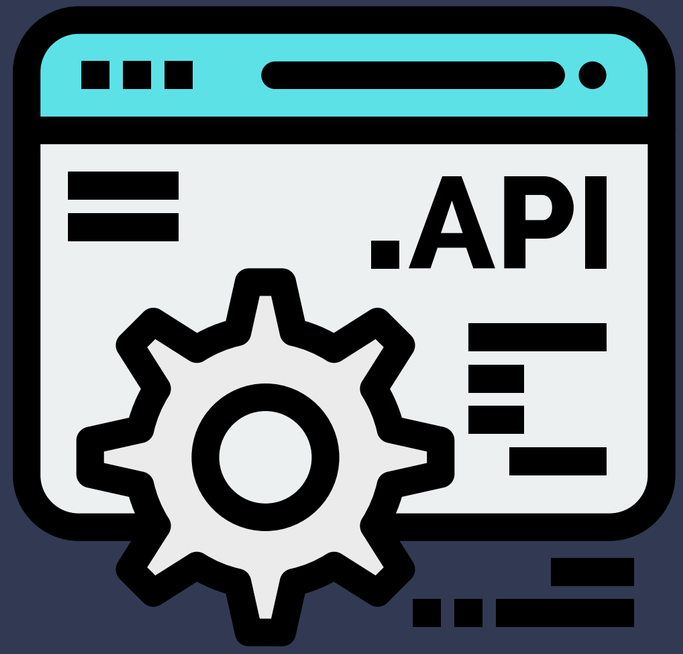
Jefri Maruli

Back-End Development

Materi 5



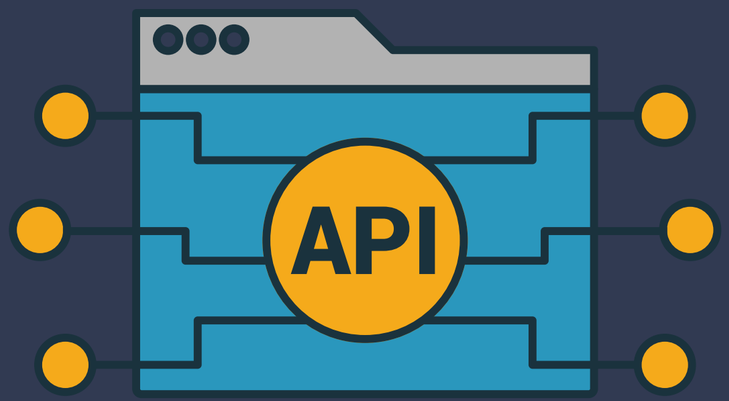
Agenda Belajar

- REST API (Representational State Transfer Application Programming Interface)

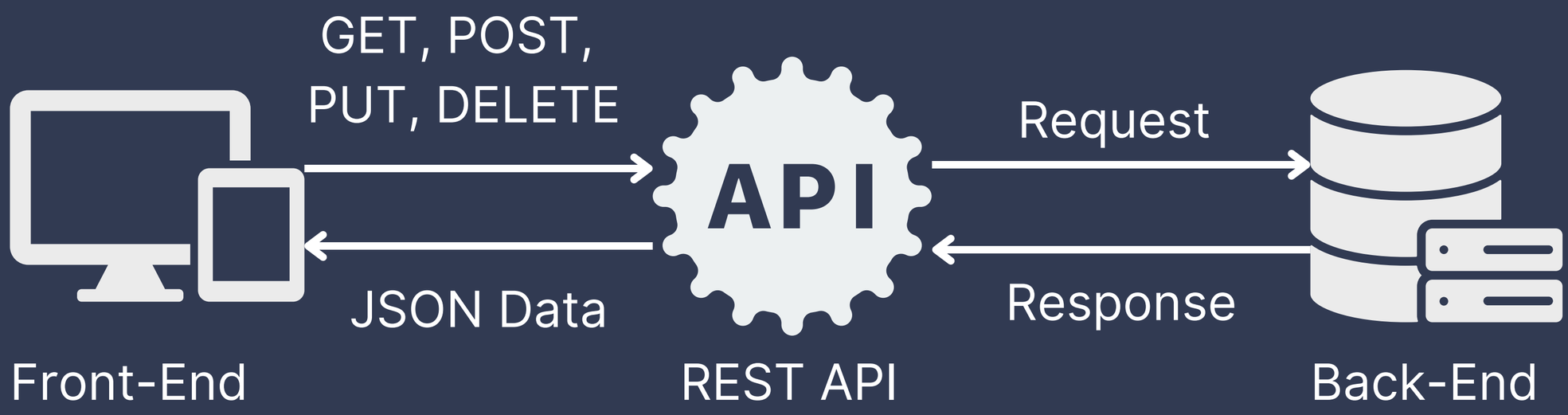
referensi : https://belajaraplikasi.com/membuat-rest-api-dengan-php-dan-mysql

REST API

REST API (Representational State Transfer Application Programming Interface) merupakan implementasi dari API yang memiliki suatu arsitektur metode komunikasi dengan menggunakan protokol HTTP untuk pertukaran Data. Metode ini sering diterapkan dalam pengembangan aplikasi yang bertujuan untuk menjadikan sistem yang memiliki performa yang baik, cepat dan mudah untuk di kembangkan terutama dalam pertukaran dan komunikasi data.



Konsep REST API

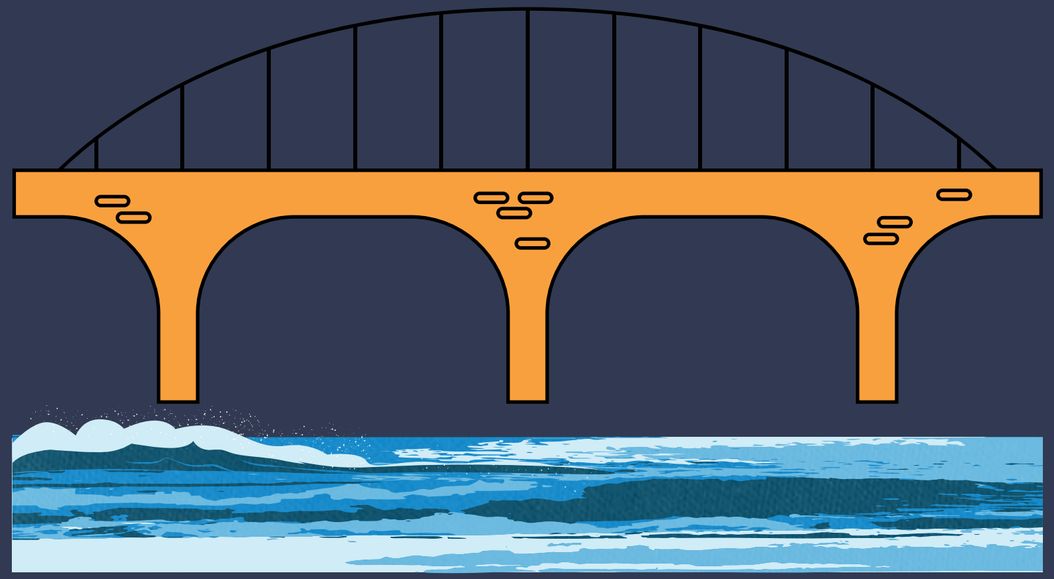


Analogi REST API

Kota A (FE)

Mobil (Request)

Jembatan (API)



Kota B (BE)

Analogi Development API

Kompor
(Server)

Koki
(BE Developer)

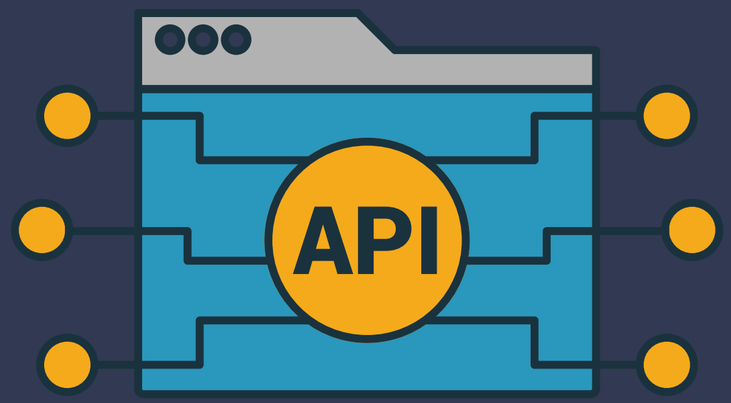
Pelayan
(FE Developer)

```
{
  "name": "Sandwich",
  "type": "Hidangan yang terbuat dari dua potong roti atau lebih dengan satu atau lebih isian",
  "varieties": [
    {
      "name": "Grilled Cheese",
      "description": "Sandwich yang terbuat dari satu atau lebih jenis keju di atas roti yang biasanya diolesi mentega dan dipanggang"
    },
    {
      "name": "Philly Cheesesteak",
      "description": "Sandwich yang terbuat dari steak yang diiris tipis dan keju leleh di atas gulungan panjang"
    }
  ]
}
```

Komponen REST API

REST API memiliki komponen penting sebagai syarat utama mengimplementasikan REST API, dan akan disebut RESTful API jika melakukan semua komponen yang ada di bawah ini:

- URL Design
- HTTP Verbs
- HTTP Response Code
- Format Response



Perbedaan API , REST API & RESTful API

Komponen	API	REST API	RESTful API
URL Design	Tidak	Tidak	Ya
HTTP Verbs	Tidak	Tidak	Ya
HTTP Response Code	Tidak	Ya	Ya
Format Response	Ya	Ya	Ya

URL Design

REST API diakses menggunakan protokol HTTP. Penamaan dan struktur URL yang konsisten akan menghasilkan API yang baik dan mudah untuk dimengerti developer. URL API biasa disebut endpoint dalam pemanggilannya. Contoh penamaan URL / endpoint yang baik adalah seperti berikut:

/users
/users/1234
/users/store/
/users/update

```
http://localhost/mardirabootcamp/users
```

Perbandingan URL Design

Tidak Menggunakan URL Design

```
http://localhost/mardirabootcamp/PDO/Users.php
```

Menggunakan URL Design

```
http://localhost/mardirabootcamp/users
```

HTTP Verbs

Setiap request yang dilakukan terdapat metode yang dipakai agar server mengerti apa yang sedang di request client atau dari front-end, diantaranya yang umum dipakai adalah:

HTTP	Deskrpsi	Contoh
GET	GET adalah metode HTTP Request yang paling simpel, metode ini digunakan untuk membaca atau mendapatkan data dari sumber	/users
POST	POST adalah metode HTTP Request yang digunakan untuk membuat data baru dengan menyisipkan data dalam body saat request dilakukan.	/users/store
PUT	PUT adalah metode HTTP Request yang biasanya digunakan untuk melakukan update data resource.	/users/update/:id
DELETE	DELETE adalah metode HTTP Request yang digunakan untuk menghapus suatu data pada resource.	/users/delete/:id

HTTP Response Code

HTTP response code adalah kode standarisasi dalam menginformasikan hasil request kepada client / Front-End. Secara umum terdapat 3 kelompok yang biasa kita jumpai pada RESTful API yaitu:

- 2xx : adalah response code yang menampilkan bahwa request berhasil
- 4xx : adalah respose code yang menampilkan bahwa request mengalami kesalahan pada sisi client / front-end
- 5xx : adalah response code yang menampilkan bahwa kesalahan pada sisi server / back-end

Contoh HTTP Response Code

- **200 (OK)** : Response code berhasil.

- **201 (Created)** : Response code data telah berhasil dibuat (POST/PUT).

- **400 (Bad Request)** : Response code request yang dibuat salah / tidak ada.

- **401 (Unauthorized)** : Response code request yang dibuat membutuhkan authentication (login) sebelum mengakses endpoint tersebut.

- **403 (Forbidden)** : Response code yang bukan memiliki hak untuk mengakses endpoint tersebut meskipun sudah melakukan login.

- **404 (Not Found)** : Response code request yang dicari tidak ditemukan.

- **405 (Method Not Allowed) :** Response code request endpoint ada tetapi HTTP verbs yang digunakan tidak diizinkan, contoh : endpoint yang dibuat menggunakan HTTP GET tetapi saat di request menggunakan HTTP POST

Contoh HTTP Response Code

- **409 (Conflict)** : Response code request yang dibuat terdapat duplikasi, biasanya data yang dikirim sudah ada sebelumnya.

- **500 (Internal Server Error)** : Response code ini menandakan bahwa request yang dilakukan terdapat kesalahan pada sisi server atau resource.

- **503 (Service Temporarily Unavailable)** : Response code untuk menandakan bahwa server sedang mengalami perbaikan (maintenance).

Referensi status code : https://id.wikipedia.org/wiki/Daftar_kode_status_HTTP

Format Response

Setiap request yang dilakukan client akan menerima data response dari server, response tersebut biasanya berupa data XML (eXtensible Markup Language) ataupun JSON (JavaScript Object Notation). Setelah mendapatkan data response tersebut barulah client bisa menggunakannya dengan cara parsing data tersebut dan diolah sesuai dengan kebutuhan.

Format JSON

```json
{
  "data": [
    {
      "id": 1,
      "name": "John Doe",
      "email": "johndoe@gmail.com"
    },
    {
      "id": 2,
      "name": "Jane Doe",
      "email": "janedoe@gmail.com"
    }
  ]
}
```

Format XML

```xml
<users>
  <user>
    <id>1</id>
    <name>John Doe</name>
    <email>johndoe@gmail.com</email>
  </user>
  <user>
    <id>2</id>
    <name>Jane Doe</name>
    <email>janedoe@gmail.com</email>
  </user>
</users>
```

Pembuatan Format Response API

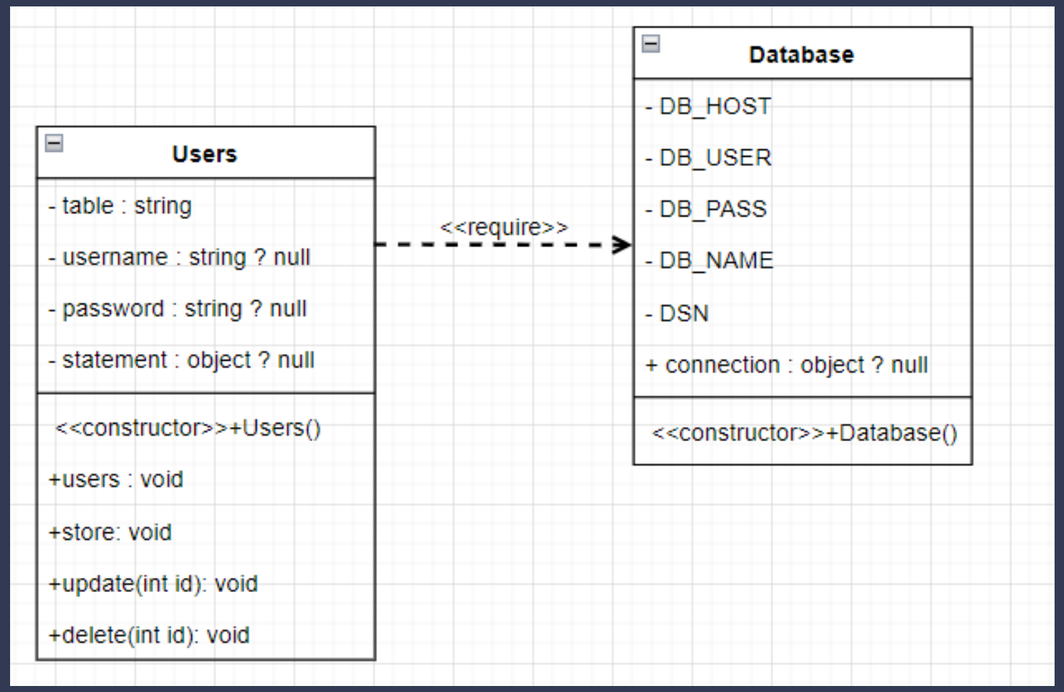
XML

```php

<?php
$users = [
    [
        'id' => 1,
        'name' => 'John Doe',
        'email' => 'johndoe@gmail.com'
    ],
    [
        'id' => 2,
        'name' => 'Jane Smith',
        'email' => 'janedoe@gmail.com'
    ]
];

function array_to_xml($array, &$xml_user_info)
{
    foreach ($array as $key => $value) {
        if (is_array($value)) {
            if (!is_numeric($key)) {
                $subnode = $xml_user_info->addChild("$key");
                array_to_xml($value, $subnode);
            } else {
                $subnode = $xml_user_info->addChild("$key");
                array_to_xml($value, $subnode);
            }
        } else {
            $xml_user_info->addChild("$key", htmlspecialchars("$value"));
        }
    }
}

$xml = new SimpleXMLElement("<?xml version=\"1.0\"?><users></users>");

//function call to convert array to xml
array_to_xml($users,$xml);

echo $xml->asXML();

```

JSON

```php
<?php
$users = [
    [
        'id' => 1,
        'name' => 'John Doe',
        'email' => 'johndoe@gmail.com'
    ],
    [
        'id' => 2,
        'name' => 'Jane Smith',
        'email' => 'janedoe@gmail.com'
    ]
];

echo json_encode($users);
```

Implementasi RESTful API



Untuk mengimplementasi RESTful API, kita tetap menggunakan studi kasus pada materi sebelumnya yang sudah dirancang. Tugas utama untuk membuat RESTful API adalah:

- URL Design
- HTTP Verbs
- HTTP Response Code
- Format Response

No	Field	Type	Size	Index
1	user_id	int	11	Primary Key
2	username	varchar	25	
3	password	varchar	64	

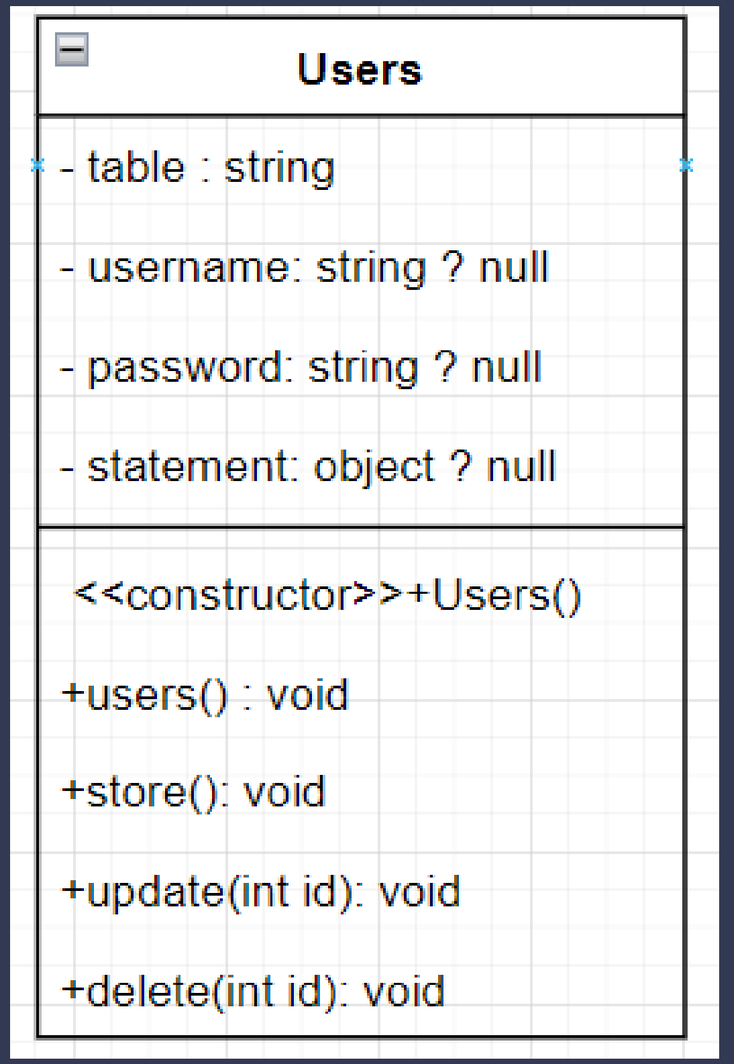
Penerapan Koneksi

```php
<?php

class Database
{
    private const DB_HOST  = 'localhost';
    private const DB_USER = 'root';
    private const DB_PASS = '';
    private const DB_NAME = 'pdo';
    private const DSN = 'mysql:host=' . self::DB_HOST . ';dbname=' . self::DB_NAME . '';
    public ?object $connection = null;

    public function __construct()
    {
        try {
            $this->connection = new PDO(
                self::DSN,
                self::DB_USER,
                self::DB_PASS
            );
            $this->connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            echo $e->getMessage();
        }
        return $this->connection;
    }
}
```

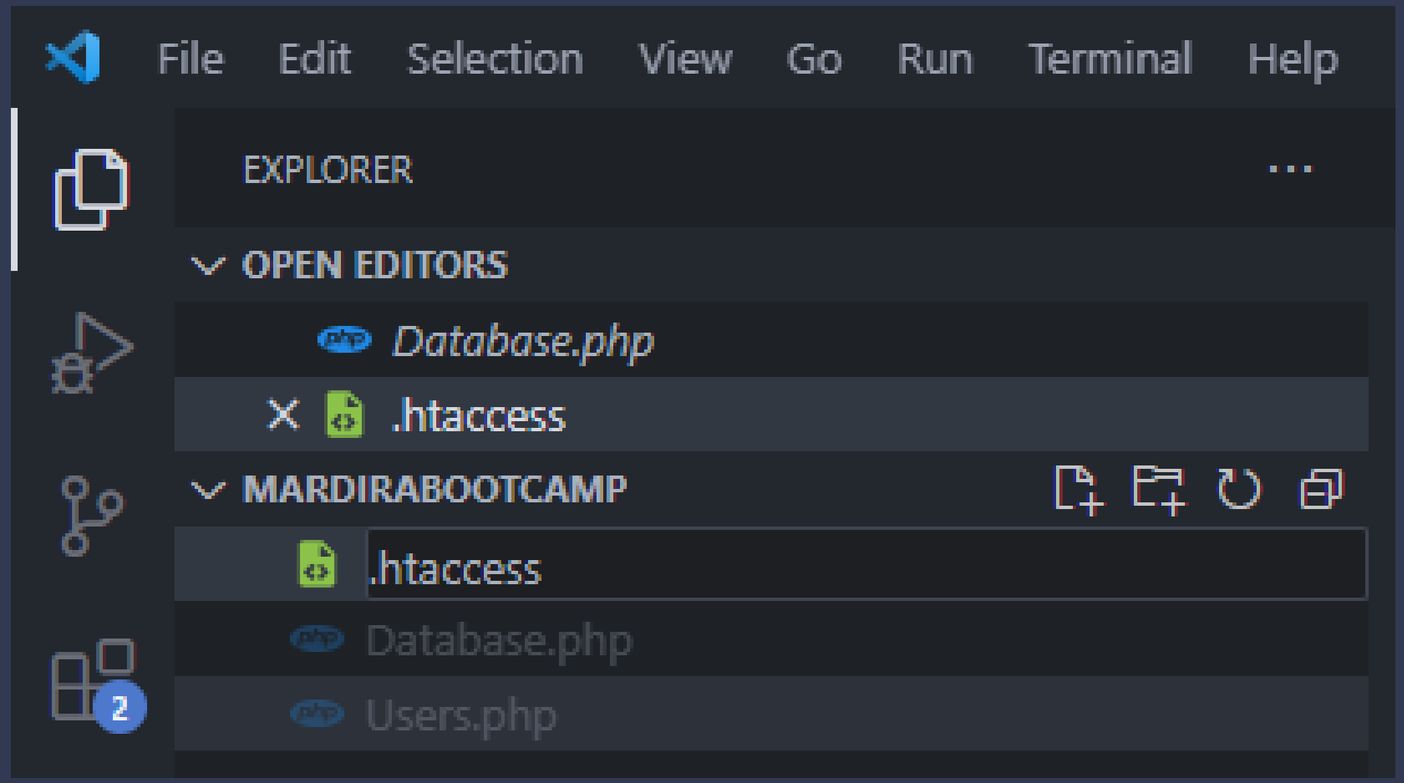
Penerapan Class Users Dari Class Diagram

Class Users secara keseluruhan dari class diagram

```php
<?php

require_once('Database.php');

class Users
{
    private string $table = 'users';
    private ?object $statement;
    private ?string $username;
    private ?string $password;

    public function __construct()
    {
        $this->statement = new Database();
        $this->statement = $this->statement->connection;
        $this->username = $_POST['username'] ?? null;
        $this->password = $_POST['password'] ?? null;
    }

    public function users(): void
    {
        $query = "SELECT * FROM $this->table";
        $statement = $this->statement->prepare($query);
        $statement->execute();
        $users = $statement->fetchAll(PDO::FETCH_OBJ);
        $total = $statement->rowCount();
        $response = [
            'message' => 'Data User',
            'total' => $total,
            'data' => $users
        ];
        echo json_encode($response, JSON_PRETTY_PRINT);
    }

    public function store(): void
    {
        $query = "SELECT * FROM {$this->table} WHERE username = :username";
        $statement = $this->statement->prepare($query);
        $statement->bindParam(':username', $this->username);
        $statement->execute();
        if ($statement->rowCount() > 0) {
            $response = [
                'message' => 'Data User Sudah Ada!',
            ];
            http_response_code(409);
        } else {
            $query = "INSERT INTO {$this->table} (username, password) VALUES (:username, :password)";
            $statement = $this->statement->prepare($query);
            $password = md5($this->password);
            $statement->bindParam(':username', $this->username);
            $statement->bindParam(':password', $password);
            $statement->execute();
            $getUsers = "SELECT * FROM users WHERE username = :username";
            $statement = $this->statement->prepare($getUsers);
            $statement->bindParam(':username', $this->username);
            $statement->execute();
            $user = $statement->fetch(PDO::FETCH_OBJ);
            $response = [
                'message' => 'Data User Berhasil Disimpan',
                'user' => $user,
            ];
        }
        echo json_encode($response);
    }

    public function update(int $id): void
    {
        $query = "UPDATE {$this->table} SET username = :username, password = :password WHERE user_id = :user_id";
        $statement = $this->statement->prepare($query);
        $password = md5($this->password);
        $statement->bindParam(':username', $this->username);
        $statement->bindParam(':password', $this->password);
        $statement->bindParam(':user_id', $id);
        $statement->execute();
        $response = [
            'message' => 'Data User Berhasil Diubah',
        ];
        echo json_encode($response);
    }

    public function delete(int $id): void
    {
        $query = "DELETE FROM {$this->table} WHERE user_id = :user_id";
        $statement = $this->statement->prepare($query);
        $statement->bindParam(':user_id', $id);
        $statement->execute();
        $response = [
            'message' => 'Data User Berhasil Dihapus',
        ];
        echo json_encode($response);
    }
}

$users = new Users();
$users->users();
```



Menerapkan URL Design Dengan .htaccess



Todo

- URL Design
- HTTP Verbs
- HTTP Response Code
- Format Response

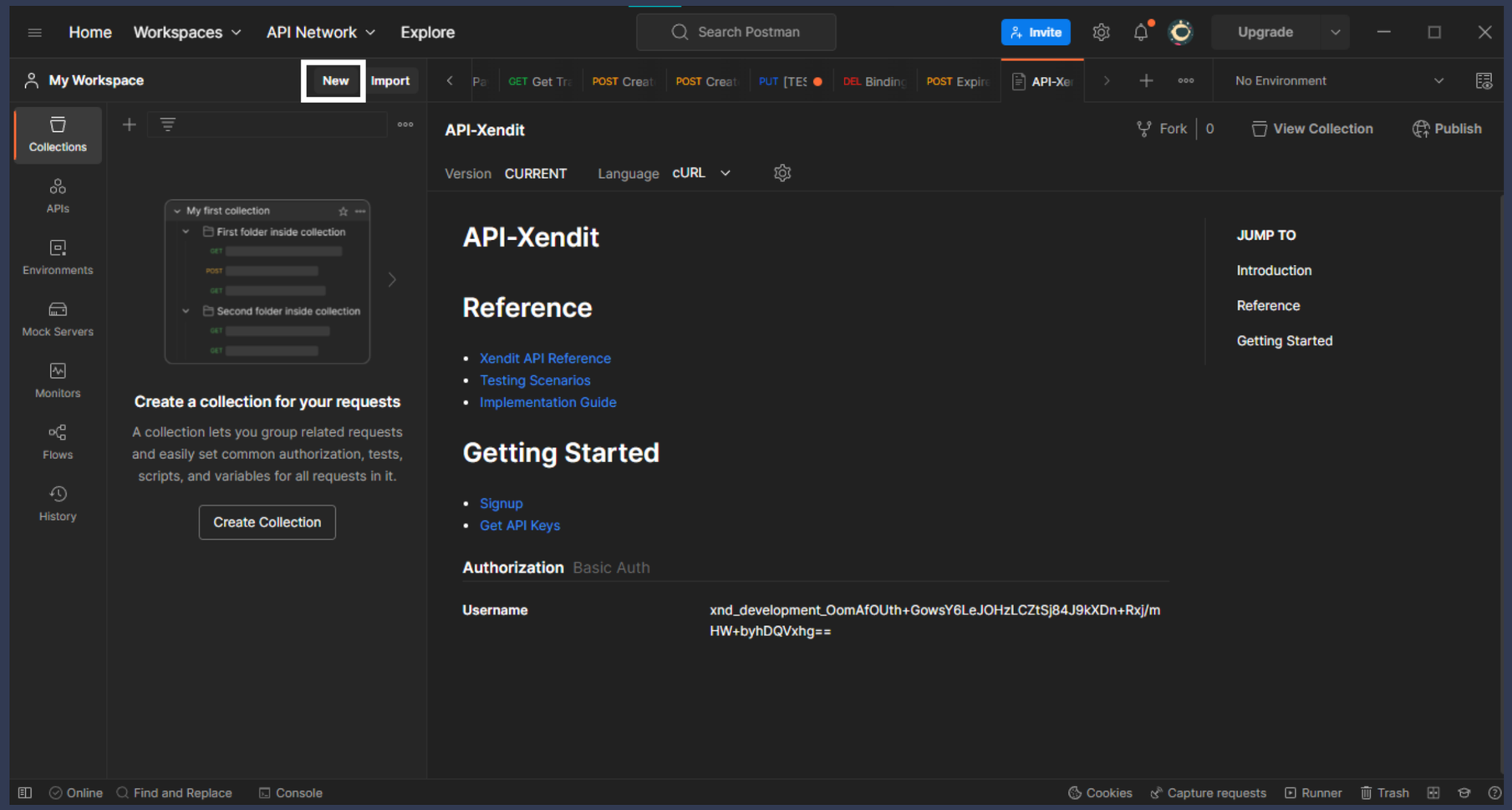
Tambahkan file .htaccess atau hypertext access untuk mengatur bagaimana mengatur sebuah script php yang berada dalam sebuah server/hosting

Isi File .htaccess

```
RewriteEngine On # Mengaktifkan modul mod_rewrite
# Mengarahkan URL ke file Users.php
RewriteRule ^users/?$ Users.php [NC,L]
# Mengarahkan URL ke file Users.php dengan parameter id
RewriteRule ^Users/([0-9]+)/?$ users.php?id=$1 [NC,L]
```

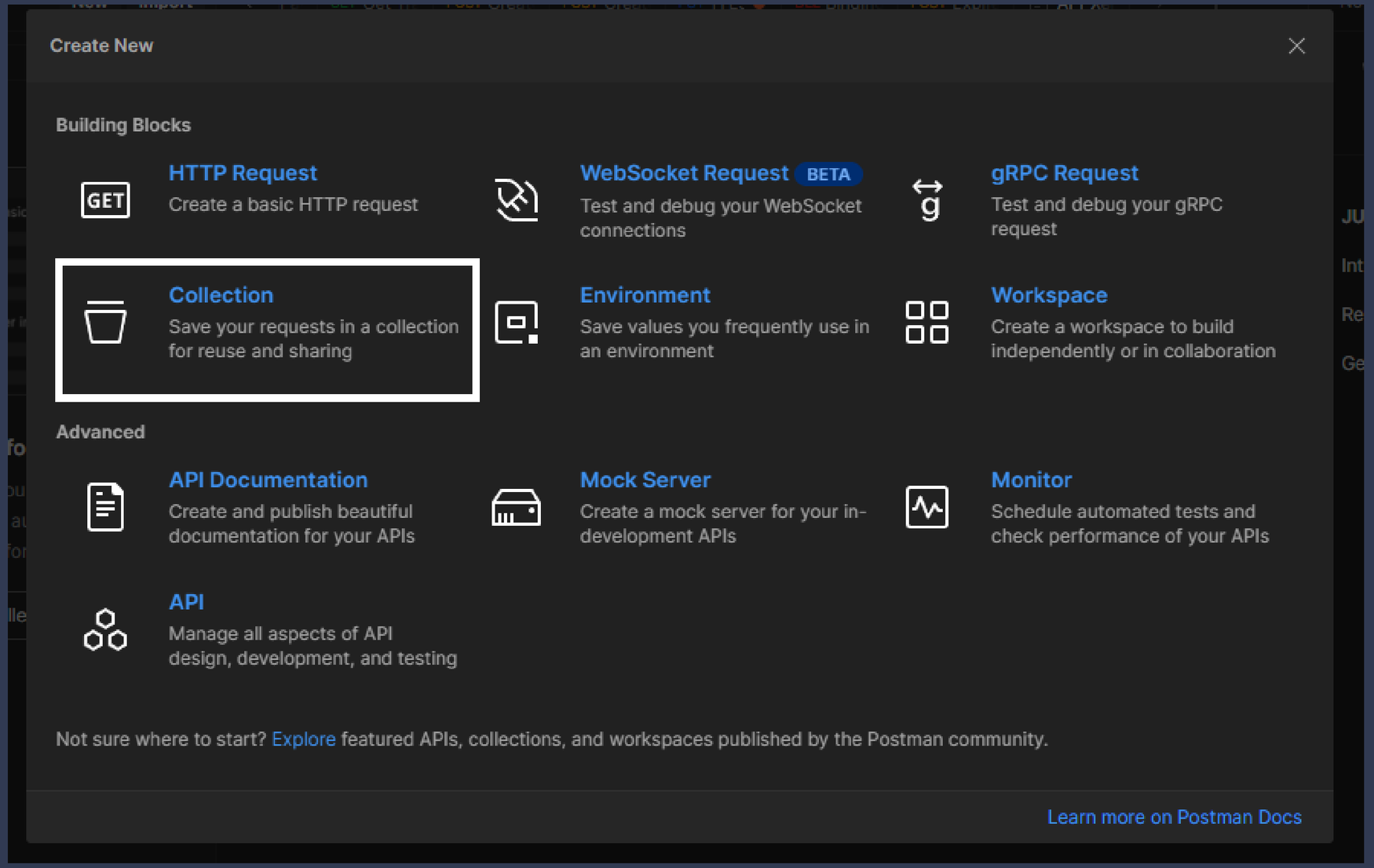
Tambahkan baris kode di atas untuk file .htaccess supaya bisa menggunakan clean URL design pada REST API.

Menggunakan Postman



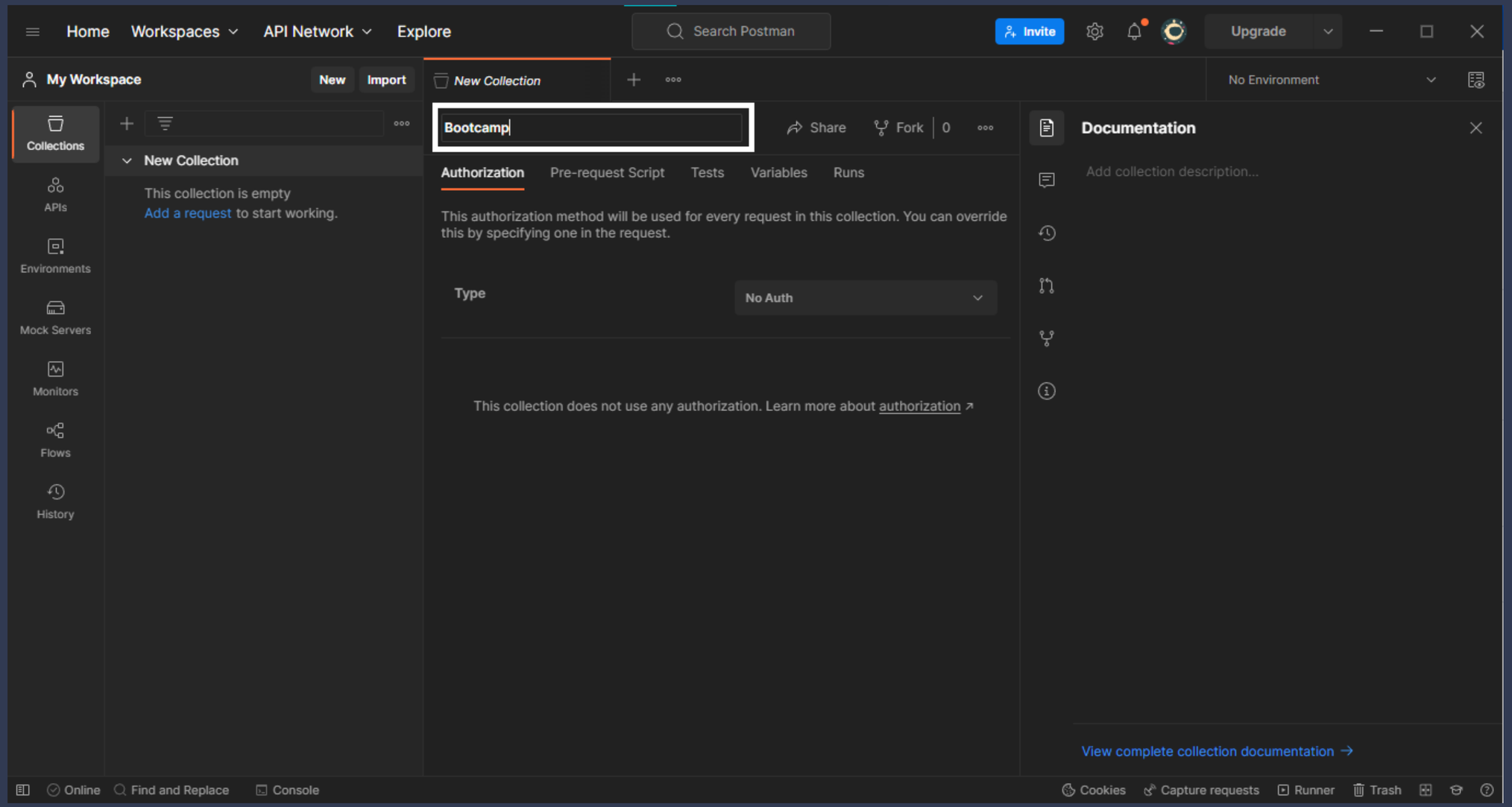
Kita akan menggunakan Postman untuk melakukan testing API yang telah dibuat, pertama setelah membuka Postman pilih tombol new

Menambahkan Collection



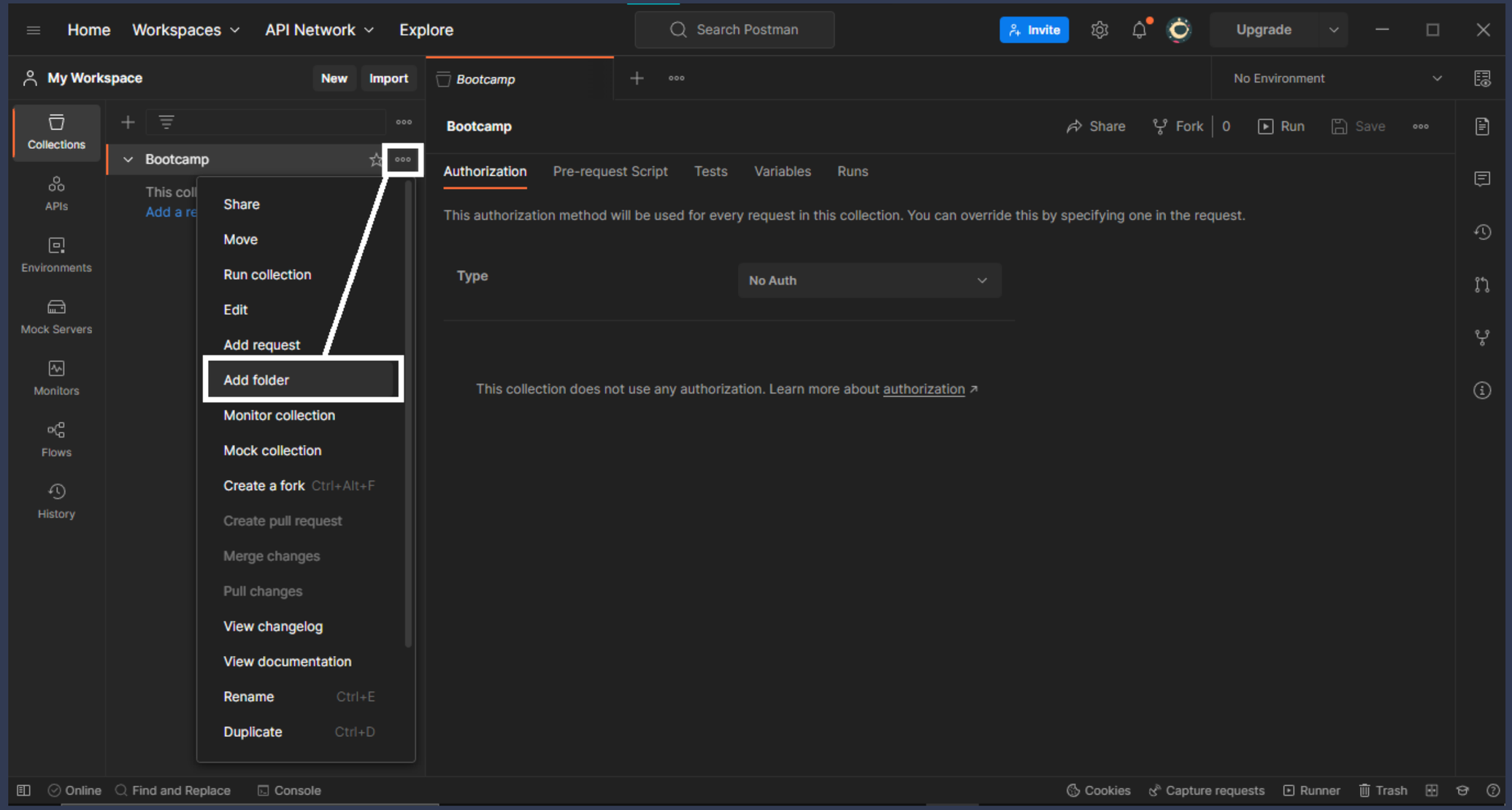
Setelah memilih tombol new kemudian pilih "Collection".

Mengubah Nama Collection



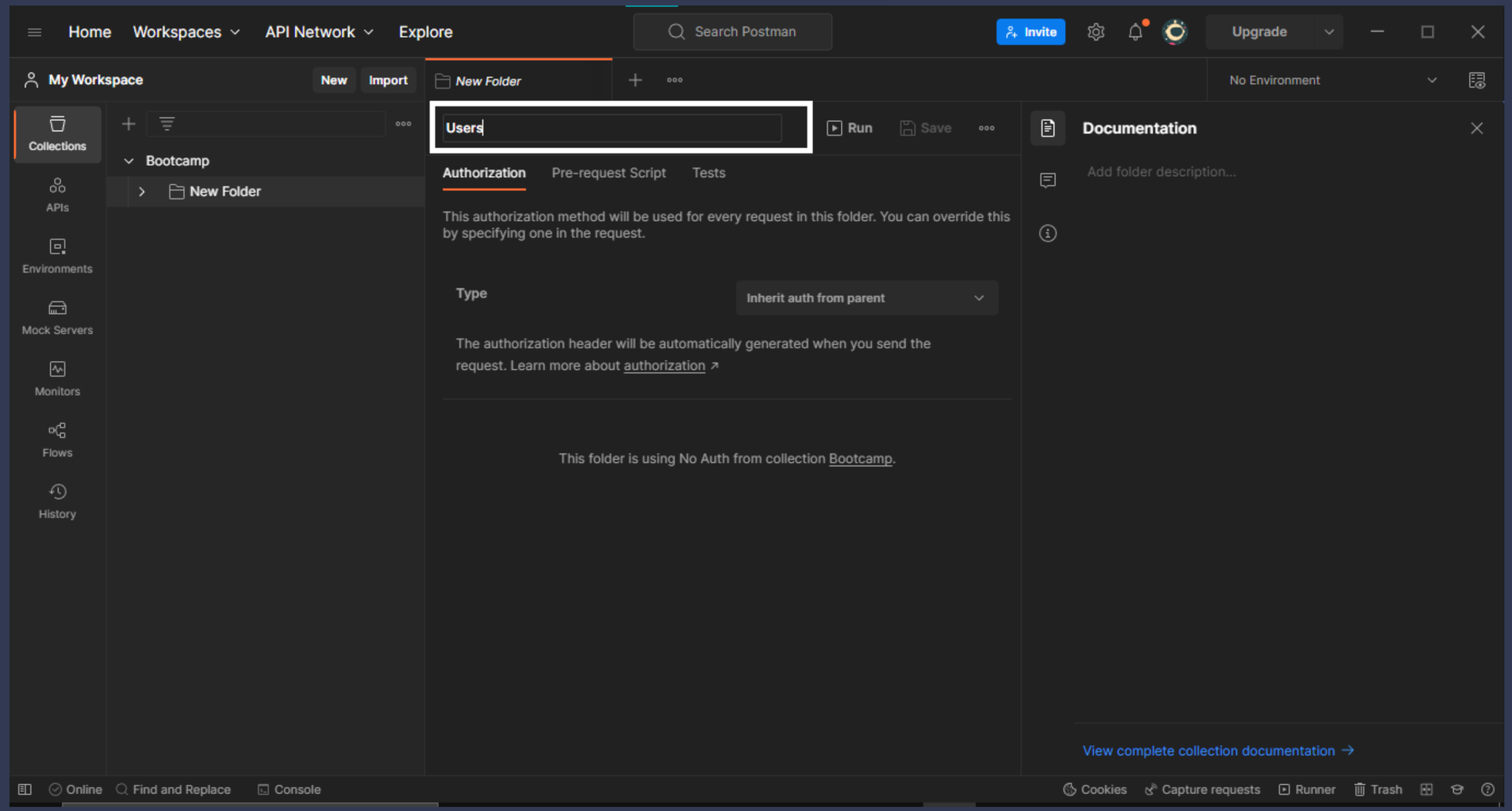
Selanjutnya rename nama collection menjadi "Bootcamp".

Menambahkan Folder



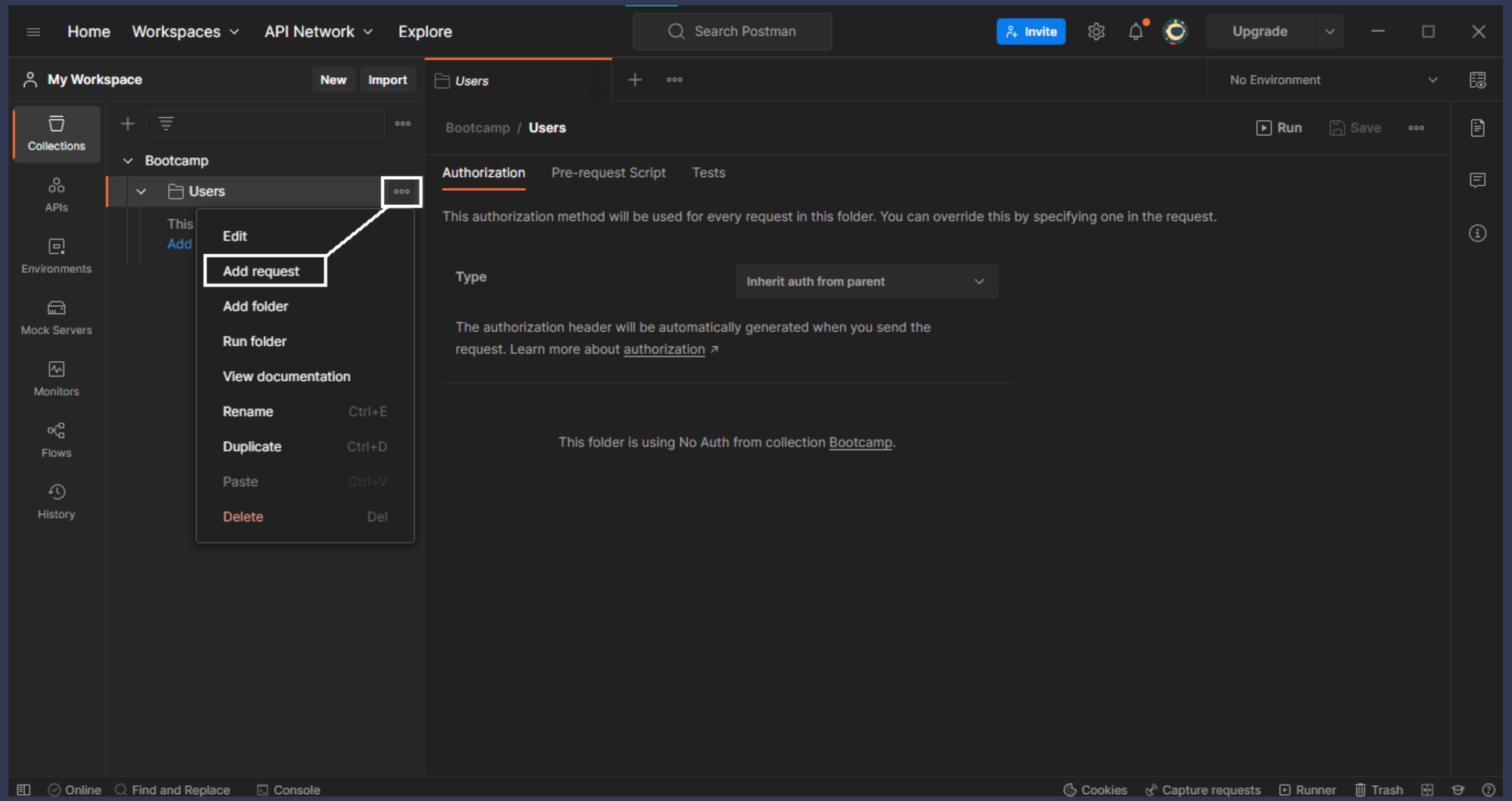
Untuk menambahkan folder pilih tombol icon triple dot "...", setelah itu pilih "Add folder"

Mengubah Nama Folder



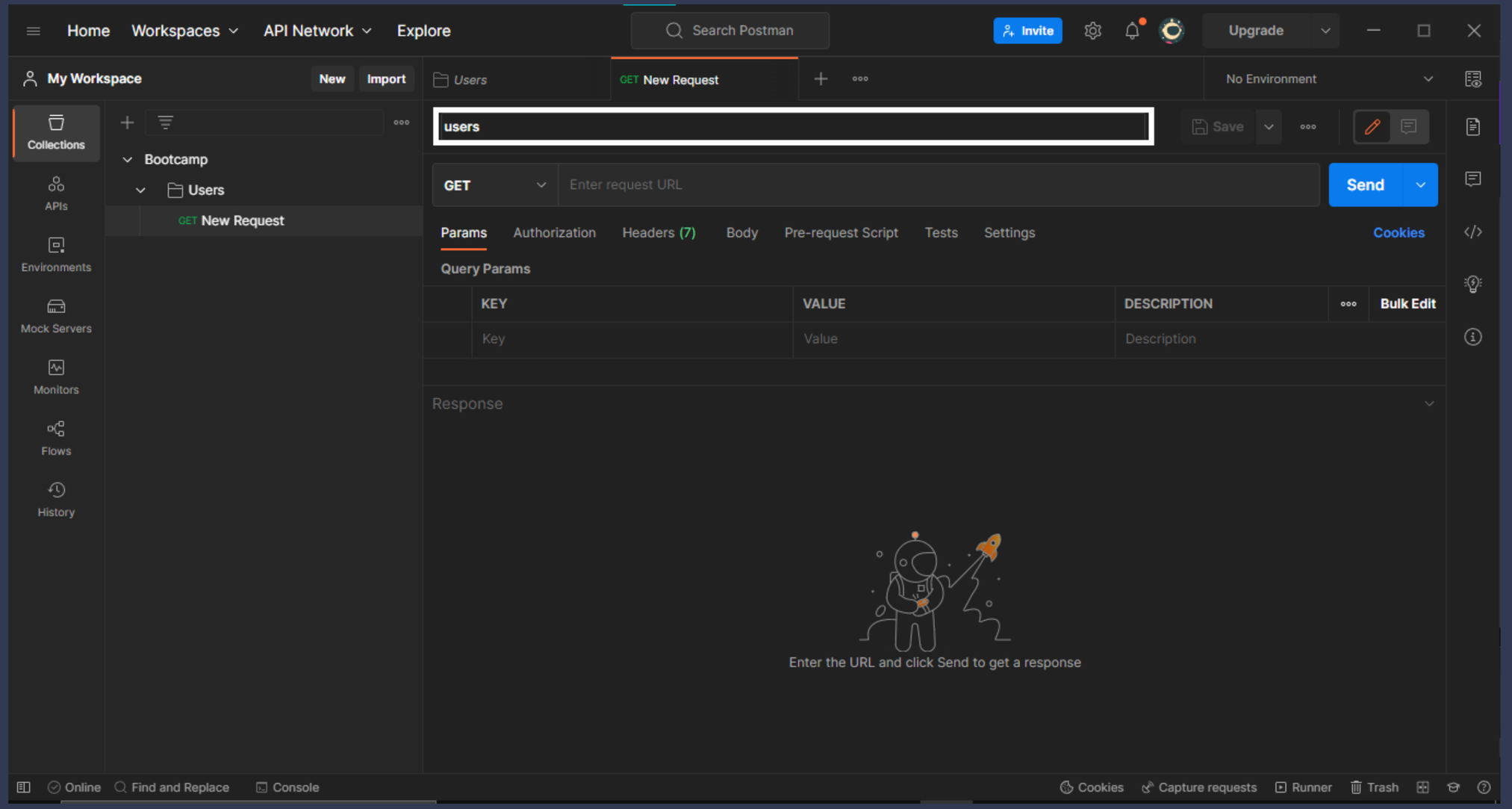
Rename folder sesuai dengan nama class yang telah dibuat yaitu "Users".

Menambahkan Request



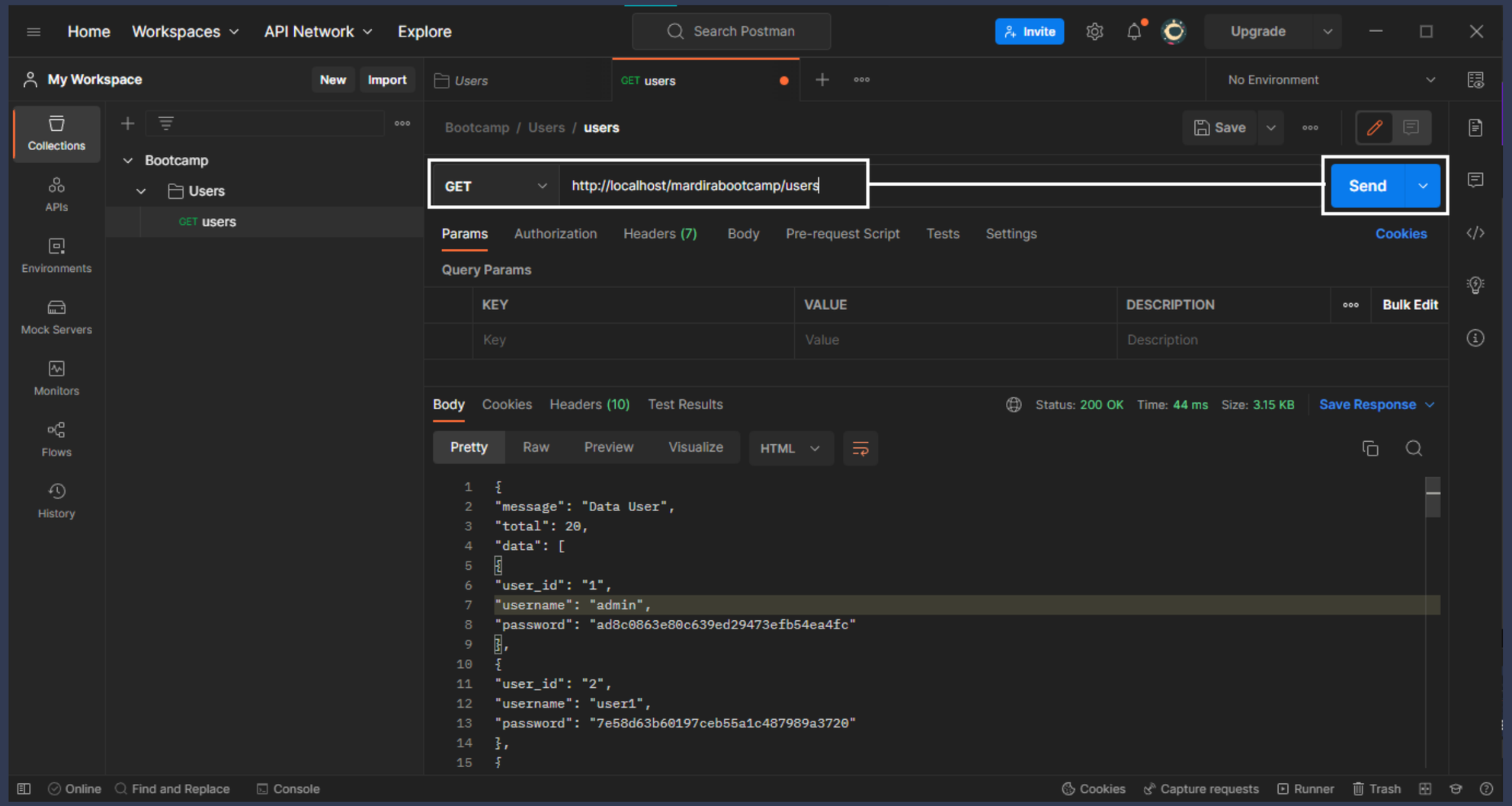
Langkah selanjutnya yaitu menambahkan request atau disebut dengan endpoint pilih tombol icon triple dot "..." kemudian pilih "Add Request".

Mengubah Nama Request



Untuk rename request sesuaikan dengan nama method yang telah dibuat

Menambahkan Endpoint URL



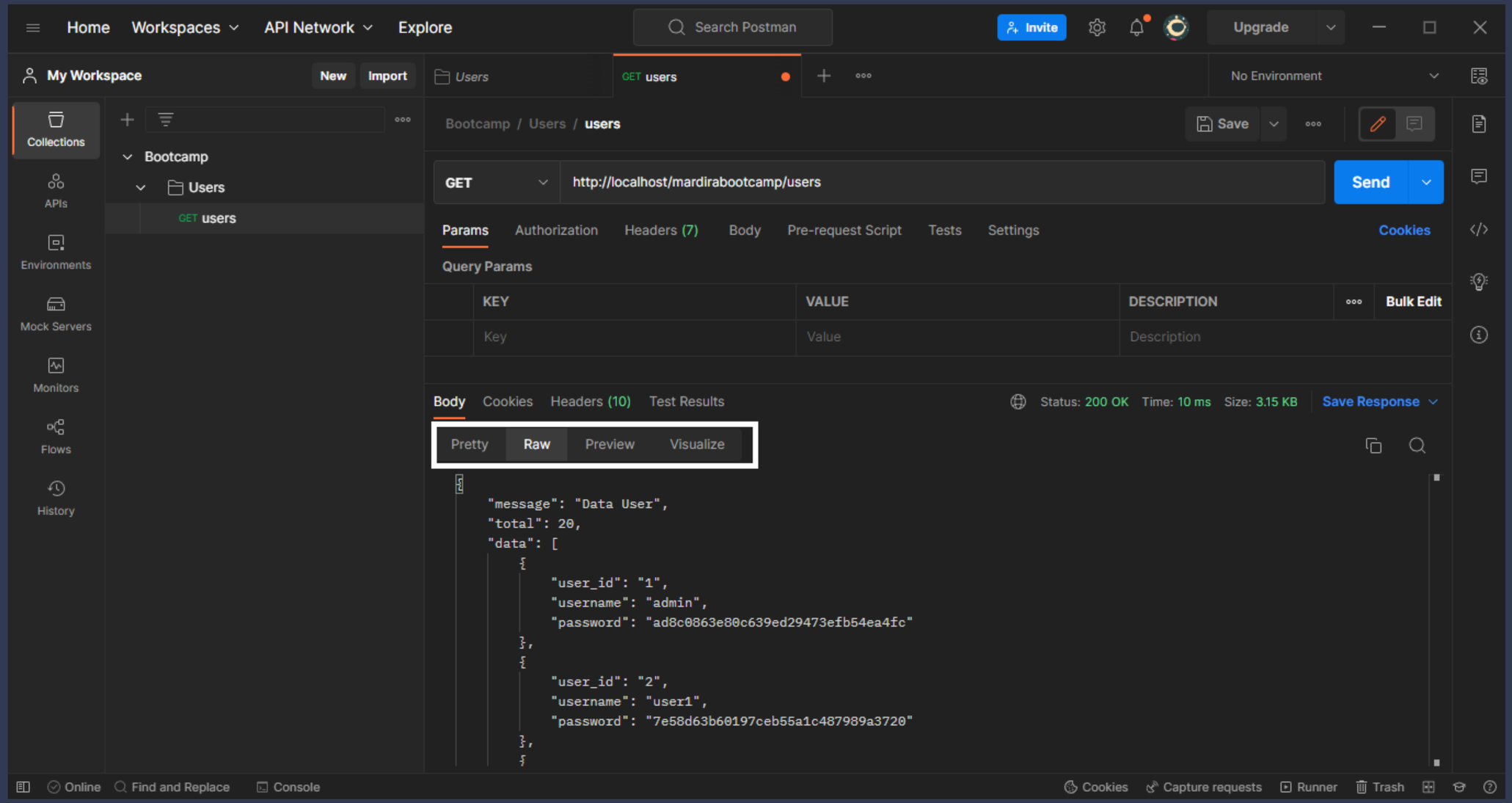
Kita tidak perlu lagi menambahkan ekstensi .php untuk URL nya karena sudah diatur oleh .htaccess itu sendiri, untuk menambahkan Endpoint URL cukup menambahkan prefix nya saja "/users", setelah itu pilih tombol send untuk melakukan request.

URL Pattern / Digital Object Indentifier

http://localhost/mardirabootcamp/users/delete/1

- Absolute URL
  - Base URL
  - Relative URL
    - Prefix
    - Suffix

Response Endpoint



Untuk melihat response pada endpoint yang telah direquest pilih satu tab yaitu pretty, raw, preview.

Menerapkan HTTP Verbs

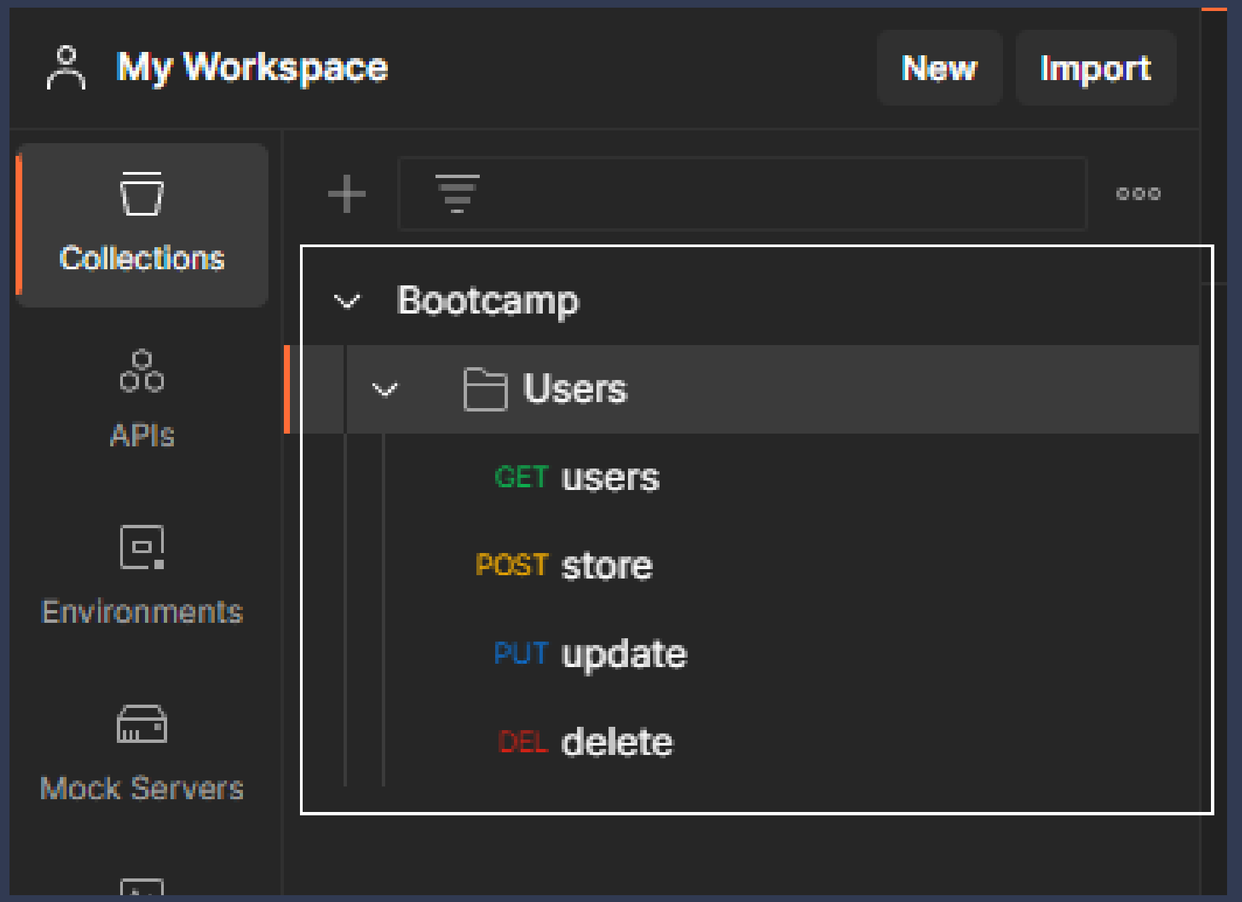
Route	Request Method	Body Request / Payload	Description
/users	GET	-	Menampilkan Semua Data Users
/users/:id	GET	-	Menampilkan satu data users sesuai id
/users	POST	• username • password	Menyimpan data User
/users/:id	PUT	• id • username • password	Mengubah data user sesuai dengan id
/users/:id	Delete	-	Menghapus data user sesuai dengan id

Todo

- URL Design 
- HTTP Verbs 
- HTTP Response Code 
- Format Response 

Setelah membuat URL Design, langkah selanjutnya ialah menerapkan HTTP verbs tujuannya untuk menggunakan semua method yang telah didefinisikan.

Menambahkan Endpoint Lainnya



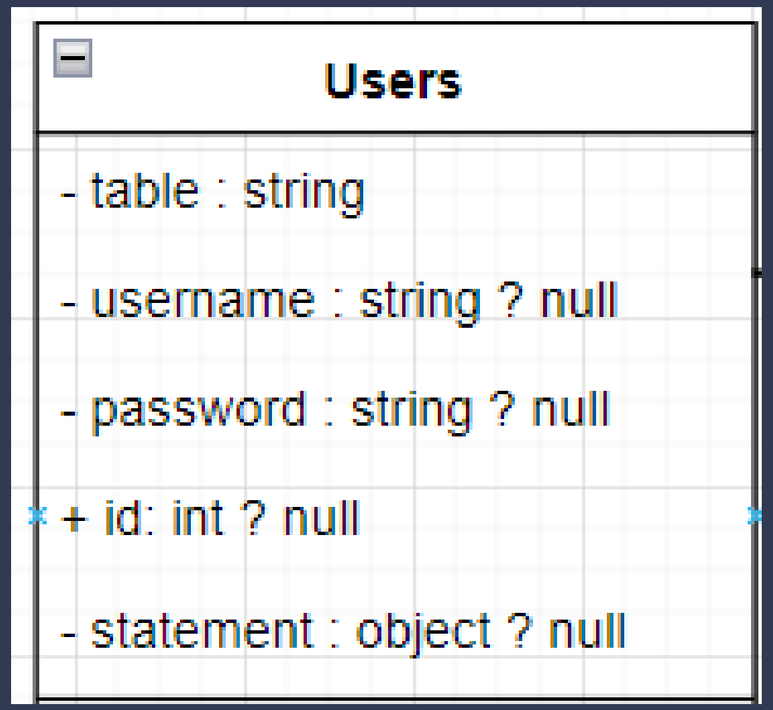
Tambahkan endpoint lainnya dengan HTTP verbs sesuai dengan di atas.

Control Statement Switch Untuk HTTP Verbs

```php
$users = new Users();

switch ($_SERVER['REQUEST_METHOD']) {
    case 'GET':
        $users->users();
        break;
    case 'POST':
        $users->store();
        break;
    case 'PUT':
        $users->update($users->id);
        break;
    case 'DELETE':
        $users->delete($users->id);
        break;
    default:
        $response = [
            'message' => 'Method Not Allowed',
        ];
        http_response_code(405);
        echo json_encode($response);
        break;
}
```

Setelah menambahkan endpoint di Postman, agar bisa menggunakan semua method untuk setiap requestnya dengan cara menggunakan control statement switch.

Menambahkan Property Class Users

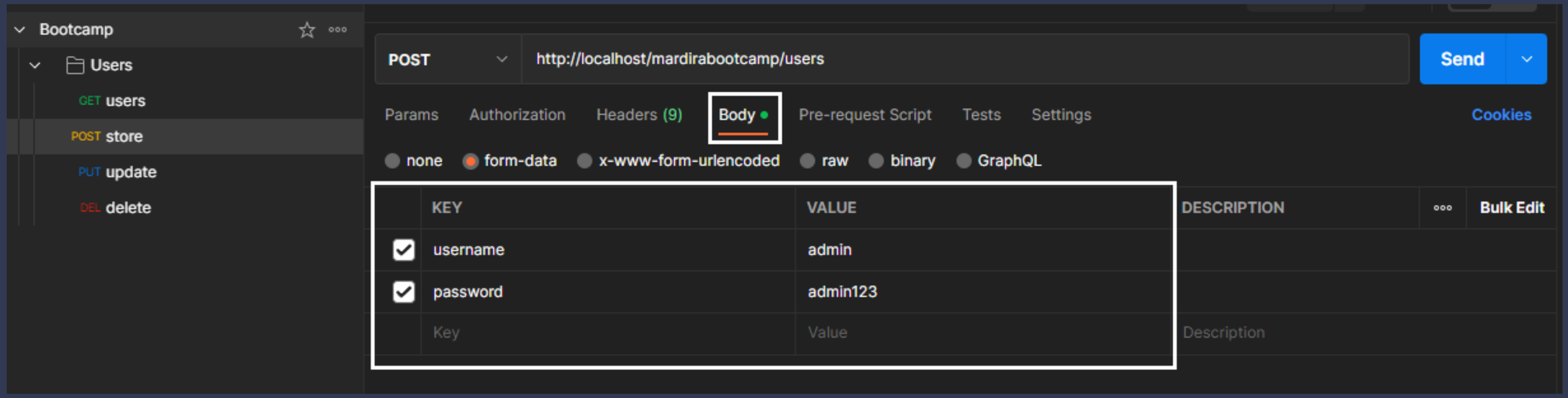



```php
class Users
{
    private string $table = 'users';
    private ?string $username;
    private ?string $password;
    public ?int $id;
    private ?object $statement;

    public function __construct()
    {
        $this->statement = new Database();
        $this->statement = $this->statement->connection;
        $this->username = $_POST['username'] ?? null;
        $this->password = $_POST['password'] ?? null;
        $this->id = $_GET['id'] ?? null;
    }
```

```php
case 'DELETE':
    $users->delete($users->id);
    break;
```

Kemudian tambahkan public property untuk id, supaya bisa digunakan diluar class untuk method delete dan update

Store Data Menggunakan Postman



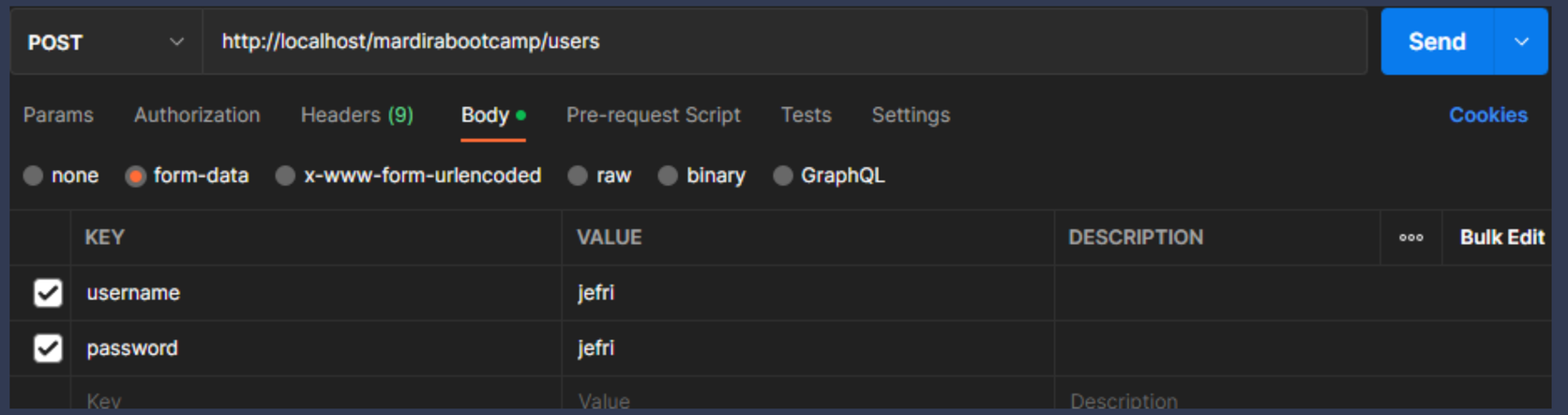
Untuk melakukan store data pada Postman, pilih tab body lalu pilih form-data kemudian masukkan key dan value sesuaikan dengan field/column yang ada di table yang akan ditambahkan.

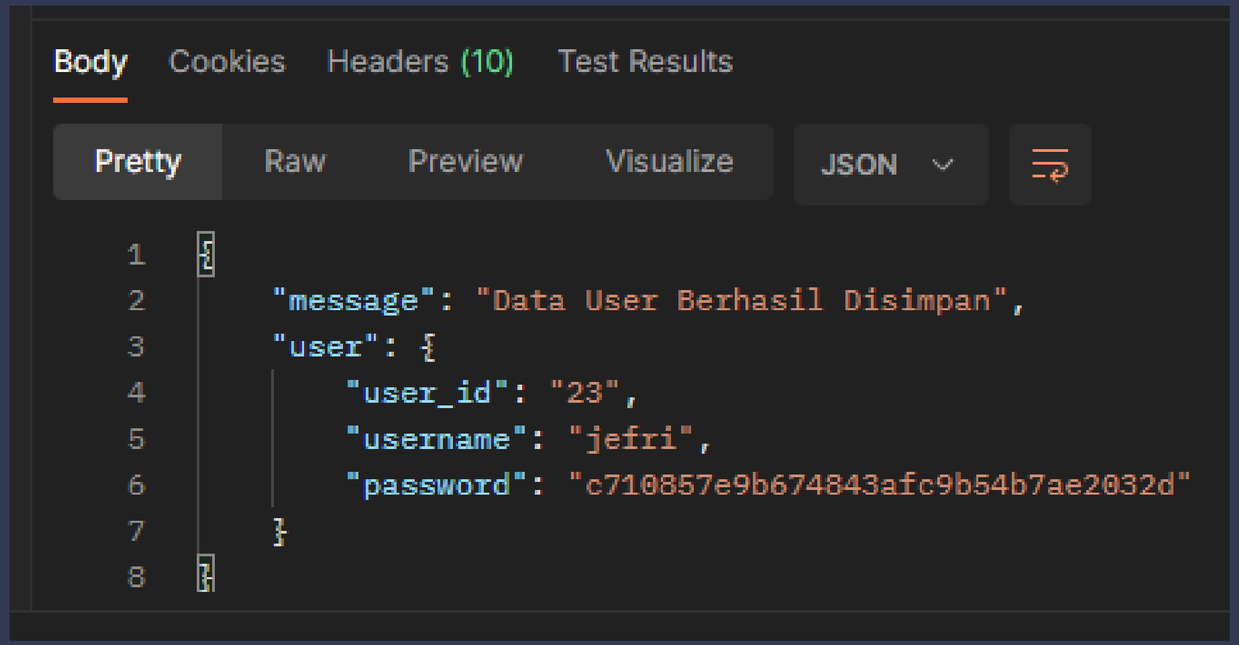
Proses Menyimpan Data Dari Postman

```php
public function store(): void
{
    $query = "SELECT * FROM {$this->table} WHERE username = :username";
    $statement = $this->statement->prepare($query);
    $statement->bindParam(':username', $this->username);
    $statement->execute();
    if ($statement->rowCount() > 0) {
        $response = [
            'message' => 'Data User Sudah Ada!',
        ];
    } else {
        $query = "INSERT INTO {$this->table} (username, password) VALUES (:username, :password)";
        $statement = $this->statement->prepare($query);
        $password = md5($this->password);
        $statement->bindParam(':username', $this->username);
        $statement->bindParam(':password', $password);
        $statement->execute();
        $getUsers = "SELECT * FROM users WHERE username = :username";
        $statement = $this->statement->prepare($getUsers);
        $statement->bindParam(':username', $this->username);
        $statement->execute();
        $user = $statement->fetch(PDO::FETCH_OBJ);
        $response = [
            'message' => 'Data User Berhasil Disimpan',
            'user' => $user,
        ];
    }
    echo json_encode($response, JSON_PRETTY_PRINT);
}
```

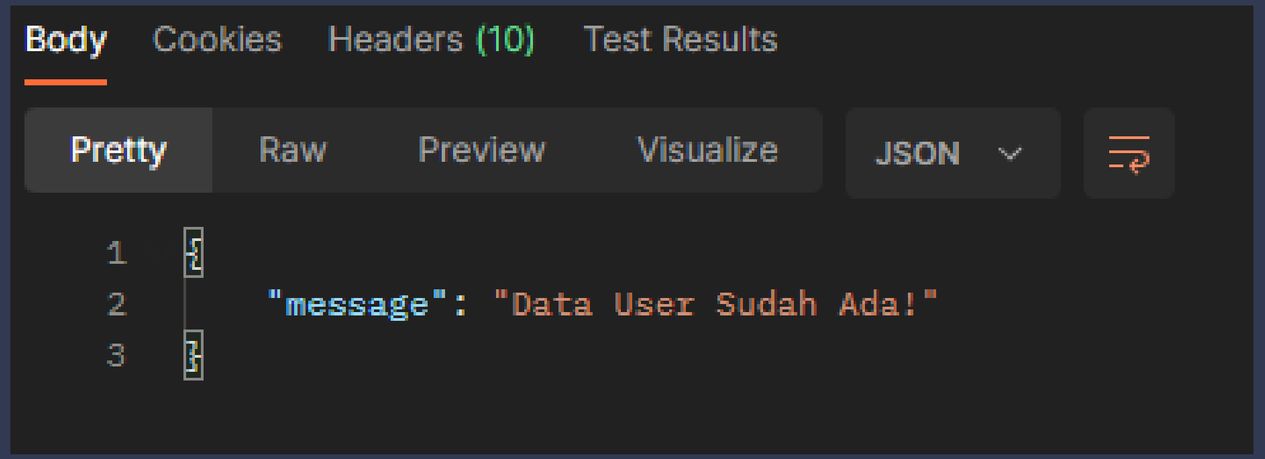
Sebelum melakukan penyimpan data akan di cek terlebih dahulu untuk memastikan data tersebut bersifat duplikat atau tidak dengan menggunakan query rowCount untuk menghitung data. Lalu menggunakan control statement untuk menjalankan salah satu aksi / response yang akan ditampilkan.

Response Store Data



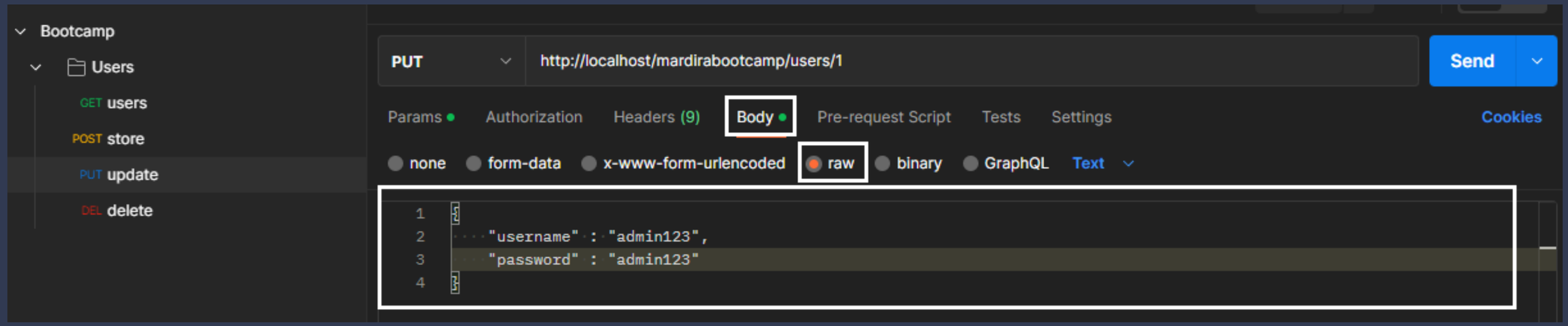


Data berhasil disimpan



Duplikat data

Update Data Menggunakan Postman



Saat melakukan update data (PUT) dari Postman berbeda dengan melakukan store (POST), karena PUT tidak bisa menggunakan form-data seperti POST. Jadi untuk melakukan perubahan data piih tab body lalu pilih raw, kemudian masukkan data berupa format JSON, dan cantumkan id yang ingin diubah datanya. Posisi id berada di suffix setelah prefix pada endpoint **"users/1"**

Proses Mengubah Data Dari Postman

```php
public function update(int $id): void
{

    $data = file_get_contents("php://input");
    $_PUT = json_decode($data, true);
    $this->username = $_PUT['username'] ?? null;
    $this->password = $_PUT['password'] ?? null;

    $query = "SELECT * FROM {$this->table} WHERE user_id = :user_id";
    $statement = $this->statement->prepare($query);
    $statement->bindParam(':user_id', $id);
    $statement->execute();

    if ($statement->rowCount() == 0) {
        $response = [
            'message' => 'Data User Tidak Ditemukan!',
        ];
        echo json_encode($response, JSON_PRETTY_PRINT);
        exit;
    }

    $query = "UPDATE {$this->table} SET username = :username, password = :password WHERE user_id = :user_id";
    $statement = $this->statement->prepare($query);
    $password = md5($this->password);
    $statement->bindParam(':username', $this->username);
    $statement->bindParam(':password', $password);
    $statement->bindParam(':user_id', $id);
    $statement->execute();
    $response = [
        'message' => 'Data User Berhasil Diubah',
    ];
    echo json_encode($response, JSON_PRETTY_PRINT);
}
```

Proses untuk mengubah data di method update, untuk menangkap payload/request data yang sifatnya raw (mentah), menggunakan fungsi file_get_contents, sebelum melakukan update data untuk memastikan data tersebut ada atau tidak dengan cara melakukan cek data dengan id, jika tidak ada maka akan memberi response tidak ada dan json_decode yang berfungsi untuk mengubah dari JSON menjadi array php.

Json Encode & Json Decode

Array PHP

```php
[
    'username' => 'admin123',
    'password' => 'admin123',
];
```

json_encode

konversi

JSON

```json
{
  "username": "admin123",
  "password": "admin123"
}
```

JSON

```json
{
  "username": "admin123",
  "password": "admin123"
}
```

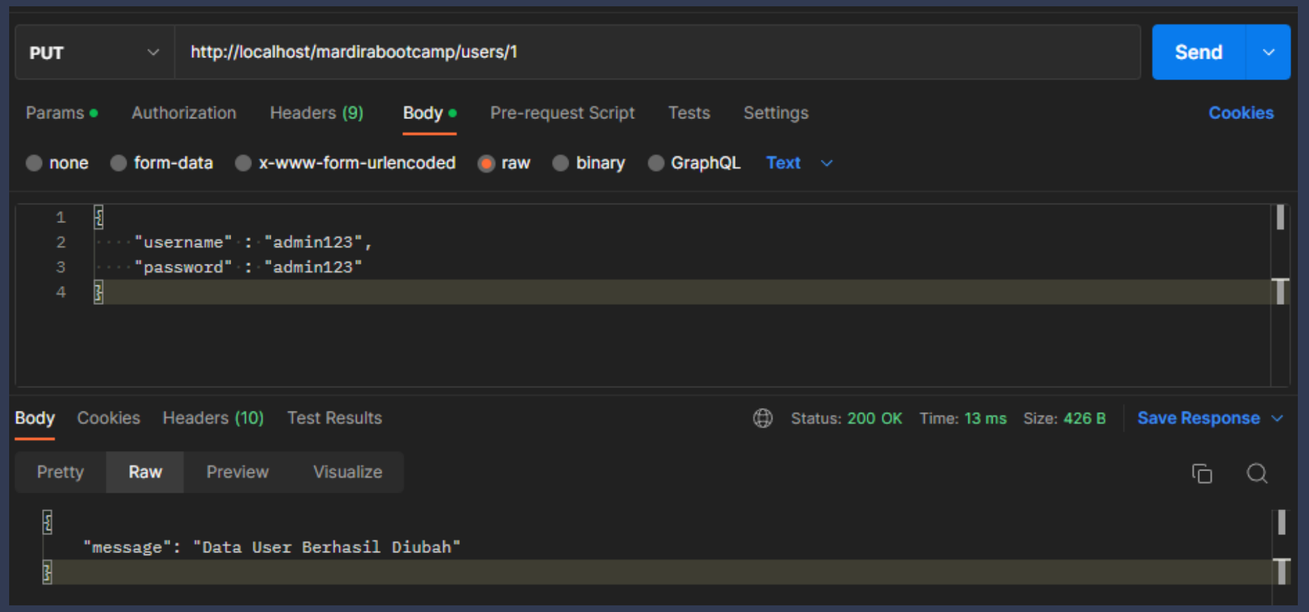
json_decode

konversi

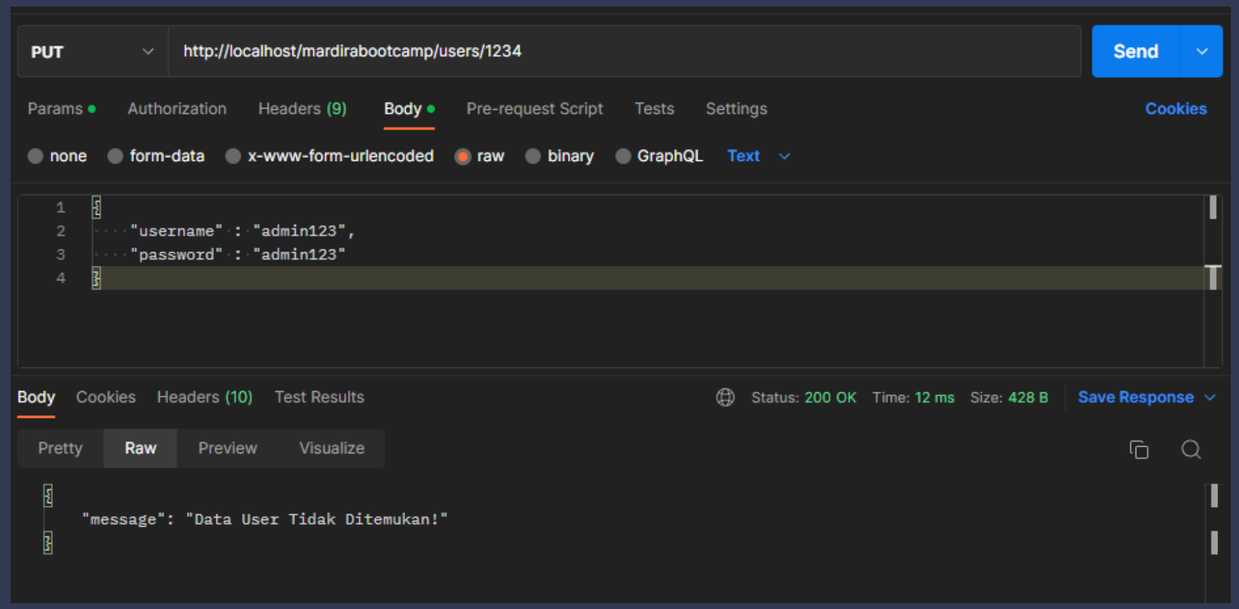
Array PHP

```php
[
    'username' => 'admin123',
    'password' => 'admin123',
];
```

Response Update Data

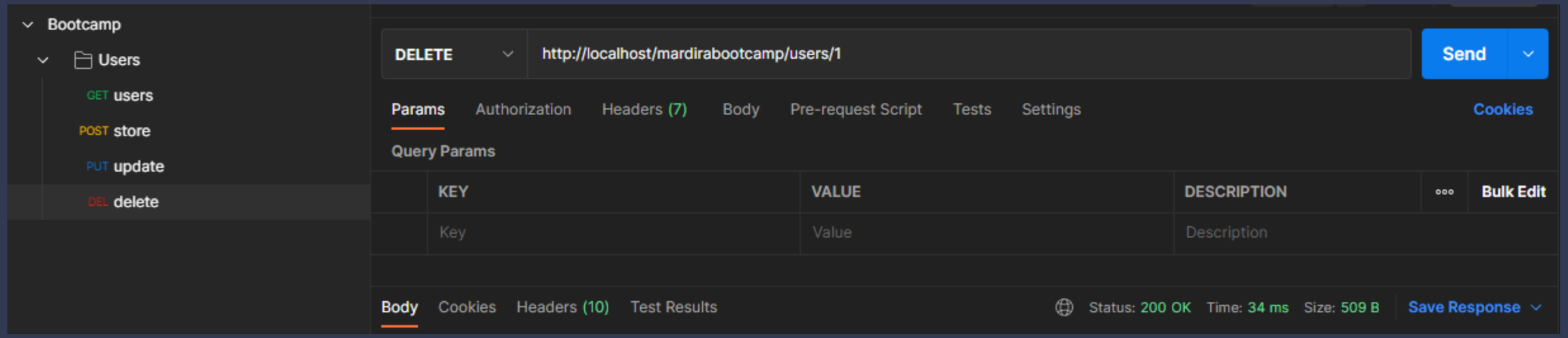


Data berhasil diubah



Data tidak ditemukan

Delete Data Menggunakan Postman



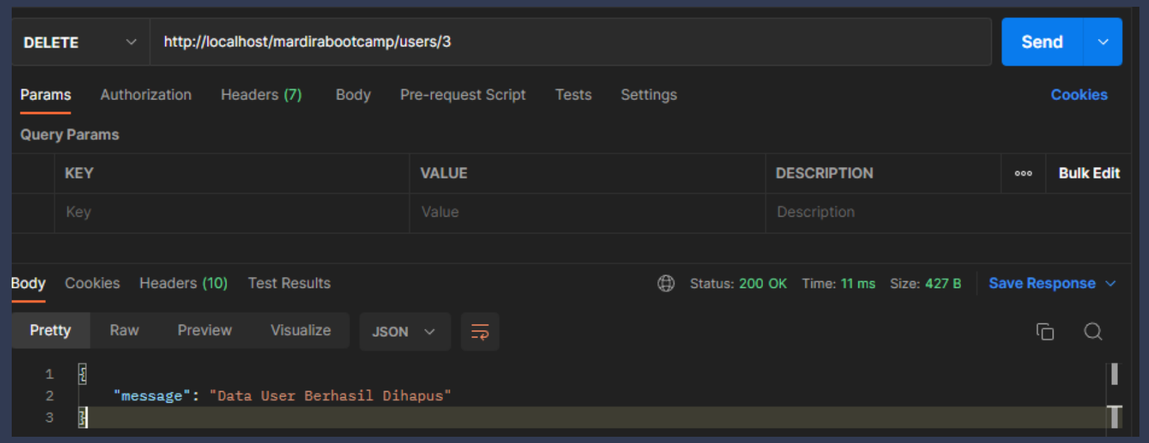
Delete data menggunakan Postman cukup menambahkan suffix id pada endpoint delete.

Proses Menghapus Data Dari Postman

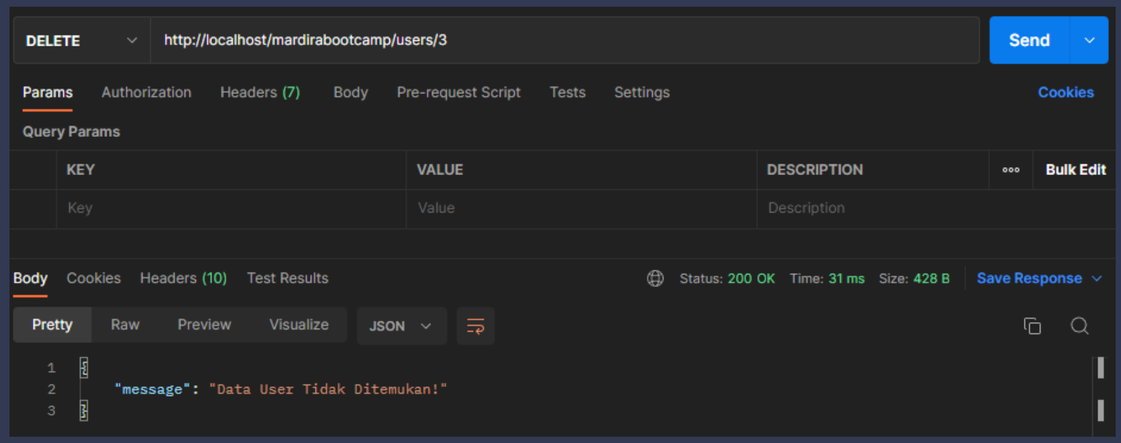
```php
public function delete(int $id): void
{
    $query = "SELECT * FROM {$this->table} WHERE user_id = :user_id";
    $statement = $this->statement->prepare($query);
    $statement->bindParam(':user_id', $id);
    $statement->execute();

    if ($statement->rowCount() == 0) {
        $response = [
            'message' => 'Data User Tidak Ditemukan!',
        ];
        echo json_encode($response, JSON_PRETTY_PRINT);
        exit;
    }

    $query = "DELETE FROM {$this->table} WHERE user_id = :user_id";
    $statement = $this->statement->prepare($query);
    $statement->bindParam(':user_id', $id);
    $statement->execute();
    $response = [
        'message' => 'Data User Berhasil Dihapus',
    ];
    echo json_encode($response, JSON_PRETTY_PRINT);
}
```

Untuk method delete tambahka select data where id untuk menentukan data yang dihapus terdaftar ditable users.

Response Delete Data

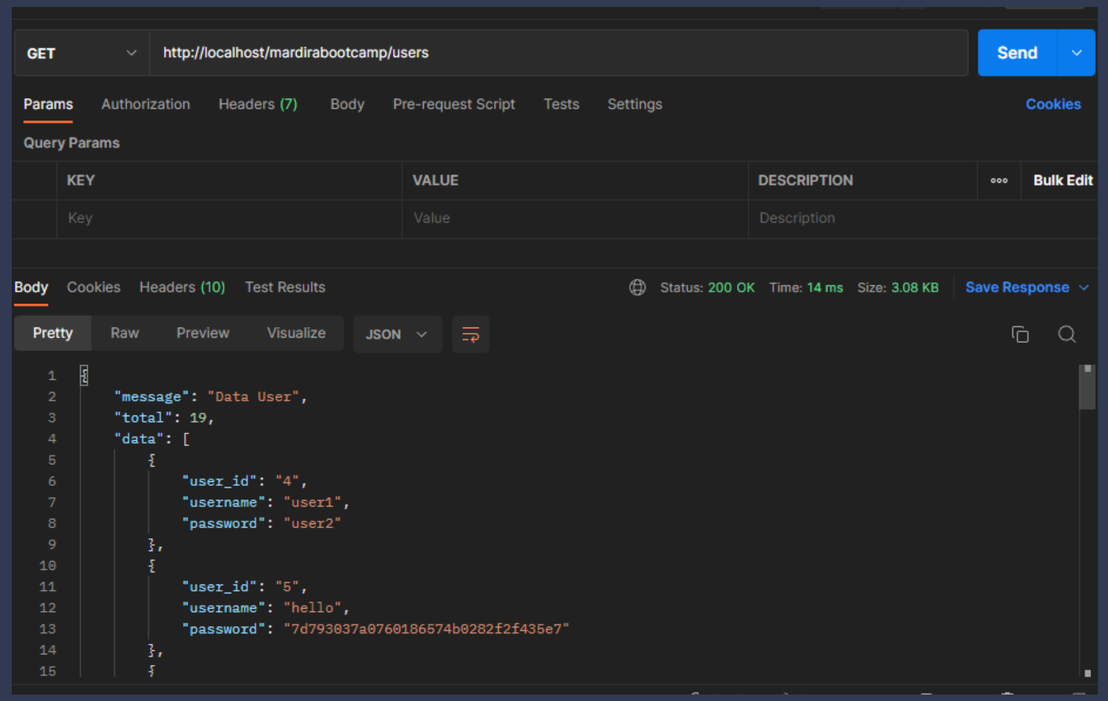


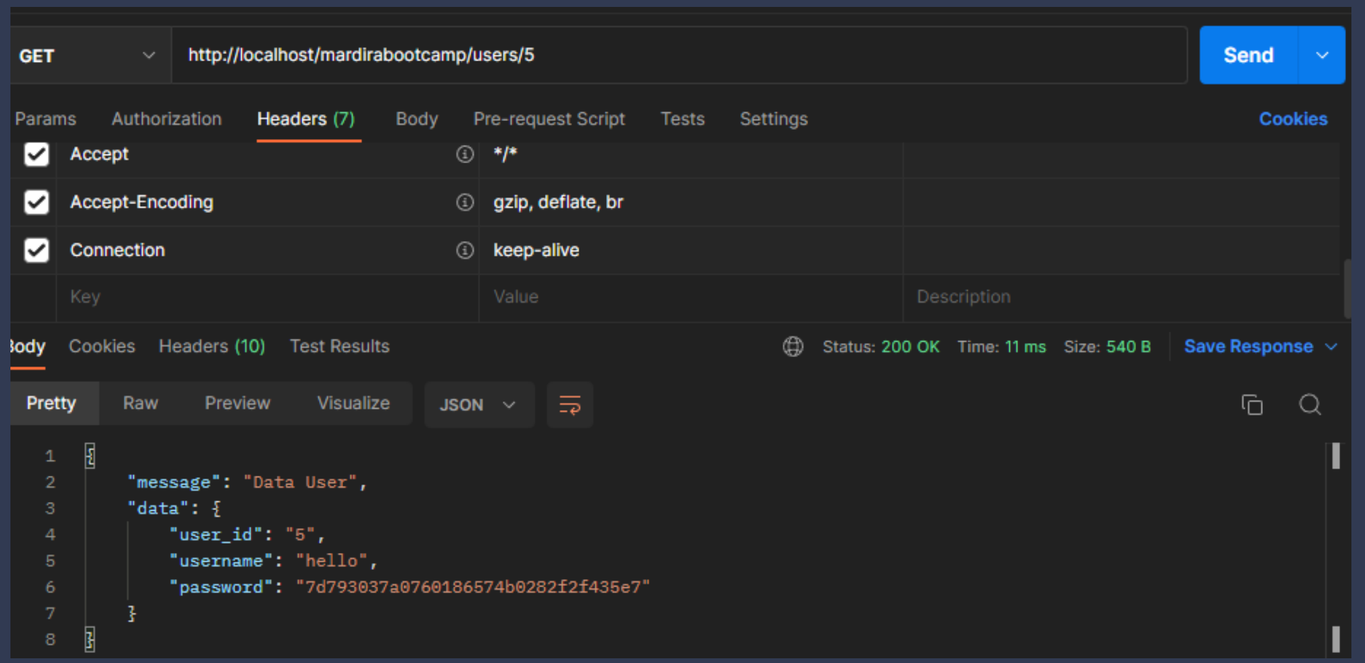
Data berhasil dihapus



Data tidak ditemukan

Menampilkan Data Berdasarkan ID





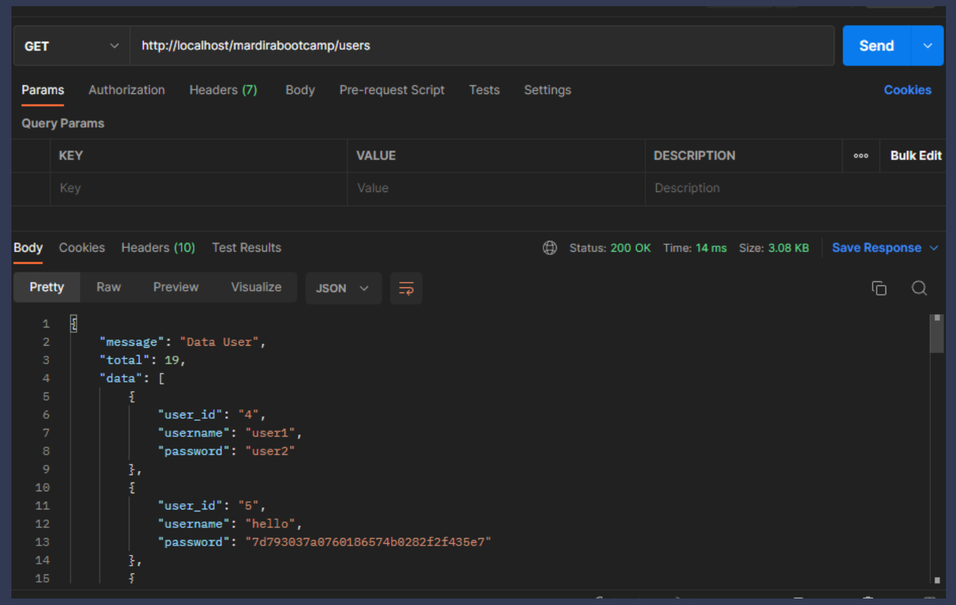
Bagaimana caranya menampilkan semua data dan menampilkan data berdasarkan id yang dipilih, dalam satu endpoint?

Method Khusus Untuk Get Data

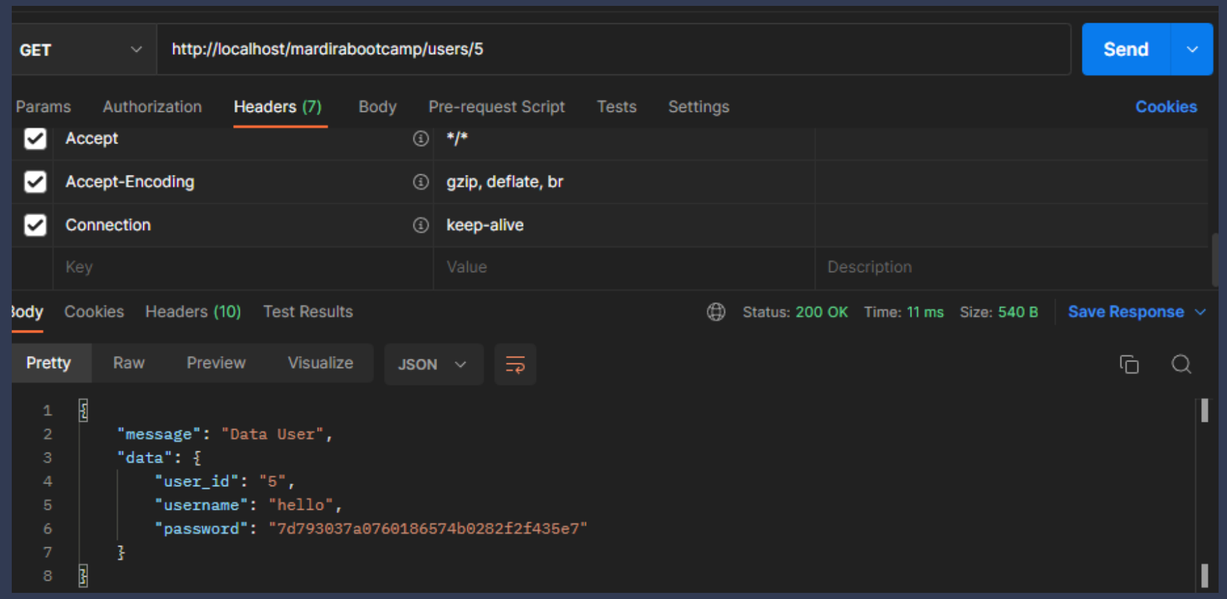
```php
public function users(): void
{
    if ($this->id) {
        $user = $this->getUser($this->id);
        $response = [
            'message' => 'Data User',
            'data' => $user
        ];
        echo json_encode($response, JSON_PRETTY_PRINT);
    } else {
        $users = $this->getUsers();
        $total = count($users);
        $response = [
            'message' => 'Data User',
            'total' => $total,
            'data' => $users
        ];
        echo json_encode($response, JSON_PRETTY_PRINT);
    }
}

private function getUsers(): array
{
    $query = "SELECT * FROM {$this->table}";
    $statement = $this->statement->prepare($query);
    $statement->execute();
    $users = $statement->fetchAll(PDO::FETCH_OBJ);
    return $users;
}

private function getUser(int $id): object
{
    $query = "SELECT * FROM {$this->table} WHERE user_id = :user_id";
    $statement = $this->statement->prepare($query);
    $statement->bindParam(':user_id', $id);
    $statement->execute();
    if ($statement->rowCount() == 0) {
        $response = [
            'message' => 'Data User Tidak Ditemukan!',
        ];
        http_response_code(404);
        echo json_encode($response, JSON_PRETTY_PRINT);
        exit;
    }
    $user = $statement->fetch(PDO::FETCH_OBJ);
    return $user;
}
```

Agar bisa menampilkan (hybrid data) semua data atau data berdasarkan id saja, dengan cara menambahkan private method, kemudian gunakan control statement jika id yang di input ada ataupun tidak.

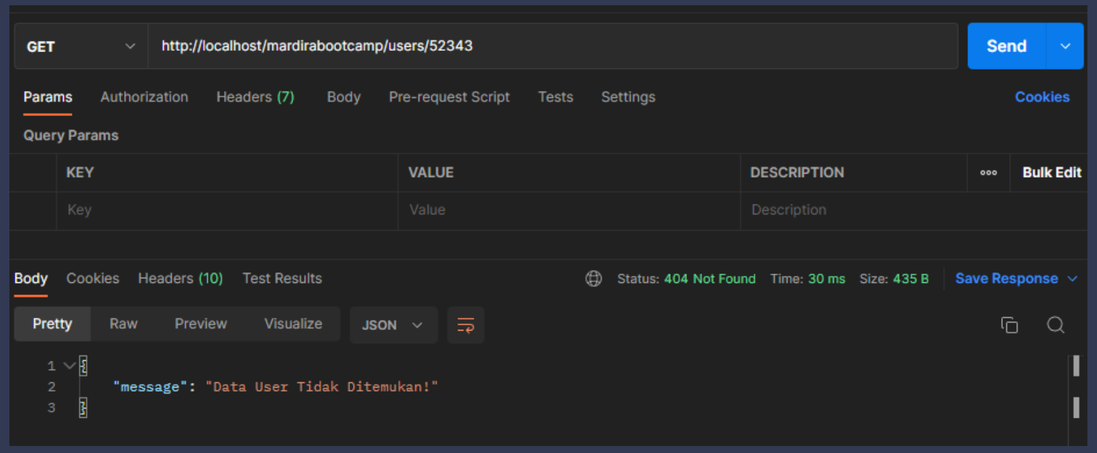
Response Get Data



Menampilkan semua data



Menampilkan data berdasarkan id



Data tidak ditemukan

Menerapkan HTTP Response Code

200 (OK)

201 (Created)

400 (Bad Request)

401 (Unauthorized)

500 (ISE)

403 (Forbidden)

404 (Not Found)

405 (Method Not Allowed)

409 (Conflict)

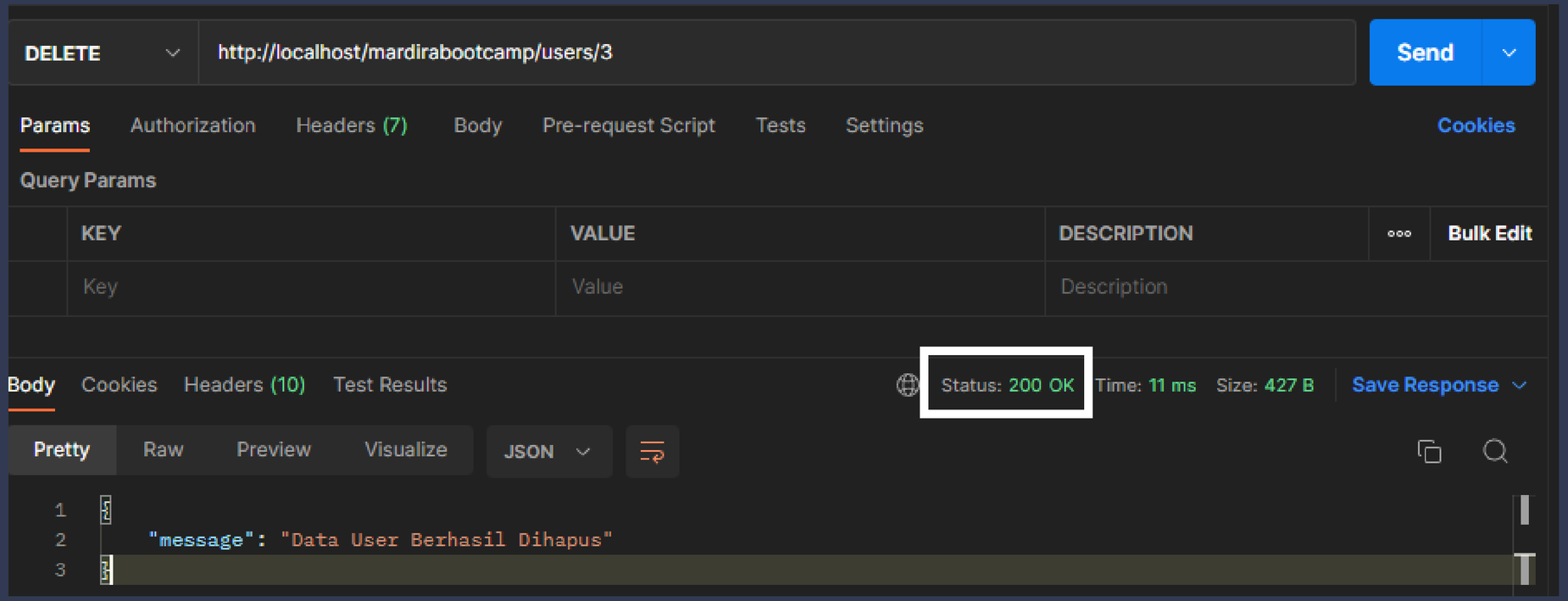
503 (Service Unavailable)

Todo

- URL Design
- HTTP Verbs
- HTTP Response Code
- Format Response

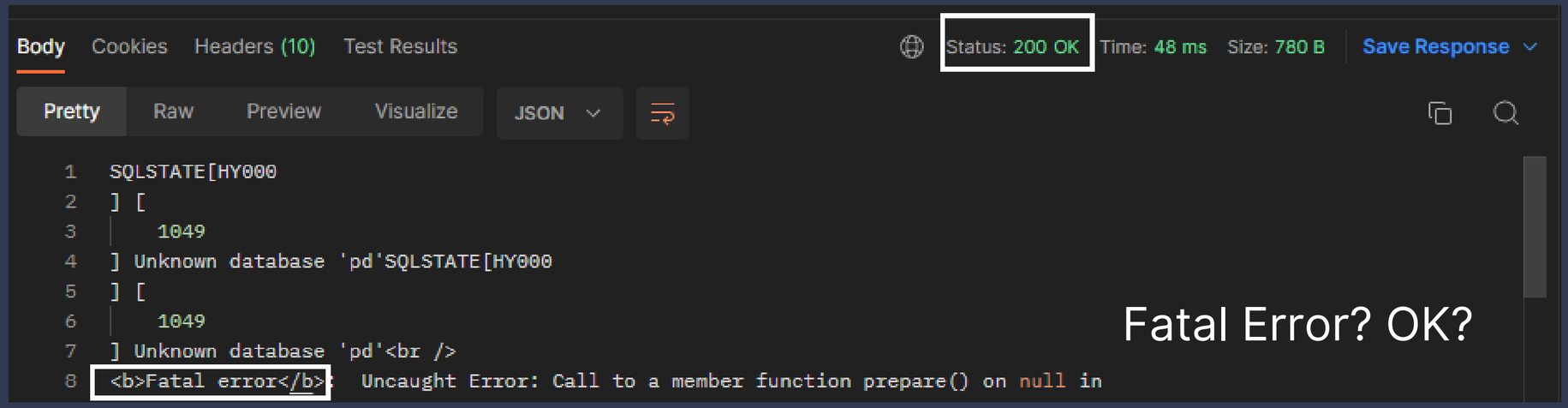
Setelah menentukan HTTP verbs, langkah selanjutnya adalah menentukan HTTP Response code setiap response dari endpoint yang sudah ditentukan.

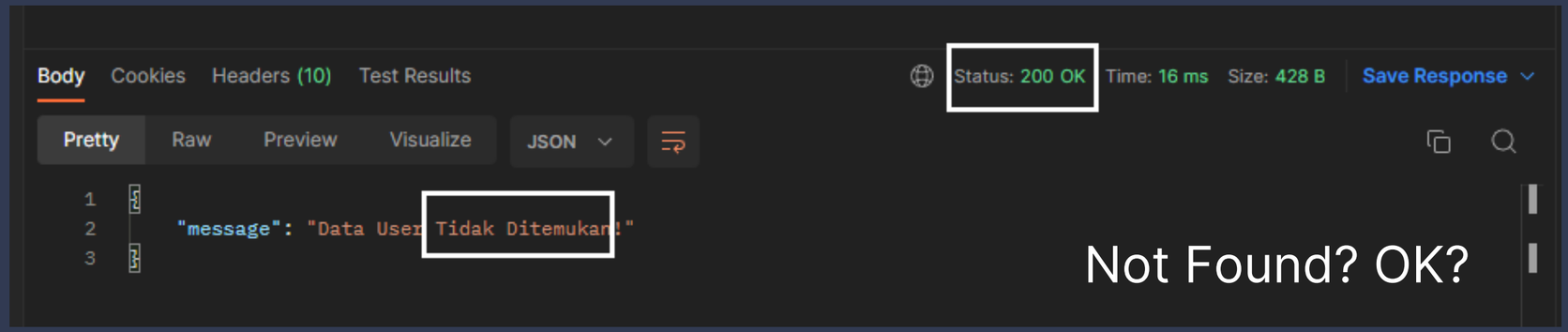
Melihat HTTP Response Code



Setelah melakukan send request akan muncul status code / response code seperti di atas.

Jika Tidak Menerapkan HTTP Response Code





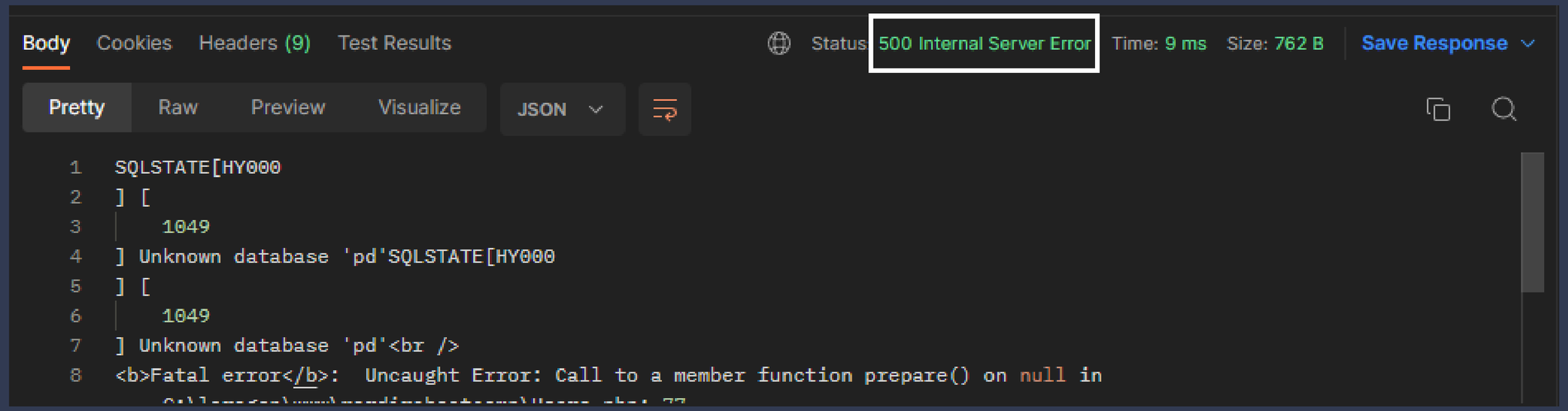
Secara default jika kita tidak menggunakan status code, status code yang akan muncul akan slalu 200 (OK), dan kalau tidak menerapkan status code justru akan menyulitkan client / Front-End yang ingin berkolaborasi dengan kalian sebagai Back-End Developer

Implementasi HTTP Response Code

```php
<?php

class Database
{
    private const DB_HOST  = 'localhost';
    private const DB_USER = 'root';
    private const DB_PASS = '';
    private const DB_NAME = 'pdo';
    private const DSN = 'mysql:host=' . self::DB_HOST . ';dbname=' . self::DB_NAME . '';
    public ?object $connection = null;

    public function __construct()
    {
        try {
            $this->connection = new PDO(
                self::DSN,
                self::DB_USER,
                self::DB_PASS
            );
            $this->connection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            http_response_code(500);
            echo $e->getMessage();
        }
        return $this->connection;
    }
}

$database = new Database();
```

Untuk menetapkan status code pada setiap response, dengan cara menambahkan http_response_code(kode) dengan parameter kodenya, untuk contoh jika ada kesalahan di server dengan menggunakan status code 500 (Internal Server Error).

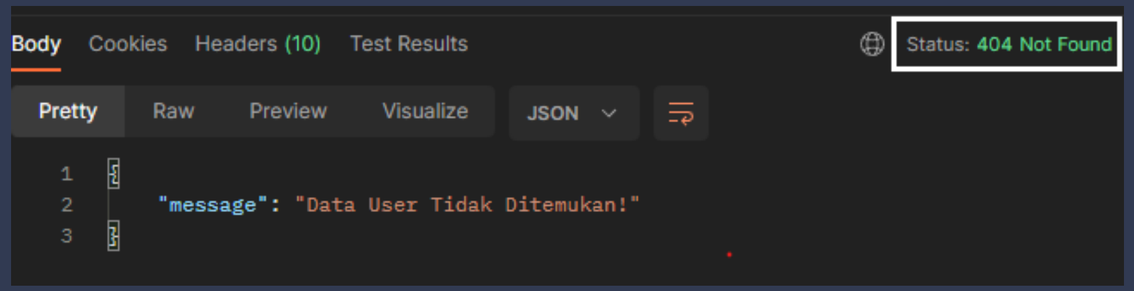
Hasil HTTP Response Code



Setelah menetapkan HTTP Response code maka hasilnya akan muncul setelah melakukan send request menggunakan Postman.

Implementasi HTTP Response Code Lainnya

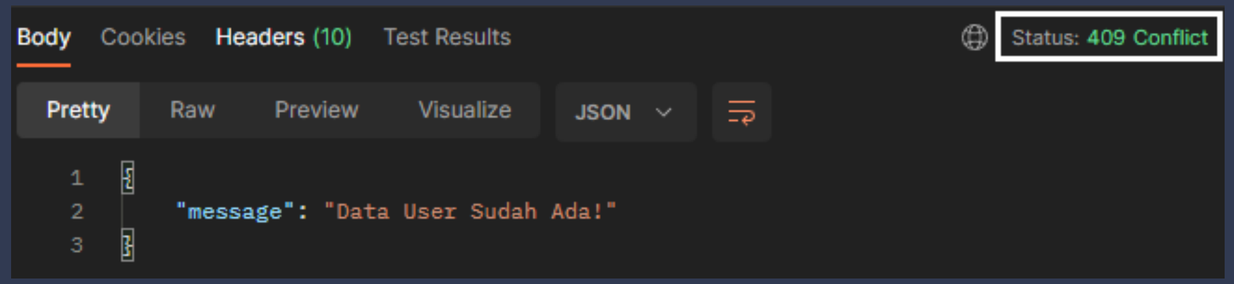
```php
private function getUser(int $id): object
{
    $query = "SELECT * FROM {$this->table} WHERE user_id = :user_id";
    $statement = $this->statement->prepare($query);
    $statement->bindParam(':user_id', $id);
    $statement->execute();
    if ($statement->rowCount() == 0) {
        $response = [
            'message' => 'Data User Tidak Ditemukan!',
        ];
        http_response_code(404);
        echo json_encode($response, JSON_PRETTY_PRINT);
        exit;
    }
    $user = $statement->fetch(PDO::FETCH_OBJ);
    return $user;
}
```



Jika data yang dicari tidak ada maka status code yang digunakan adalah 404 (Not Found).

Implementasi HTTP Response Code Lainnya

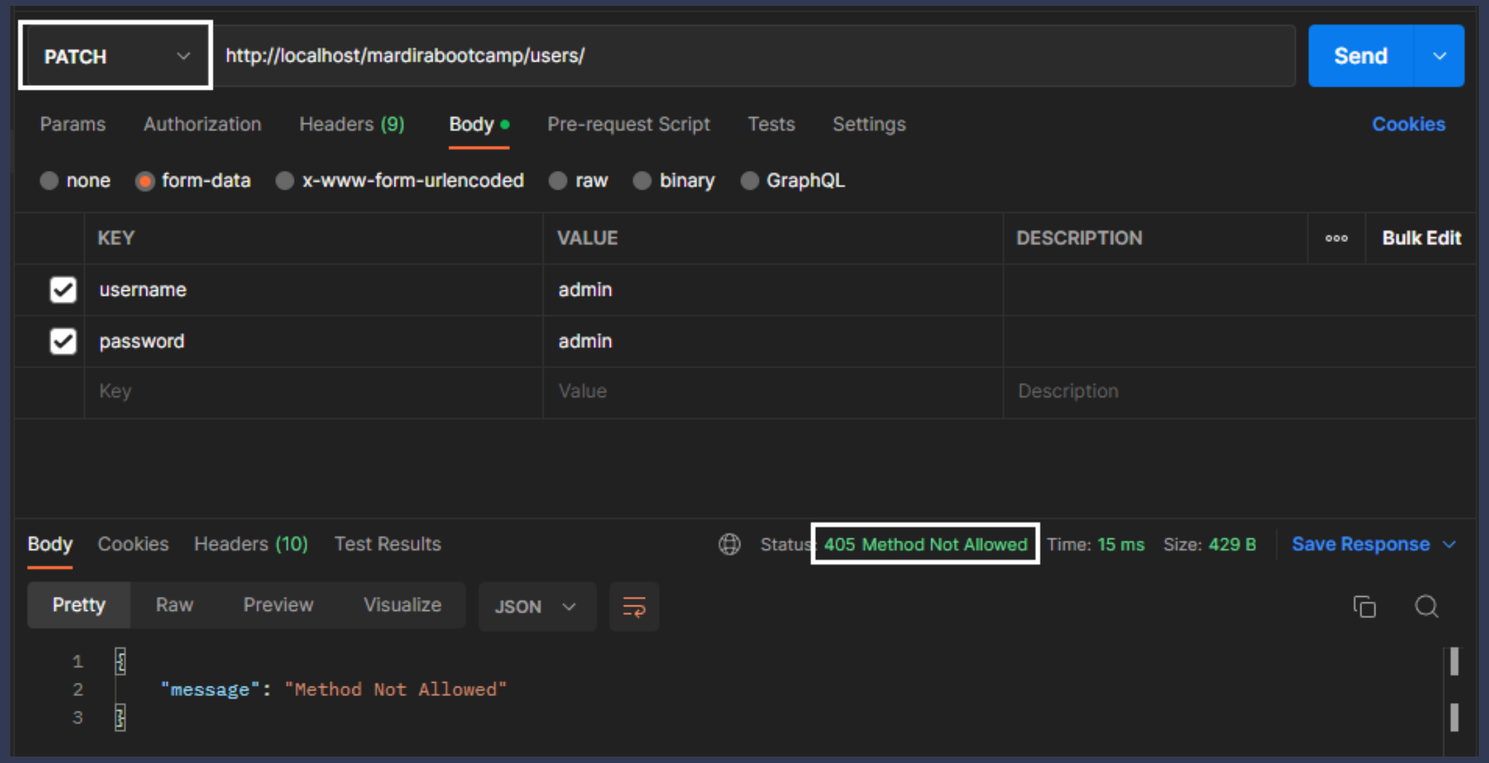
```php
public function store(): void
{
    $query = "SELECT * FROM {$this->table} WHERE username = :username";
    $statement = $this->statement->prepare($query);
    $statement->bindParam(':username', $this->username);
    $statement->execute();
    if ($statement->rowCount() > 0) {
        $response = [
            'message' => 'Data User Sudah Ada!',
        ];
        http_response_code(409);
    } else {
        $query = "INSERT INTO {$this->table} (username, password) VALUES (:username, :password)";
        $statement = $this->statement->prepare($query);
        $password = md5($this->password);
        $statement->bindParam(':username', $this->username);
        $statement->bindParam(':password', $password);
        $statement->execute();
        $getUsers = "SELECT * FROM users WHERE username = :username";
        $statement = $this->statement->prepare($getUsers);
        $statement->bindParam(':username', $this->username);
        $statement->execute();
        $user = $statement->fetch(PDO::FETCH_OBJ);
        $response = [
            'message' => 'Data User Berhasil Disimpan',
            'user' => $user,
        ];
    }
    echo json_encode($response, JSON_PRETTY_PRINT);
}
```



Untuk data yang bersifat duplikat gunakan 409 (Conflict)

Implementasi HTTP Response Code Lainnya

```php

switch ($_SERVER['REQUEST_METHOD']) {
    case 'GET':
        $users->users();
        break;
    case 'POST':
        $users->store();
        break;
    case 'PUT':
        $users->update($users->id);
        break;
    case 'DELETE':
        $users->delete($users->id);
        break;
    default:
        $response = [
            'message' => 'Method Not Allowed',
        ];
        http_response_code(405);
        echo json_encode($response);
        break;
}
```



405 (Method Not Allowed) untuk HTTP verbs yang tidak digunakan tidak diizinkan karena hanya bisa GET, PUT , POST, dan DELETE saja.

Implementasi HTTP Response Code Lainnya

```php
public function store(): void
{
    $query = "SELECT * FROM {$this->table} WHERE username = :username";
    $statement = $this->statement->prepare($query);
    $statement->bindParam(':username', $this->username);
    $statement->execute();
    if ($statement->rowCount() > 0) {
        $response = [
            'message' => 'Data User Sudah Ada!',
        ];
        http_response_code(409);
    } else {
        $query = "INSERT INTO {$this->table} (username, password) VALUES (:username, :password)";
        $statement = $this->statement->prepare($query);
        $password = md5($this->password);
        $statement->bindParam(':username', $this->username);
        $statement->bindParam(':password', $password);
        $statement->execute();
        $getUsers = "SELECT * FROM users WHERE username = :username";
        $statement = $this->statement->prepare($getUsers);
        $statement->bindParam(':username', $this->username);
        $statement->execute();
        $user = $statement->fetch(PDO::FETCH_OBJ);
        $response = [
            'message' => 'Data User Berhasil Disimpan',
            'user' => $user,
        ];
        http_response_code(201);
    }
    echo json_encode($response, JSON_PRETTY_PRINT);
}
```

Body Cookies Headers (10) Test Results — Status: 201 Created

Pretty Raw Preview Visualize JSON

```json
{
    "message": "Data User Berhasil Disimpan",
    "user": {
        "user_id": "24",
        "username": "stafff",
        "password": "1253208465b1efa876f982d8a9e73eef"
    }
}
```

Untuk berhasil melakukan input data gunakan status code 201 (Created)

Menerapkan Format Response


```
{
    message: "Data User",
  ▾ data: {
        user_id: "5",
        username: "hello",
        password: "7d793037a0760186574b0282f2f435e7"
    }
}
```


Todo

- URL Design 
- HTTP Verbs 
- HTTP Response Code 
- Format Response 

Komponen yang terakhir adalah format response dimana JSON adalah format yang sering digunakan untuk berinteraksi antara FE & BE

Format Reponse

konversi

```php
[
    'username' => 'admin123',
    'password' => 'admin123',
];
```

```json
{
  "username": "admin123",
  "password": "admin123"
}
```

Todo

- URL Design
- HTTP Verbs
- HTTP Response Code
- Format Response

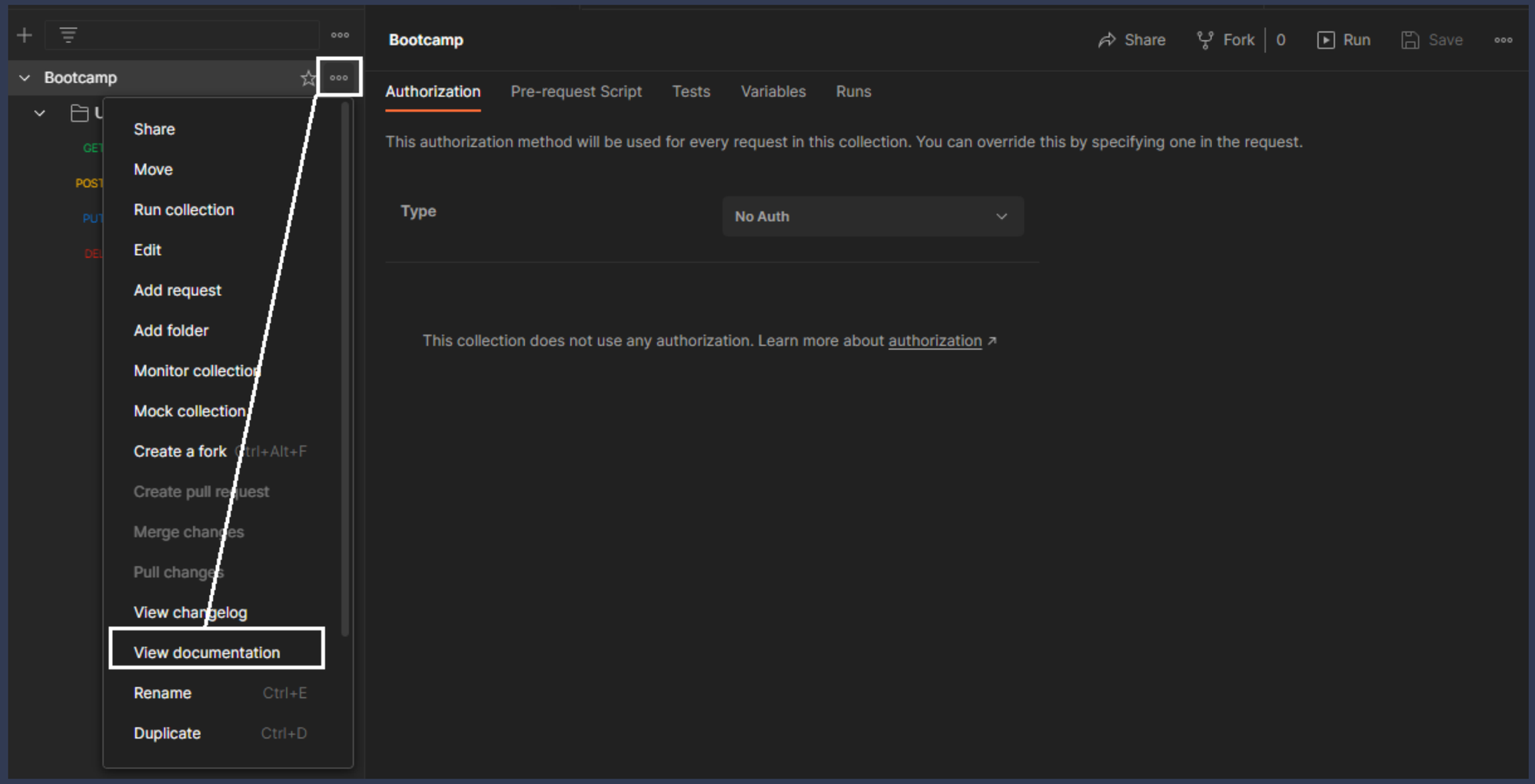
Secara tidak langsung JSON (JavaScript Object Notation) adalah kumpulan beberapa aray yang dikonversi oleh bahasa pemrograman PHP supaya FE bisa mengambil data yang telah dikonversi ke bahasa yang sifatnya client-side cth : JavaScript, TypeScript dll.

Langka Terakhir Membuat Restful API

```php
<?php

require_once('Database.php');

header('Content-Type: application/json');
header('Access-Control-Allow-Origin: *');
header('Access-Control-Allow-Methods: GET, POST, PUT, DELETE');
header('Access-Control-Allow-Headers: X-Requested-With');

class Users
{
    private string $table = 'users';
    private ?string $username;
    private ?string $password;
    public ?int $id;
    private ?object $statement;

```

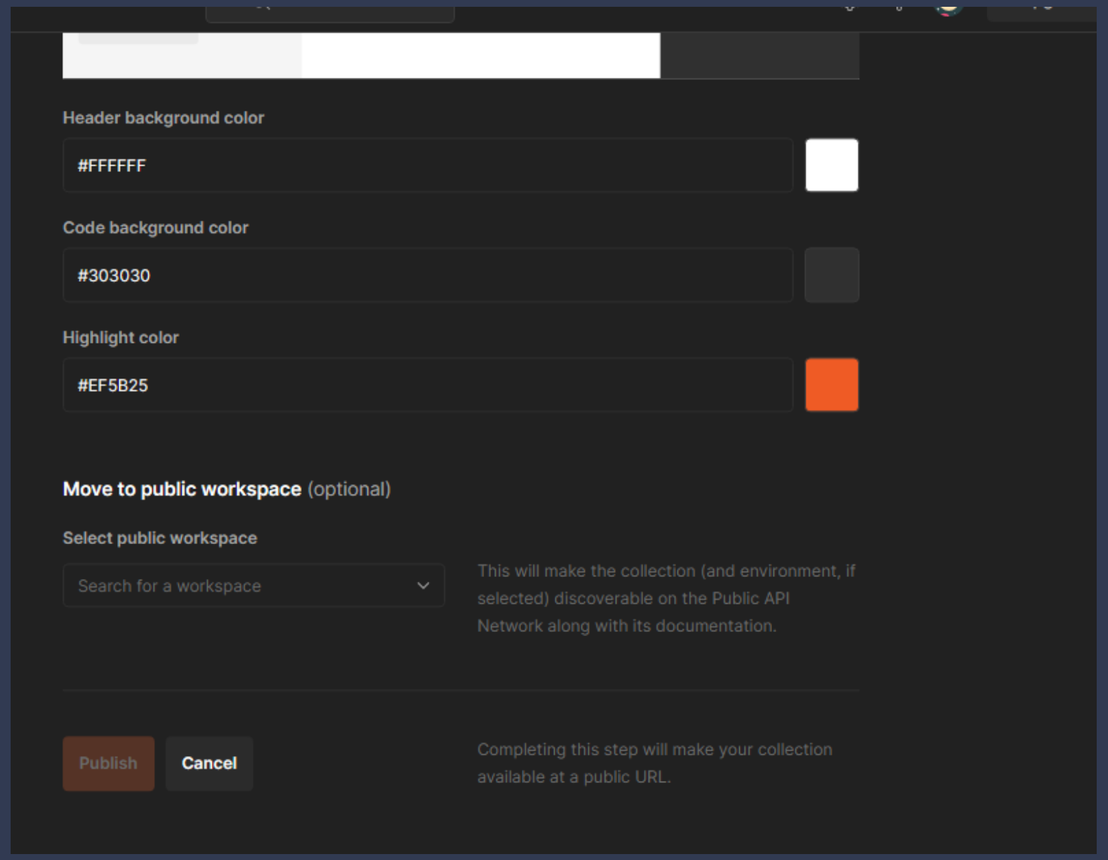
Setelah melalui 4 komponen penting dalam membuat RESTful API, selanjutnya dengan menambahkan beberapa header sebelum class Users supaya tidak terjadi error CORS (Cross-Origin Resource Sharing) kepada Front-End Developer, hal ini sering terjadi jika BE tidak mengijinkan untuk melakukan request method tertentu kepada FE Developer

Membuat Dokumentasi API Menggunakan Postman



Pilih tombol icon triple dot kemudian pilih view documentation

Publish Dokumentasi API



Setelah itu , kemudian pilih tombol publish untuk mendapatkan URL dokumentasi API yang untuk diserahkan kepada FE supaya RESTful API yang telah kita buat bisa diintegrasikan oleh FE Developer.

Kesimpulan

Setelah mempelajari beberapa materi back-end development, mulai dari fundamental hingga membuat sebuah API dengan konsep REST/RESTful API yang bertujuan untuk memudahkan kolaborasi antara back-end dan front-end developer, tetap semangat dan berlatih terus tentang back-end development. Sekian dari penulis, mohon maaf jika ada kesalahan dalam penulisan atau istilah asing buat yang awam, materi bootcamp ini akan terus disempurnakan seiring berkembangnya waktu.

Terimakasih