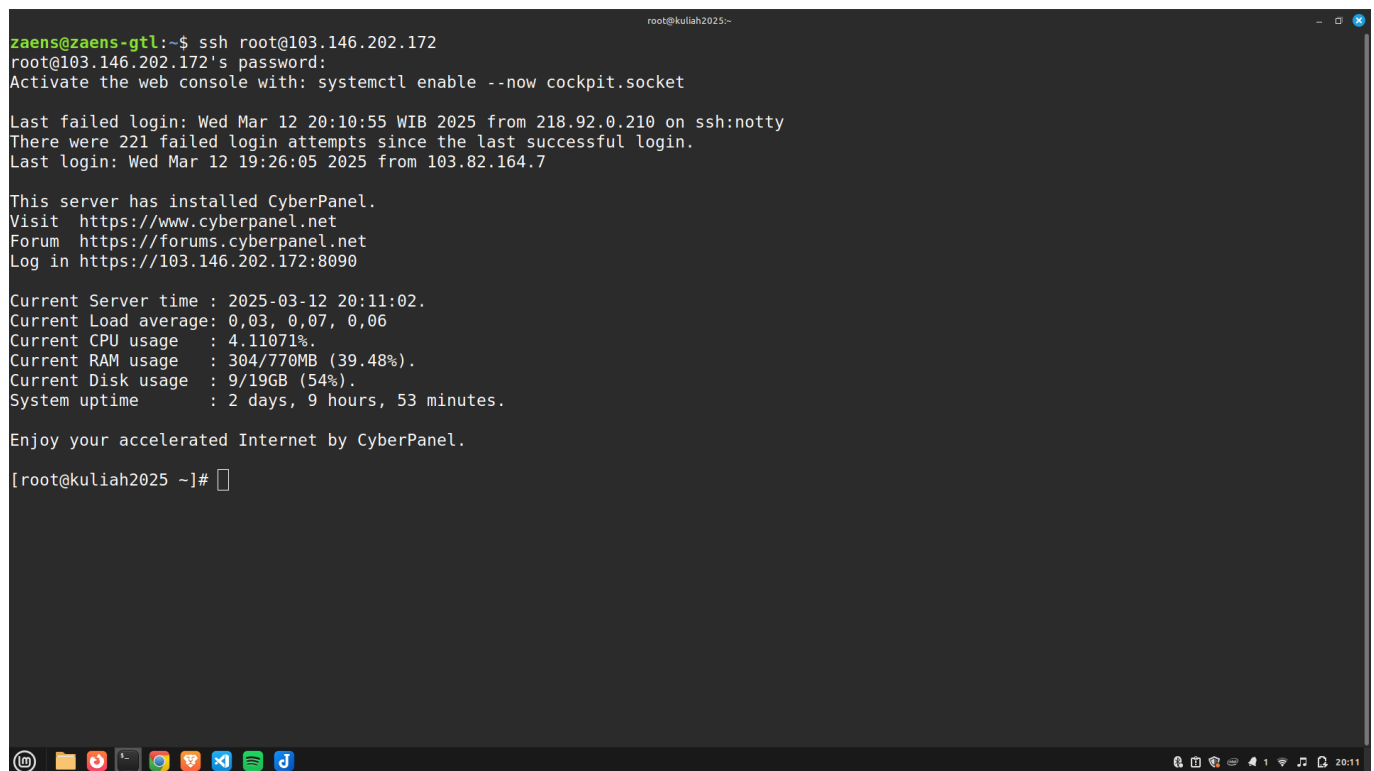
buat sub folder dengan nama t[nim]

ssh ke server

buka terminal untuk mengakses ssh

```
ssh root@103.146.202.172
```

masukan password, dan jika berhasil maka akan seperti gambar berikut

A screenshot of a terminal window. The terminal shows the command 'ssh root@103.146.202.172' being executed from a host named 'zaens@zaens-gtl'. The user is prompted for a password. After login, the terminal displays system information: 'Last failed login: Wed Mar 12 20:10:55 WIB 2025 from 218.92.0.210 on ssh:notty', 'There were 221 failed login attempts since the last successful login.', 'Last login: Wed Mar 12 19:26:05 2025 from 103.82.164.7', and 'This server has installed CyberPanel.' It also shows system statistics: 'Current Server time : 2025-03-12 20:11:02.', 'Current Load average: 0,03, 0,07, 0,06', 'Current CPU usage : 4.11071%', 'Current RAM usage : 304/770MB (39.48%).', 'Current Disk usage : 9/19GB (54%).', and 'System uptime : 2 days, 9 hours, 53 minutes.' The prompt changes to '[root@kuliah2025 ~]#'.

```
zaens@zaens-gtl:~$ ssh root@103.146.202.172
root@103.146.202.172's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last failed login: Wed Mar 12 20:10:55 WIB 2025 from 218.92.0.210 on ssh:notty
There were 221 failed login attempts since the last successful login.
Last login: Wed Mar 12 19:26:05 2025 from 103.82.164.7

This server has installed CyberPanel.
Visit https://www.cyberpanel.net
Forum https://forums.cyberpanel.net
Log in https://103.146.202.172:8090

Current Server time : 2025-03-12 20:11:02.
Current Load average: 0,03, 0,07, 0,06
Current CPU usage : 4.11071%.
Current RAM usage : 304/770MB (39.48%).
Current Disk usage : 9/19GB (54%).
System uptime : 2 days, 9 hours, 53 minutes.

Enjoy your accelerated Internet by CyberPanel.
[root@kuliah2025 ~]#
```

Membuat Subfolder

Setelah masuk ke vps melalui ssh, selanjutnya lakukan change directory atau cd ke /home/kuliah2025.my.id/public_html. Lalu ketikkan perintah terminal mkdir untuk membuat directory baru, dan ls untuk menampilkan directory.

```
cd /home/kuliah2025.my.id/public_html/
mkdir t21-186
```

```
ls
```

dan hasilnya akan seperti gambar berikut

A terminal window screenshot showing a series of commands and their outputs. The user is in a directory named 'public_html' and has just created a new directory 't21-186'. The 'ls' command shows the contents of the current directory, including a file '21-017_fahrurrosi' and several other files and directories. The terminal has a dark background with light blue and white text. The window title is 'root@kuliah2025/home/kuliah2025.my.id/public_html'. The bottom of the window shows a taskbar with various application icons and system status icons on the right.

```
root@kuliah2025 ~/# cd /home/kuliah2025.my.id/public_html/
root@kuliah2025 public_html# mkdir t21-186
root@kuliah2025 public_html# ls
21-017_fahrurrosi  index.html  info.php  t01  t060  t061  t064  t087  t089  t099  t126  t21-186  t21-1863  t220411100037  test_db.php
root@kuliah2025 public_html#
```

directory t21-186 adalah folder tempat code yang saya gunakan untuk membuat api, change directory ke folder tersebut

```
cd t21-186
```

membuat subfolder telah selesai, selanjutnya tinggal membuat file code api

buat file code apinya

membuat file php dengan mengetikan perintah berikut

```
nano test.php
```

berikut setelah mengetikan perintah diatas

```
GNU nano 2.9.8                                test.php
root@kuliah2025:/home/kuliah2025.my.id/public_html/121-186

<?php
// test_api.php

// Set header untuk JSON
header('Content-Type: application/json');

// Mengirim response sederhana dalam format JSON
$response = [
    'message' => 'Test API working successfully'
];

echo json_encode($response);
?>
```

lalu masukan kode berikut

```
<?php
// Set header untuk JSON
header('Content-Type: application/json');

// Mengirim response sederhana dalam format JSON
$response = [
    'message' => 'Test API working successfully'
];

echo json_encode($response);
?>
```

penjelasan code

```
header('Content-Type: application/json');
```

code di atas digunakan untuk mengeset header agar server mengembalikan json.

```
$response = [
    'message' => 'Test API working successfully'
];

echo json_encode($response);
```

code diatas untuk mengirimkan json berupa response yang didalamnya terdapat message 'Test API working successfully'

pembuatan file code api selesai dilanjutkan pengujian api menggunakan CURL dan Postman.

lakukan pengujian api

ada beberapa opsi untuk melakukan pengujian api seperti menggunakan CURL dan postman. Postman dapat digunakan untuk pengujian api dengan menggunakan Interface atau GUI. sementara CURL hanya dapat diakses melalui terminal saja. pengujian dilakukan dengan memasukkan url dari api yaitu <https://kuliah2025.my.id/t21-186/test.php>

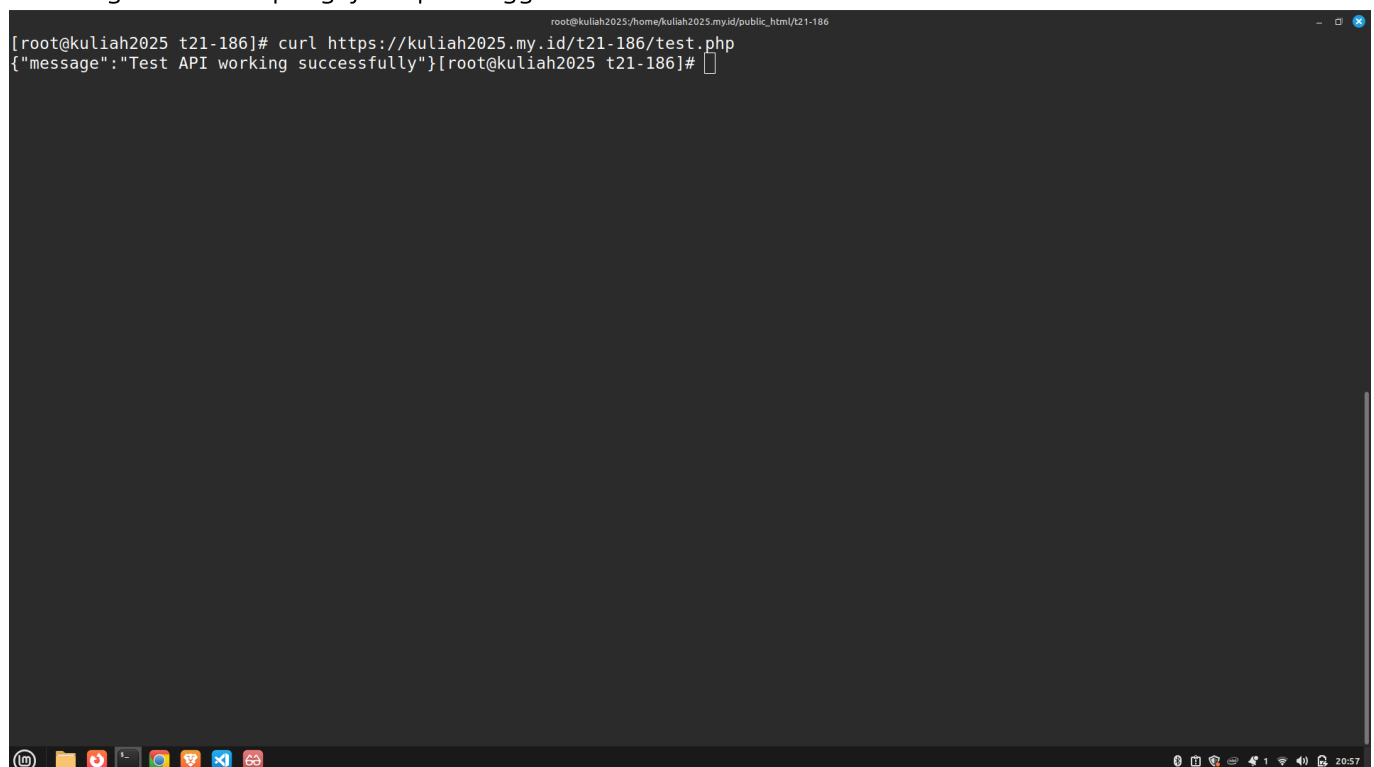
untuk menggunakan curl ketikkan perintah berikut

```
curl https://kuliah2025.my.id/t21-186/test.php
```

akan didapatkan hasil seperti berikut

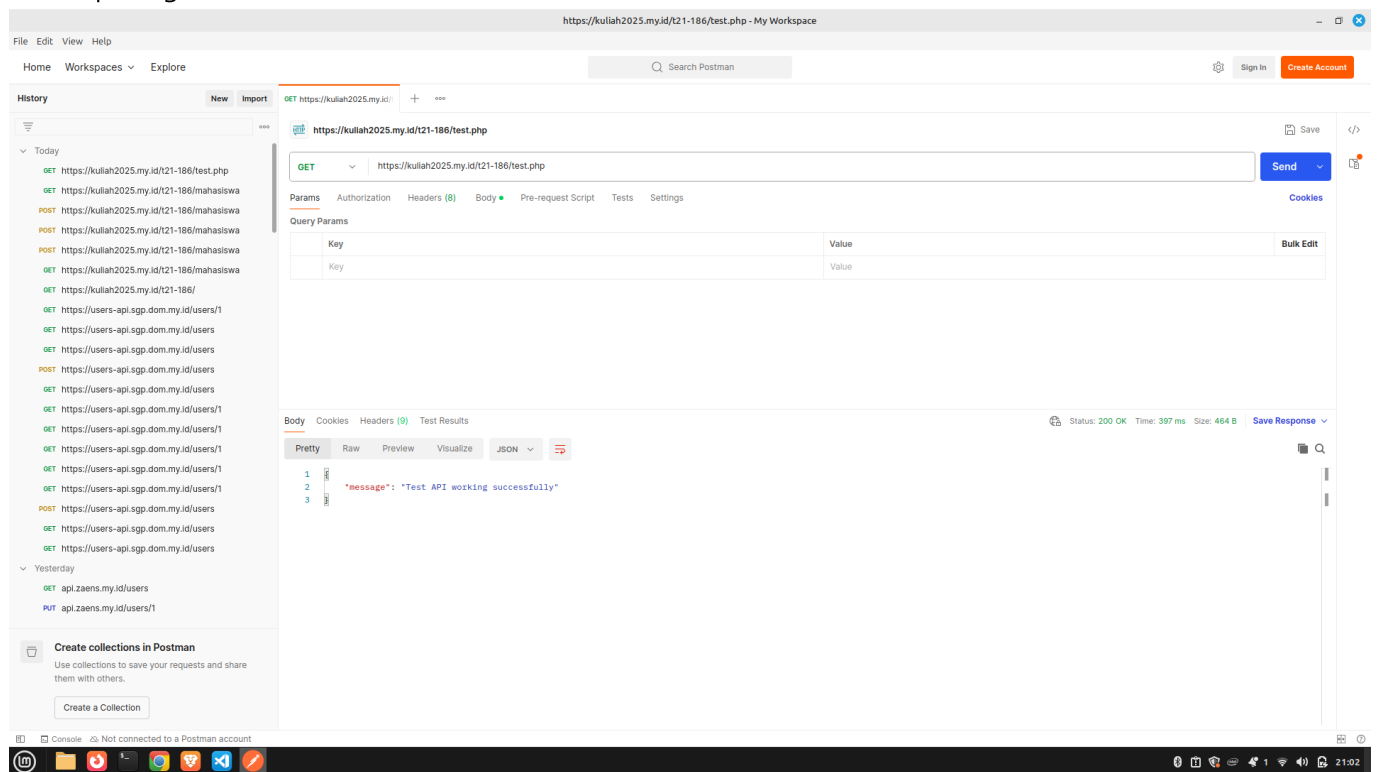
```
{ "message": "Test API working successfully" }
```

berikut gambar hasil pengujian api menggunakan curl



untuk menggunakan postman buka aplikasi postman masukan url test api <https://kuliah2025.my.id/t21-186/test.php>, lalu pilih metode get lalu tekan send untuk mengirim request ke api. untuk pengujiannya dapat

dilihat pada gambar berikut.



untuk melihat response dari server bisa dilihat di bagian bawah. server mengirimkan response json seperti berikut

```
{
  "message": "Test API working successfully"
}
```

pengujian api berhasil dilakukan menggunakan CURL maupun postman. selanjut dapat membuat api yang terintegrasi dengan database.

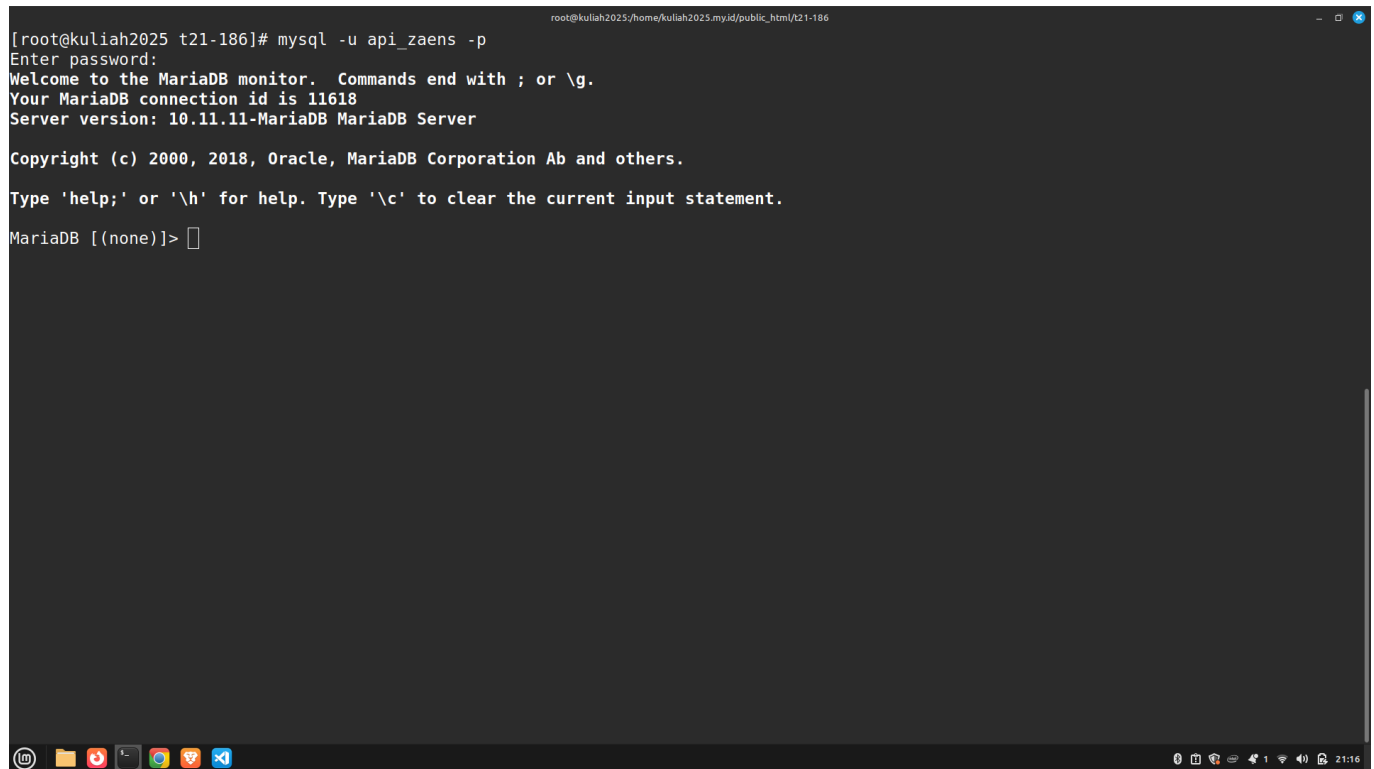
membuat api (GET) dari database mysql,tabel mahasiswa, dengan parameter nama mahasiswa

membuat database

masuk ke mysql untuk membuat database

```
mysql -u api_zuens -p
```

masukan password, jika berhasil maka akan tampil seperti gambar berikut

A screenshot of a terminal window with a dark background. The terminal shows the command `mysql -u api_zaens -p` being executed. It prompts for a password, then displays a welcome message from the MariaDB monitor, including the connection ID (11618) and server version (10.11.11-MariaDB). It also shows copyright information and instructions on how to use help or clear the input. The prompt `MariaDB [(none)]>` is visible at the bottom of the terminal output.

```
[root@kuliah2025 t21-186]# mysql -u api_zaens -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11618
Server version: 10.11.11-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

selanjutnya membuat database 21_mahasiswadatabase dengan perintah berikut

```
create database 21_mahasiswadatabase;
```

setelah membuat database selanjutnya membuat table mahasiswa

```
use 21_mahasiswadatabase;
```

setelah mengubah database ke 21_mahasiswadatabase selanjutnya membuat table mahasiswa dengan perintah berikut

```
CREATE TABLE `mahasiswa` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL,
  `nim` varchar(20) NOT NULL,
  `email` varchar(100) NOT NULL,
  PRIMARY KEY (`id`)
)
```

hasilnya akan seperti gambar berikut

```
[root@kuliah2025 t21-186]# mysql -u api_zaens -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 11622
Server version: 10.11.11-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database 21_mahasiswadatabase;
Query OK, 1 row affected (0,002 sec)

MariaDB [(none)]> use 21_mahasiswadatabase;
Database changed
MariaDB [21_mahasiswadatabase]> CREATE TABLE `mahasiswa` (
  -> `id` int(11) NOT NULL AUTO_INCREMENT,
  -> `name` varchar(100) NOT NULL,
  -> `nim` varchar(20) NOT NULL,
  -> `email` varchar(100) NOT NULL,
  -> PRIMARY KEY (`id`)
  -> );
Query OK, 0 rows affected (0,060 sec)
```

setelah membuat table mahasiswa selanjutnya isi table mahasiswa dengan perintah berikut

```
INSERT INTO mahasiswa(name, nim, email) VALUES
('Zaenal Abidin
Syah', '210411100186', '210411100186@student.trunojoyo.ac.id'),
('Budi Santoso', '210411100187', '210411100187@student.trunojoyo.ac.id'),
('Citra Dewi', '210411100188', '210411100188@student.trunojoyo.ac.id'),
('Dewi Lestari', '210411100189', '210411100189@student.trunojoyo.ac.id'),
('Eko Prasetyo', '210411100190', '210411100190@student.trunojoyo.ac.id'),
('Fajar Hidayat', '210411100191', '210411100191@student.trunojoyo.ac.id'),
('Gita Permata', '210411100192', '210411100192@student.trunojoyo.ac.id'),
('Hendra Wijaya', '210411100193', '210411100193@student.trunojoyo.ac.id'),
('Indra Kurniawan', '210411100194', '210411100194@student.trunojoyo.ac.id'),
('Joko Susilo', '210411100195', '210411100195@student.trunojoyo.ac.id'),
('Kiki Amalia', '210411100196', '210411100196@student.trunojoyo.ac.id'),
('Lina Marlina', '210411100197', '210411100197@student.trunojoyo.ac.id'),
('Maman Subagyo', '210411100198', '210411100198@student.trunojoyo.ac.id'),
('Nina Puspita', '210411100199', '210411100199@student.trunojoyo.ac.id'),
('Oki Setiawan', '210411100200', '210411100200@student.trunojoyo.ac.id'),
('Putu Dwi', '210411100201', '210411100201@student.trunojoyo.ac.id'),
('Rina Sari', '210411100202', '210411100202@student.trunojoyo.ac.id'),
('Sari Wulandari', '210411100203', '210411100203@student.trunojoyo.ac.id'),
('Teguh Pratama', '210411100204', '210411100204@student.trunojoyo.ac.id'),
('Umar Hidayat', '210411100205', '210411100205@student.trunojoyo.ac.id');
```

perintah berikut akan menambahkan data mahasiswa sebanyak 20 data. hasilnya dapat dilihat dari gambar berikut

```
root@kuliah2025:/home/kuliah2025.my.id/public_html/k21-186

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use 21_mahasiswadatabase;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [21_mahasiswadatabase]> INSERT INTO mahasiswa(name, nim, email) VALUES
-> ('Zaenal Abidin Syah','210411100186','210411100186@student.trunojoyo.ac.id'),
-> ('Budi Santoso','210411100187','210411100187@student.trunojoyo.ac.id'),
-> ('Citra Dewi','210411100188','210411100188@student.trunojoyo.ac.id'),
-> ('Dewi Lestari','210411100189','210411100189@student.trunojoyo.ac.id'),
-> ('Eko Prasetyo','210411100190','210411100190@student.trunojoyo.ac.id'),
-> ('Fajar Hidayat','210411100191','210411100191@student.trunojoyo.ac.id'),
-> ('Gita Permata','210411100192','210411100192@student.trunojoyo.ac.id'),
-> ('Hendra Wijaya','210411100193','210411100193@student.trunojoyo.ac.id'),
-> ('Indra Kurniawan','210411100194','210411100194@student.trunojoyo.ac.id'),
-> ('Joko Susilo','210411100195','210411100195@student.trunojoyo.ac.id'),
-> ('Kiki Amalia','210411100196','210411100196@student.trunojoyo.ac.id'),
-> ('Lina Marlina','210411100197','210411100197@student.trunojoyo.ac.id'),
-> ('Maman Subagyo','210411100198','210411100198@student.trunojoyo.ac.id'),
-> ('Nina Puspita','210411100199','210411100199@student.trunojoyo.ac.id'),
-> ('Oki Setiawan','210411100200','210411100200@student.trunojoyo.ac.id'),
-> ('Putu Dwi','210411100201','210411100201@student.trunojoyo.ac.id'),
-> ('Rina Sari','210411100202','210411100202@student.trunojoyo.ac.id'),
-> ('Sari Wulandari','210411100203','210411100203@student.trunojoyo.ac.id'),
-> ('Teguh Pratama','210411100204','210411100204@student.trunojoyo.ac.id'),
-> ('Umar Hidayat','210411100205','210411100205@student.trunojoyo.ac.id');
Query OK, 20 rows affected (0.026 sec)
Records: 20  Duplicates: 0  Warnings: 0

MariaDB [21_mahasiswadatabase]> 
```

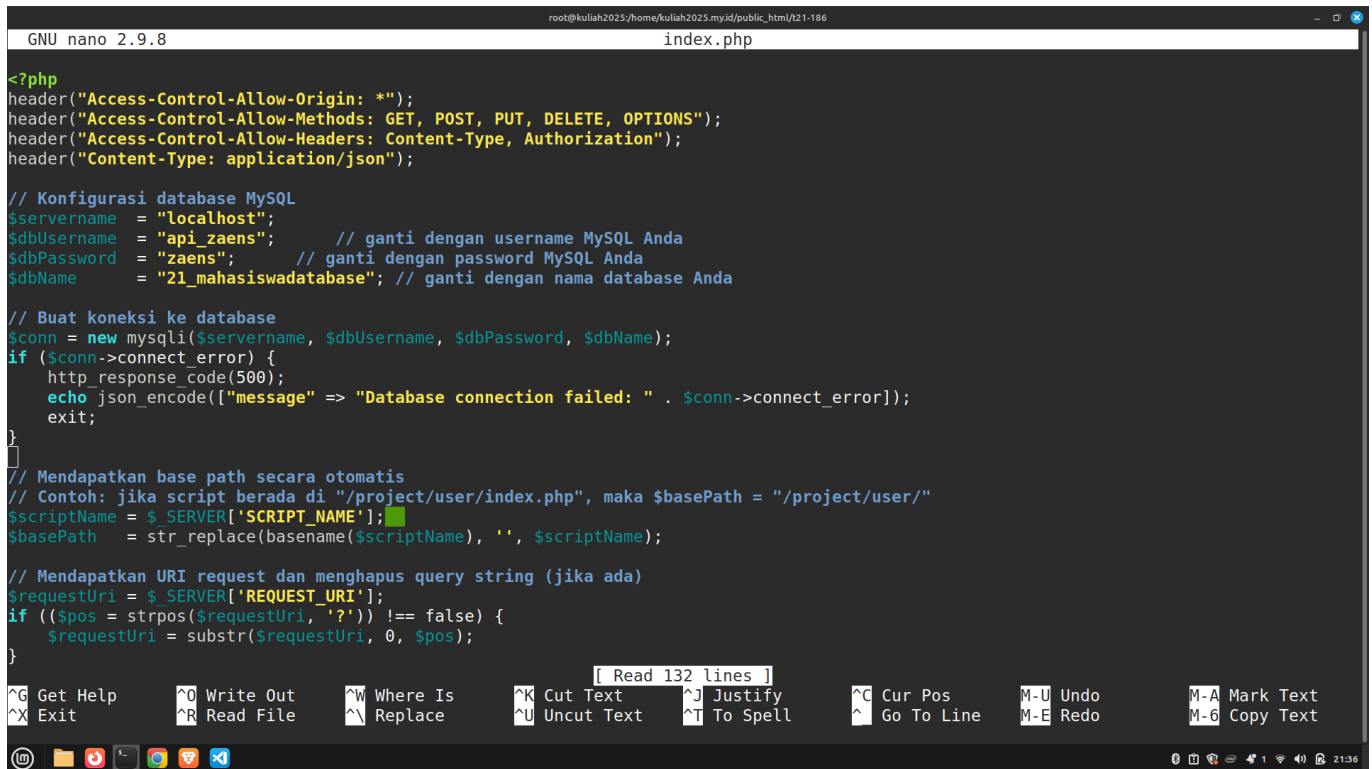
setelah membuat database, table mahasiswa dan insert mahasiswa berhasil selanjutnya membuat code yang berintegrasi dengan database mahasiswa.

code yang terintegrasi dengan database

buat file menggunakan nano dengan perintah berikut

```
nano index.php
```


lalu isi dengan code api dan hasilnya akan seperti gambar berikut



```
GNU nano 2.9.8 index.php

<?php
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");
header("Access-Control-Allow-Headers: Content-Type, Authorization");
header("Content-Type: application/json");

// Konfigurasi database MySQL
$servername = "localhost";
$dbUsername = "api_zuens"; // ganti dengan username MySQL Anda
$dbPassword = "zaens"; // ganti dengan password MySQL Anda
$dbName = "21_mahasiswadatabase"; // ganti dengan nama database Anda

// Buat koneksi ke database
$conn = new mysqli($servername, $dbUsername, $dbPassword, $dbName);
if ($conn->connect_error) {
    http_response_code(500);
    echo json_encode(["message" => "Database connection failed: " . $conn->connect_error]);
    exit;
}

// Mendapatkan base path secara otomatis
// Contoh: jika script berada di "/project/user/index.php", maka $basePath = "/project/user/"
$scriptName = $_SERVER['SCRIPT_NAME'];
$basePath = str_replace(basename($scriptName), '', $scriptName);

// Mendapatkan URI request dan menghapus query string (jika ada)
$requestUri = $_SERVER['REQUEST_URI'];
if (($pos = strpos($requestUri, '?')) !== false) {
    $requestUri = substr($requestUri, 0, $pos);
}

[ Read 132 lines ]
^G Get Help      ^O Write Out     ^W Where Is      ^K Cut Text      ^J Justify       ^C Cur Pos       M-U Undo        M-A Mark Text
^X Exit          ^R Read File     ^_ Replace       ^U Uncut Text    ^T To Spell      ^_ Go To Line    M-E Redo        M-6 Copy Text
```

code api dari index.php adalah

```
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");
header("Access-Control-Allow-Headers: Content-Type, Authorization");
header("Content-Type: application/json");

// Konfigurasi database MySQL
$servername = "localhost";
$dbUsername = "zaens"; // ganti dengan username MySQL Anda
$dbPassword = "zaens"; // ganti dengan password MySQL Anda
$dbName = "mahasiswa"; // ganti dengan nama database Anda

// Buat koneksi ke database
$conn = new mysqli($servername, $dbUsername, $dbPassword, $dbName);
if ($conn->connect_error) {
    http_response_code(500);
    echo json_encode(["message" => "Database connection failed: " . $conn->connect_error]);
    exit;
}

// Mendapatkan base path secara otomatis
// Contoh: jika script berada di "/project/user/index.php", maka $basePath = "/project/user/"
$scriptName = $_SERVER['SCRIPT_NAME'];
$basePath = str_replace(basename($scriptName), '', $scriptName);

// Mendapatkan URI request dan menghapus query string (jika ada)
$requestUri = $_SERVER['REQUEST_URI'];
```

```

if (($pos = strpos($requestUri, '?')) !== false) {
    $requestUri = substr($requestUri, 0, $pos);
}

// Hapus base path dari URI, jika ada
if (strpos($requestUri, $basePath) === 0) {
    $requestUri = substr($requestUri, strlen($basePath));
}

$requestUri = trim($requestUri, '/');
$segments = explode('/', $requestUri);

// Cek apakah URL diawali dengan "users"
if (isset($segments[0]) && $segments[0] === 'mahasiswa') {
    // Optional user ID dari URI (misalnya: /users/1)
    $name = isset($segments[1]) ? intval($segments[1]) : null;

    if ($_SERVER['REQUEST_METHOD'] === 'GET') {
        if ($name !== null) {
            // Mengambil satu user berdasarkan name dengan prepared
statement
            $stmt = $conn->prepare("SELECT * FROM mahasiswa WHERE name =
?");
            $stmt->bind_param("i", $name);
            $stmt->execute();
            $result = $stmt->get_result();

            if ($result->num_rows > 0) {
                $user = $result->fetch_assoc();
                http_response_code(200);
                echo json_encode($user);
            } else {
                http_response_code(404);
                echo json_encode(["message" => "mahasiswa not found"]);
            }
            $stmt->close();
        } else {
            // Mengambil semua user
            $result = $conn->query("SELECT * FROM mahasiswa");
            $users = [];
            while ($row = $result->fetch_assoc()) {
                $users[] = $row;
            }
            http_response_code(200);
            echo json_encode($users);
        }
    } elseif ($_SERVER['REQUEST_METHOD'] === 'POST') {
        // Mendapatkan data yang dikirim
        $data = json_decode(file_get_contents("php://input"), true);
        if (!isset($data["name"]) || !isset($data["nim"]) ||
!isset($data["email"])) {
            http_response_code(400);
            echo json_encode(["message" => "Please, fill name, email!"]);
        } else {

```

```

        // Menambahkan user baru menggunakan prepared statement
        $stmt = $conn->prepare("INSERT INTO mahasiswa (name, nim,
email) VALUES (?, ?, ?)");
        $stmt->bind_param("sss", $data["name"], $data["nim"],
$data["email"]);

        if ($stmt->execute()) {
            $newId = $stmt->insert_id;
            http_response_code(201);
            echo json_encode(["message" => "mahasiswa added
successfully!", "id" => $newId]);
        } else {
            http_response_code(500);
            echo json_encode(["message" => "Error: " . $stmt->error]);
        }
        $stmt->close();
    }
} else {
    http_response_code(405);
    echo json_encode(['error' => 'Method not allowed']);
}
} else if (isset($segments[0]) && $segments[0] === '') {
    header("Content-Type: text/html");
?>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="description" content="SwaggerUI" />
    <title>SwaggerUI</title>
    <link rel="stylesheet" href="https://unpkg.com/swagger-ui-
dist@5.11.0/swagger-ui.css" />
</head>

<body>
    <div id="swagger-ui"></div>
    <script src="https://unpkg.com/swagger-ui-dist@5.11.0/swagger-ui-
bundle.js" crossorigin></script>
    <script>
        window.onload = () => {
            window.ui = SwaggerUIBundle({
                url: 'swagger.json',
                dom_id: '#swagger-ui',
            });
        };
    </script>
</body>

</html>
<?php
}

```

```
?>
```

```
$conn->close();
```

untuk penjelasan code adalah sebagai berikut

```
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS");
header("Access-Control-Allow-Headers: Content-Type, Authorization");
header("Content-Type: application/json");
```

Kode di atas digunakan untuk mengatur header HTTP pada API yang dibuat dengan PHP native. Header "Access-Control-Allow-Origin: *" memungkinkan akses dari semua domain, sedangkan "Access-Control-Allow-Methods" menentukan metode HTTP yang diperbolehkan. Header "Access-Control-Allow-Headers" menetapkan jenis header yang diizinkan (seperti Content-Type dan Authorization), dan "Content-Type: application/json" memastikan bahwa respon dikirim dalam format JSON.

```
// Konfigurasi database MySQL
$servername = "localhost";
$dbUsername = "zaens"; // ganti dengan username MySQL Anda
$dbPassword = "zaens"; // ganti dengan password MySQL Anda
$dbName = "mahasiswa"; // ganti dengan nama database Anda

// Buat koneksi ke database
$conn = new mysqli($servername, $dbUsername, $dbPassword, $dbName);
if ($conn->connect_error) {
    http_response_code(500);
    echo json_encode(["message" => "Database connection failed: " . $conn->connect_error]);
    exit;
}
```

Kode ini untuk menghubungkan server api dengan database MySQL menggunakan PHP native. Pertama, mendefinisikan variabel untuk host, username, password, dan nama database, kemudian dibuat koneksi menggunakan mysqli. Jika koneksi gagal, kode akan mengirimkan HTTP response dengan status 500 beserta pesan error dalam format JSON.

```
// Mendapatkan base path secara otomatis
// Contoh: jika script berada di "/file/user/index.php", maka $basePath = "/file/user/"
$scriptName = $_SERVER['SCRIPT_NAME'];
$basePath = str_replace(basename($scriptName), '', $scriptName);

// Mendapatkan URI request dan menghapus query string (jika ada)
$requestUri = $_SERVER['REQUEST_URI'];
if (($pos = strpos($requestUri, '?')) !== false) {
    $requestUri = substr($requestUri, 0, $pos);
}
```

```

}

// Hapus base path dari URI, jika ada
if (strpos($requestUri, $basePath) === 0) {
    $requestUri = substr($requestUri, strlen($basePath));
}

$requestUri = trim($requestUri, '/');
$segments = explode('/', $requestUri);

```

Kode ini untuk melakukan secara otomatis menentukan base path dari file script, misalnya "/file/user/", dengan menghapus nama file dari \$_SERVER['SCRIPT_NAME']. Selanjutnya, URI request diambil dari \$_SERVER['REQUEST_URI'] dan query string dihapus. Jika URI tersebut mengandung base path, maka bagian tersebut juga dihilangkan, kemudian URI di-trim dan dipecah menjadi segmen-segmen menggunakan explode('/'), yang memudahkan proses routing dalam aplikasi API.

```

if (isset($segments[0]) && $segments[0] === 'mahasiswa') {

```

Kode ini memeriksa apakah segmen pertama dari URI ada dan bernilai "mahasiswa". Jika kondisi ini terpenuhi, maka aplikasi akan menjalankan logika khusus untuk menangani request yang berkaitan dengan entitas mahasiswa.

```

else if (isset($segments[0]) && $segments[0] === '') {
    header("Content-Type: text/html");
?>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="description" content="SwaggerUI" />
    <title>SwaggerUI</title>
    <link rel="stylesheet" href="https://unpkg.com/swagger-ui-
dist@5.11.0/swagger-ui.css" />
</head>

<body>
    <div id="swagger-ui"></div>
    <script src="https://unpkg.com/swagger-ui-dist@5.11.0/swagger-ui-
bundle.js" crossorigin></script>
    <script>
        window.onload = () => {
            window.ui = SwaggerUIBundle({
                url: 'swagger.json',
                dom_id: '#swagger-ui',
            });
        };
    </script>

```

```

    </script>
  </body>

</html>
<?php
}

```

Kode ini memeriksa apakah segmen pertama dari URI adalah kosong. Jika kondisi tersebut terpenuhi, maka header konten diatur menjadi "text/html" dan halaman HTML yang berisi SwaggerUI ditampilkan. Halaman ini memuat file konfigurasi Swagger (swagger.json) untuk menampilkan dokumentasi API secara interaktif.

```
$name = isset($segments[1]) ? $segments[1] : null;
```

Baris kode ini memeriksa apakah segmen kedua dari URI ada. Jika ada, nilainya diubah menjadi string dan disimpan dalam variabel \$name. Jika tidak ada, maka \$name akan bernilai null.

```

if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    if ($name !== null) {
        // Mengambil satu user berdasarkan name dengan prepared statement
        $stmt = $conn->prepare("SELECT * FROM mahasiswa WHERE name = ?");
        $stmt->bind_param("i", $name);
        $stmt->execute();
        $result = $stmt->get_result();

        if ($result->num_rows > 0) {
            $user = $result->fetch_assoc();
            http_response_code(200);
            echo json_encode($user);
        } else {
            http_response_code(404);
            echo json_encode(["message" => "mahasiswa not found"]);
        }
        $stmt->close();
    } else {
        // Mengambil semua user
        $result = $conn->query("SELECT * FROM mahasiswa");
        $users = [];
        while ($row = $result->fetch_assoc()) {
            $users[] = $row;
        }
        http_response_code(200);
        echo json_encode($users);
    }
}

```

Kode di atas menangani permintaan HTTP GET. Jika variabel \$name tidak null, maka akan dilakukan query dengan prepared statement untuk mengambil satu data mahasiswa berdasarkan name; jika data ditemukan,

akan dikembalikan dalam format JSON dengan status 200, dan jika tidak, dikembalikan pesan error dengan status 404. Sebaliknya, jika \$name bernilai null, maka semua data mahasiswa akan diambil dan dikembalikan sebagai JSON dengan status 200.

```
elseif ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Mendapatkan data yang dikirim
    $data = json_decode(file_get_contents("php://input"), true);
    if (!isset($data["name"]) || !isset($data["nim"]) ||
    !isset($data["email"])) {
        http_response_code(400);
        echo json_encode(["message" => "Please, fill name, email!"]);
    } else {
        // Menambahkan user baru menggunakan prepared statement
        $stmt = $conn->prepare("INSERT INTO mahasiswa (name, nim, email)
VALUES (?, ?, ?)");
        $stmt->bind_param("sss", $data["name"], $data["nim"],
        $data["email"]);

        if ($stmt->execute()) {
            $newId = $stmt->insert_id;
            http_response_code(201);
            echo json_encode(["message" => "mahasiswa added successfully!",
            "id" => $newId]);
        } else {
            http_response_code(500);
            echo json_encode(["message" => "Error: " . $stmt->error]);
        }
        $stmt->close();
    }
}
```

Kode di atas menangani permintaan HTTP POST untuk menambahkan data mahasiswa. Pertama, data JSON yang dikirimkan didekode menjadi array asosiatif. Jika field "name", "nim", atau "email" tidak ada, maka server mengembalikan respons 400 beserta pesan error. Jika data lengkap, maka prepared statement digunakan untuk menyisipkan data baru ke tabel mahasiswa. Apabila eksekusi berhasil, server merespons dengan status 201 dan mengirimkan ID mahasiswa yang baru ditambahkan; jika gagal, akan mengembalikan status 500 dengan pesan error.

```
else {
    http_response_code(405);
    echo json_encode(['error' => 'Method not allowed']);
}
```

Kode di atas merupakan kondisi fallback yang menangani permintaan dengan metode HTTP yang tidak diizinkan. Jika request method tidak sesuai dengan kondisi yang telah ditangani (misalnya, GET atau POST), maka server akan mengembalikan response dengan status code 405 ("Method not allowed") dan pesan error dalam format JSON

pengujian api menggunakan curl dan postman

untuk menguji api url yang digunakan adalah `https://kuliah2025.my.id/t21-186/mahasiswa/{nama mahasiswa}`

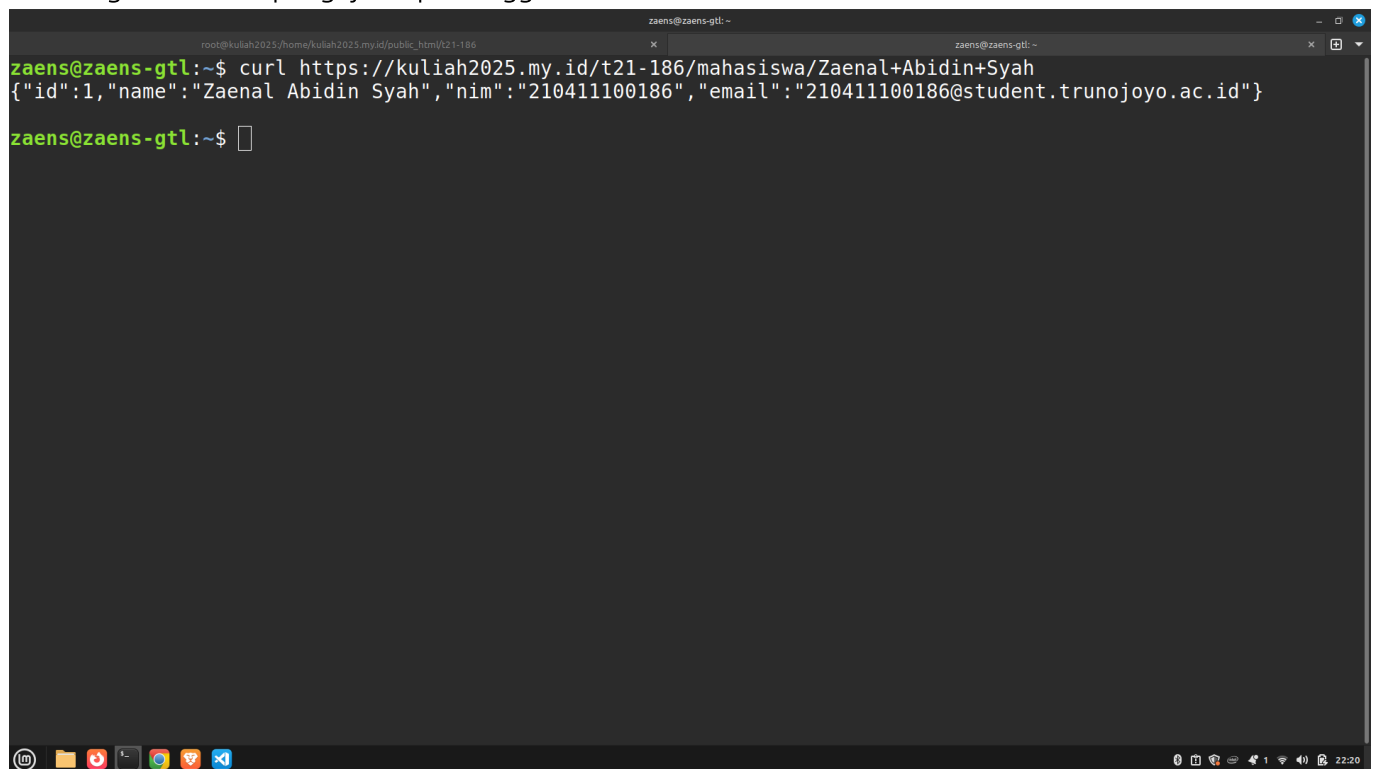
untuk menggunakan curl ketikkan perintah berikut

```
curl https://kuliah2025.my.id/t21-186/mahasiswa/Zaenal+Abidin+Syah
```

akan didapatkan hasil seperti berikut

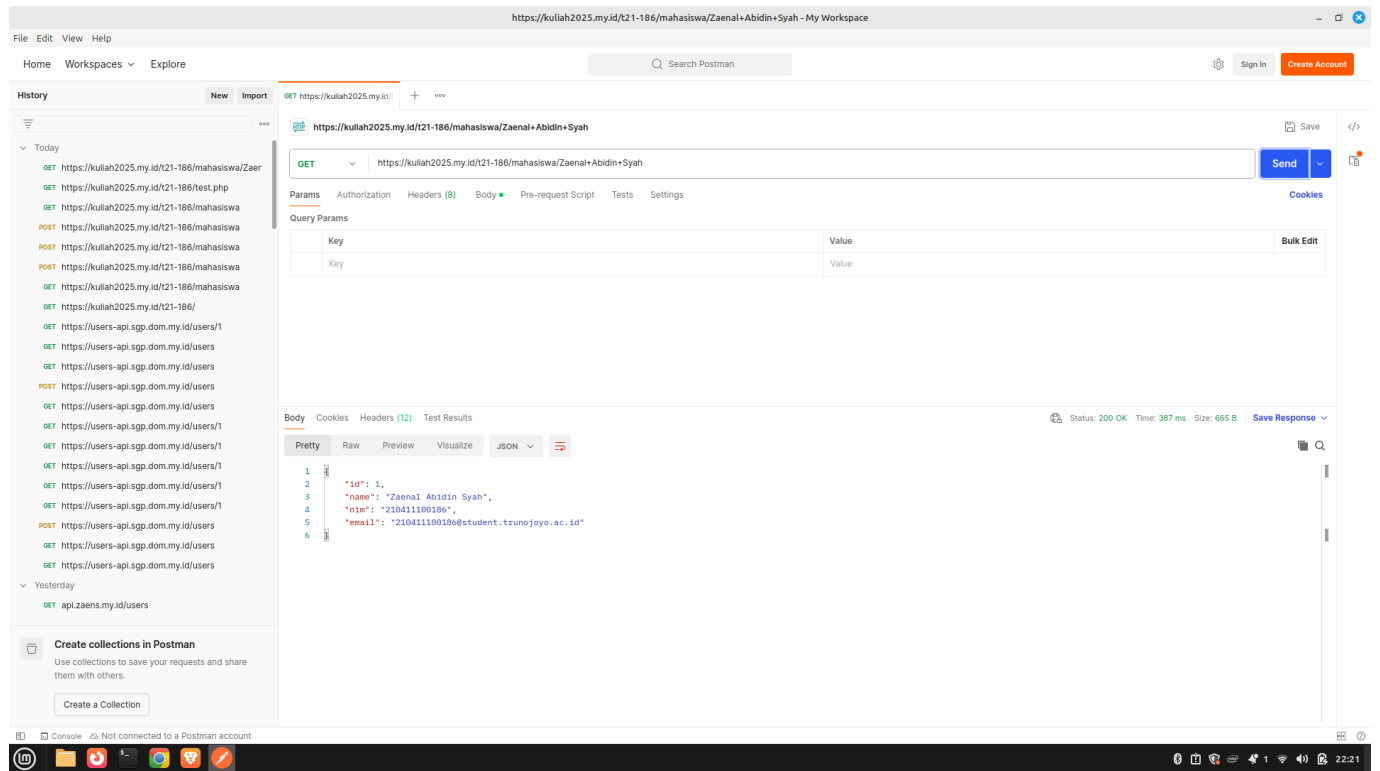
```
{
  "id": 1,
  "name": "Zaenal Abidin Syah",
  "nim": "210411100186",
  "email": "210411100186@student.trunojoyo.ac.id"
}
```

berikut gambar hasil pengujian api menggunakan curl

A screenshot of a Linux terminal window. The prompt is `zaens@zaens-gtl:~$`. The user has entered the command `curl https://kuliah2025.my.id/t21-186/mahasiswa/Zaenal+Abidin+Syah`. The terminal output shows the JSON response: `{"id":1,"name":"Zaenal Abidin Syah","nim":"210411100186","email":"210411100186@student.trunojoyo.ac.id"}`. The prompt is now `zaens@zaens-gtl:~$` with a cursor. The terminal window has a dark background and a standard Linux desktop environment at the bottom with various icons and a system clock showing 22:20.

untuk menggunakan postman buka aplikasi postman masukan url test api `https://kuliah2025.my.id/t21-186/mahasiswa/Zaenal+Abidin+Syah`, lalu pilih metode get lalu tekan send untuk mengirim request ke api.

untuk pengujiannya dapat dilihat pada gambar berikut.



untuk melihat response dari server bisa dilihat di bagian bawah. server mengirimkan response json seperti berikut

```
{
  "id": 1,
  "name": "Zaenal Abidin Syah",
  "nim": "210411100186",
  "email": "210411100186@student.trunojoyo.ac.id"
}
```