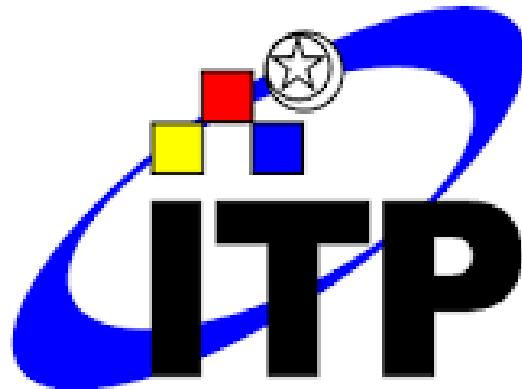


LAPORAN PEMOGRAMAN WEB



Disusun Oleh:

RAHMA (2020610025)

NABILA FITRI FITRIYANTI (2020610024)

Wulan Dari (2020610008)

Tahun Ajaran

2022/2023

➤ Table User

Scriptnya :

```
<?php

defined('BASEPATH') or exit('No direct script access allowed');

// This can be removed if you use __autoload() in config.php OR use Modular Extensions
/** @noinspection PhpIncludeInspection */
require APPPATH . '/libraries/REST_Controller.php';

// use namespace
use Restserver\Libraries\REST_Controller;

/**
 * This is an example of a few basic user interaction methods you could use
 * all done with a hardcoded array
 *
 * @package    CodeIgniter
 * @subpackage Rest Server
 * @category   Controller
 * @author    Phil Sturgeon, Chris Kacerguis
 * @license   MIT
 * @link      https://github.com/chriskacerguis/codeigniter-restserver
 */
class User extends REST_Controller
{

    function __construct()
    {
        // Construct the parent class
        parent::__construct();

        // Configure limits on our controller methods
        // Ensure you have created the 'limits' table and enabled 'limits' within
        application/config/rest.php
        $this->methods['users_get']['limit'] = 500; // 500 requests per hour per user/key
        $this->methods['users_post']['limit'] = 100; // 100 requests per hour per user/key
        $this->methods['users_delete']['limit'] = 50; // 50 requests per hour per user/key
    }
}
```

```

public function users_get()
{
    // Users from a data store e.g. database
    $users = [
        // ['id' => 1, 'name' => 'John', 'email' => 'john@example.com', 'fact' => 'Loves
        coding'],
        // ['id' => 2, 'name' => 'Jim', 'email' => 'jim@example.com', 'fact' => 'Developed
        on CodeIgniter'],
        // ['id' => 3, 'name' => 'Jane', 'email' => 'jane@example.com', 'fact' => 'Lives in the
        USA', ['hobbies' => ['guitar', 'cycling']]],
    ];

    $id = $this->get('username');

    // If the id parameter doesn't exist return all the users

    if ($id === NULL) {
        $users = $this->db->get("user")->result_array();
        // Check if the users data store contains users (in case the database result returns
        NULL)
        if ($users) {
            // Set the response and exit
            $this->response($users, REST_Controller::HTTP_OK); // OK (200) being the
            HTTP response code
        } else {
            // Set the response and exit
            $this->response([
                'status' => FALSE,
                'message' => 'No users were found'
            ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
            response code
        }
    }

    // Find and return a single record for a particular user.
    else {
        $id = (int) $id;

        // Validate the id.
        if ($id <= 0) {
            // Invalid id, set the response and exit.

```

```

        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); //  

        BAD_REQUEST (400) being the HTTP response code  

    }  
  

    // Get the user from the array, using the id as key for retrieval.  

    // Usually a model is to be used for this.  
  

    $user = NULL;  
  

    if (!empty($users)) {  

        foreach ($users as $key => $value) {  

            if (isset($value['username']) && $value['username'] === $id) {  

                $user = $value;  

            }
        }
    }  
  

    if (!empty($user)) {  

        $this->set_response($user, REST_Controller::HTTP_OK); // OK (200) being the  

        HTTP response code  

    } else {  

        $this->set_response([  

            'status' => FALSE,  

            'message' => 'User could not be found'  

        ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP  

        response code  

    }
}  
  

public function users_post()  

{
    // $this->some_model->update_user( ... );  

    $message = [  

        'username' => $this->post('username'),  

        'password' => $this->post('password'),  

        'nama' => $this->post('nama'),  

        'alamat' => $this->post('alamat'),  

        'email' => $this->post('email'),  

        'no_hp' => $this->post('no_hp'),  

        'level' => $this->post('level'),
}

```

```

];
$this->db->insert("user", $message);
$this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
$this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
}
function users_put()
{
$id = $this->put('username');
$message = [
'username' => $this->put('username'),
'password' => $this->put('password'),
'nama' => $this->put('nama'),
'alamat' => $this->put('alamat'),
'email' => $this->put('email'),
'no_hp' => $this->put('no_hp'),
'level' => $this->put('level'),
];
$this->db->where('username', $id);
$update = $this->db->update('user', $message);
if ($update) {
$this->response($message, 200);
} else {
$this->response(array('status' => 'fail', 502));
}
}

public function users_delete()
{
$id = $this->delete('username');
$this->db->where('username', $id);
$delete = $this->db->delete('user');

// Validate the id.
if ($delete) {
// Set the response and exit
$this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); // BAD_REQUEST
(400) being the HTTP response code
}
}

```

```

// $this->some_model->delete_something($id);
$message = [
    'username' => $id,
    'message' => 'Deleted the resource'
];

$this->set_response($message, REST_Controller::HTTP_NO_CONTENT); // NO_CONTENT (204) being the HTTP response code
}
}

```

Menampilkan database ke postman

The screenshot shows the Postman interface with a GET request to `http://localhost/nabila/ci-api/index.php/api/user/users`. The response status is 200 OK, and the response body contains the following JSON data:

```

[{"id": 4, "password": "1234", "nama": "wulan dari padang", "alamat": "abcdfgh", "email": "asd@gmail.com", "no_hp": "123", "level": "client"}, {"id": 11, "password": "2026610068", "nama": "wulan123", "alamat": "padang", "email": "as@gmail.com", "no_hp": "1234231", "level": "client"}]

```

Menambahkan Dari postman ke database

The screenshot shows the Postman interface with a POST request to `http://localhost/ci-api/index.php/api/user/users?username=2026610010`. The request body is set to `x-www-form-urlencoded` and contains the following data:

password	asd
nama	fitri
alamat	padang
email	as@gmail.com
no_hp	1234231
level	client

The response status is 201 Created, and the response body shows the newly added user record:

```

{"id": 2, "password": "asd", "nama": "fitri", "alamat": "padang", "email": "as@gmail.com", "no_hp": "1234231", "level": "client"}

```

Update

The screenshot shows the phpMyAdmin interface for the 'inventori' database. The 'user' table is selected. The table structure includes columns: username, password, nama, alamat, email, no_hp, and level. The data shows three users: landr, wulan, and Rahma, each with their respective details and a 'client' or 'admin' level.

	username	password	nama	alamat	email	no_hp	level
1	landr	wulan	qwee	gddgddh@gmail.com	081234567890		client
2	1234	wulandaripadang	abcefg	asd@gmail.com	133		client
3	wulan123	Wulan Dari	padang	wulann@gmail.com	1213342		client
4	dream	Rahma	jalan gajah mada	rahma@gmail.com	081234567812		admin

Delete

Before di database

This screenshot is identical to the one above, showing the 'user' table in the 'inventori' database with the same 3 rows of data: landr, wulan, and Rahma.

	username	password	nama	alamat	email	no_hp	level
1	landr	wulan	qwee	gddgddh@gmail.com	081234567890		client
2	1234	wulandaripadang	abcefg	asd@gmail.com	133		client
3	wulan123	Wulan Dari	padang	wulann@gmail.com	1213342		client
4	dream	Rahma	jalan gajah mada	rahma@gmail.com	081234567812		admin

Dari postman delete

The screenshot shows the Postman interface. A DELETE request is made to `http://localhost/nabila/ci-api/index.php/api/user/users`. In the 'Body' tab, there is a single form-data entry with key 'username' and value '2020610003'. The response status is 204 No Content.

After dipostman

The screenshot shows the phpMyAdmin interface connected to a MySQL database named 'inventori'. The 'user' table is selected, showing two rows of data:

	username	password	nama	alamat	email	no_hp	level
1	wulan123	2020610008	Wulan Dari	padang	wulan@gmail.com	1213342	client
2	dream	2020610025	Rahma	jalan gajah mada	rahma@gmail.com	081234567812	admin

➤ Table pengembalian

Scriptnya :

<?php

```
defined('BASEPATH') or exit('No direct script access allowed');
```

```
// This can be removed if you use __autoload() in config.php OR use Modular Extensions
/** @noinspection PhpIncludeInspection */
require APPPATH . '/libraries/REST_Controller.php';
```

```
// use namespace
use Restserver\Libraries\REST_Controller;
```

```

/**
 * This is an example of a few basic user interaction methods you could use
 * all done with a hardcoded array
 *
 * @package    CodeIgniter
 * @subpackage Rest Server
 * @category   Controller
 * @author     Phil Sturgeon, Chris Kacerguis
 * @license    MIT
 * @link       https://github.com/chriskacerguis/codeigniter-restserver
 */
class pengembalian extends REST_Controller
{

    function __construct()
    {
        // Construct the parent class
        parent::__construct();

        // Configure limits on our controller methods
        // Ensure you have created the 'limits' table and enabled 'limits' within
        application/config/rest.php
        $this->methods['users_get']['limit'] = 500; // 500 requests per hour per user/key
        $this->methods['users_post']['limit'] = 100; // 100 requests per hour per user/key
        $this->methods['users_delete']['limit'] = 50; // 50 requests per hour per user/key
    }

    public function users_get()
    {
        // Users from a data store e.g. database
        $users = [
            // ['id' => 1, 'name' => 'John', 'email' => 'john@example.com', 'fact' => 'Loves
            coding'],
            // ['id' => 2, 'name' => 'Jim', 'email' => 'jim@example.com', 'fact' => 'Developed
            on CodeIgniter'],
            // ['id' => 3, 'name' => 'Jane', 'email' => 'jane@example.com', 'fact' => 'Lives in the
            USA', ['hobbies' => ['guitar', 'cycling']]],
        ];
    }

    $id = $this->get('kd_kembali');

```

```

// If the id parameter doesn't exist return all the users

if ($id === NULL) {
    $users = $this->db->get("pengembalian")->result_array();
    // Check if the users data store contains users (in case the database result returns
    NULL)
    if ($users) {
        // Set the response and exit
        $this->response($users, REST_Controller::HTTP_OK); // OK (200) being the
        HTTP response code
    } else {
        // Set the response and exit
        $this->response([
            'status' => FALSE,
            'message' => 'No users were found'
        ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
        response code
    }
}

// Find and return a single record for a particular user.
else {
    $id = (int) $id;

    // Validate the id.
    if ($id <= 0) {
        // Invalid id, set the response and exit.
        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); //
        BAD_REQUEST (400) being the HTTP response code
    }

    // Get the user from the array, using the id as key for retrieval.
    // Usually a model is to be used for this.

    $user = NULL;

    if (!empty($users)) {
        foreach ($users as $key => $value) {
            if (isset($value['kd_kembali']) && $value['kd_kembali'] === $id) {
                $user = $value;
            }
        }
    }
}

```

```

        }
    }
}

if (!empty($user)) {
    $this->set_response($user, REST_Controller::HTTP_OK); // OK (200) being the
HTTP response code
} else {
    $this->set_response([
        'status' => FALSE,
        'message' => 'User could not be found'
    ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
response code
}
}

public function users_post()
{
// $this->some_model->update_user( ... );
$message = [
    'kd_kembali' => $this->post('kd_kembali'),
    'kd_pinjam' => $this->post('kd_pinjam'),
    'tgl_kembali' => $this->post('tgl_kembali'),
];

$this->db->insert("pengembalian", $message);
$this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
$this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
}

function users_put()
{
$id = $this->put('kd_kembali');
$message = [
    'kd_kembali' => $this->put('kd_kembali'),
    'kd_pinjam' => $this->put('kd_pinjam'),
    'tgl_kembali' => $this->put('tgl_kembali'),
]
}

```

```

];
$this->db->where('kd_kembali', $id);
$update = $this->db->update('pengembalian', $message);
if ($update) {
    $this->response($message,200);
} else {
    $this->response(array('status' => 'fail', 502));
}
}

public function users_delete()
{
    $id = $this->delete('kd_kembali');
    $this->db->where('kd_kembali', $id);
    $delete = $this->db->delete('pengembalian');

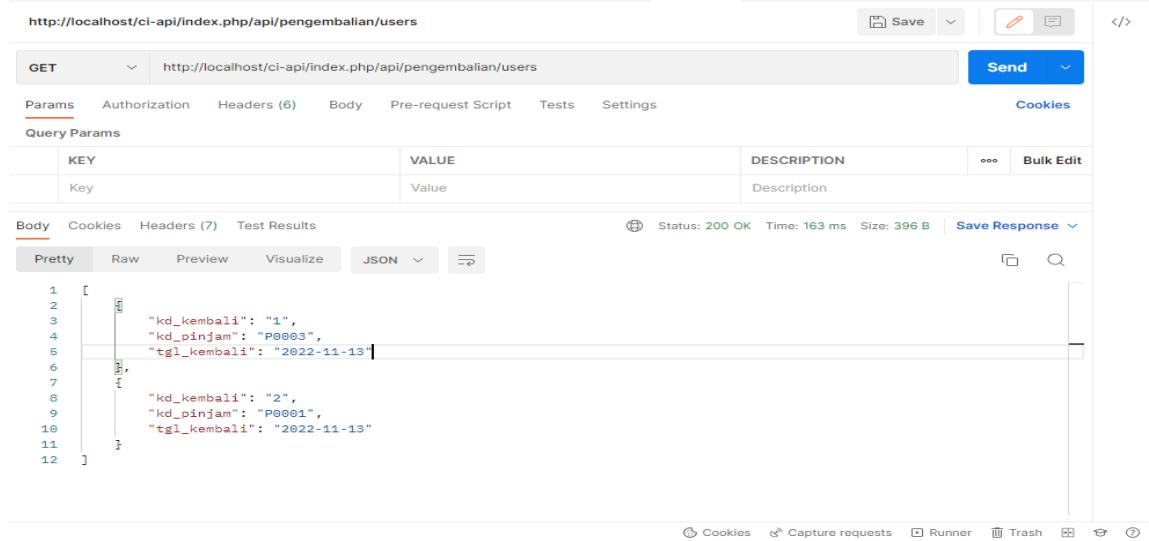
    // Validate the id.
    if ($delete) {
        // Set the response and exit
        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); // BAD_REQUEST
(400) being the HTTP response code
    }

    // $this->some_model->delete_something($id);
    $message = [
        'username' => $id,
        'message' => 'Deleted the resource'
    ];

    $this->set_response($message, REST_Controller::HTTP_NO_CONTENT); // NO_CONTENT (204) being the HTTP response code
}
}

```

Menampilkan database ke postman



http://localhost/ci-api/index.php/api/pengembalian/users

GET http://localhost/ci-api/index.php/api/pengembalian/users

Params Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| KEY | VALUE | DESCRIPTION | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description | |

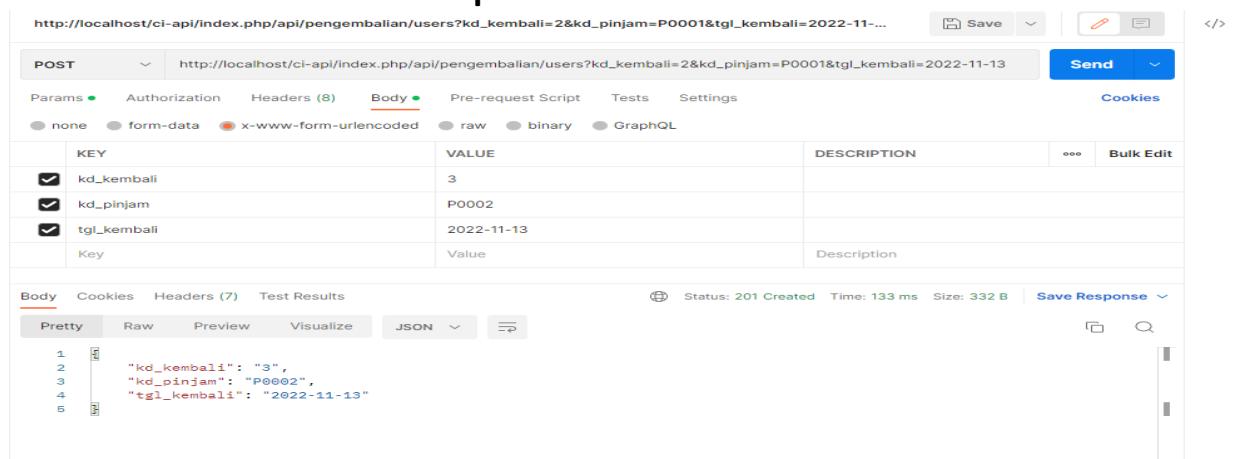
Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
[{"kd_kembali": "1", "kd_pinjam": "P0003", "tgl_kembali": "2022-11-13"}, {"kd_kembali": "2", "kd_pinjam": "P0001", "tgl_kembali": "2022-11-13"}]
```

Status: 200 OK Time: 163 ms Size: 396 B Save Response

Menambahkan database dari postman



http://localhost/ci-api/index.php/api/pengembalian/users?kd_kembali=2&kd_pinjam=P0001&tgl_kembali=2022-11-13

POST http://localhost/ci-api/index.php/api/pengembalian/users?kd_kembali=2&kd_pinjam=P0001&tgl_kembali=2022-11-13

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

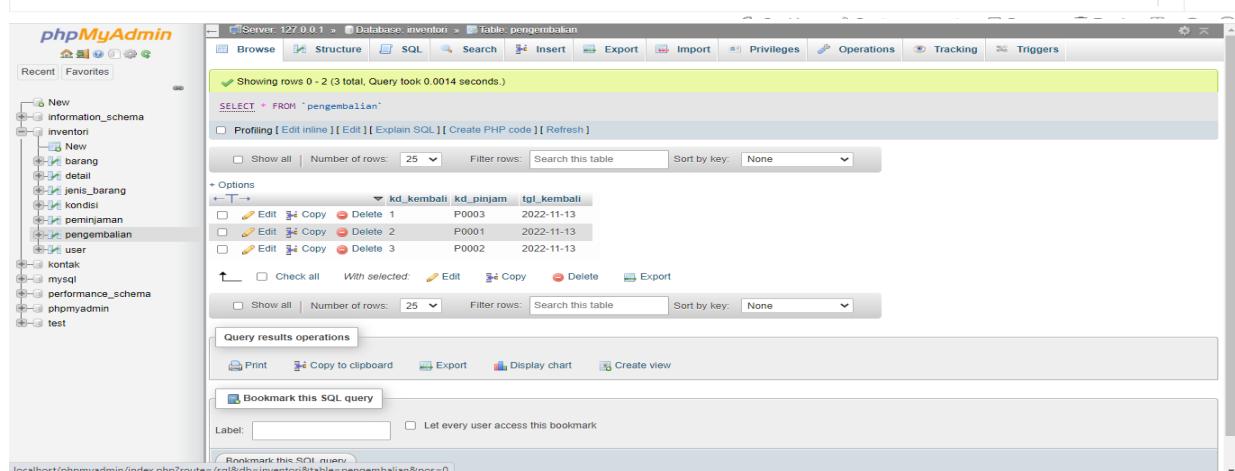
| KEY | VALUE | DESCRIPTION | Bulk Edit |
|-------------|------------|-------------|-----------|
| kd_kembali | 3 | | |
| kd_pinjam | P0002 | | |
| tgl_kembali | 2022-11-13 | | |
| Key | Value | Description | |

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{"kd_kembali": "3", "kd_pinjam": "P0002", "tgl_kembali": "2022-11-13"}
```

Status: 201 Created Time: 133 ms Size: 332 B Save Response



phpMyAdmin

Server: 127.0.0.1 > Database: inventori > Table: pengembalian

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Showing rows 0 - 2 (3 total), Query took 0.0014 seconds.

SELECT * FROM `pengembalian`

Options

| kd_kembali | kd_pinjam | tgl_kembali |
|------------|-----------|-------------|
| 1 | P0003 | 2022-11-13 |
| 2 | P0001 | 2022-11-13 |
| 3 | P0002 | 2022-11-13 |

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query

Label: Let every user access this bookmark

Update

Showing rows 0 - 2 (3 total, Query took 0.0010 seconds)

| | kd_kembali | kd_pinjam | tgl_kembali |
|---|------------|------------|-------------|
| 1 | P0003 | 2022-11-13 | |
| 2 | P0001 | 2022-11-13 | |
| 3 | P0004 | 2022-11-13 | |

PUT http://localhost/ci-api/index.php/api/pengembalian/users

| Params | Authorization | Headers (12) | Body | Pre-request Script | Tests | Settings |
|--------|---------------|-----------------------|--|--------------------|-------|----------|
| none | form-data | x-www-form-urlencoded | kd_kembali: 3
kd_pinjam: P0004
tgl_kembali: 2022-11-13 | | | |

Delete

Before di database

Showing rows 0 - 1 (2 total, Query took 0.0011 seconds)

| | kd_kembali | kd_pinjam | tgl_kembali |
|---|------------|------------|-------------|
| 1 | P0003 | 2022-11-13 | |
| 2 | P0001 | 2022-11-13 | |

Dari postman

DELETE http://localhost/ci-api/index.php/api/pengembalian/users?kd_kembali=2&kd_pinjam=P0001&tgl_kembali=2022-11-13

| KEY | VALUE | DESCRIPTION |
|-------------|------------|-------------|
| kd_kembali | 2 | |
| kd_pinjam | P0001 | |
| tgl_kembali | 2022-11-13 | |
| Key | Value | Description |

After dipostman

➤ Tabel peminjaman

Scriptnya :

```
<?php
```

```
defined('BASEPATH') or exit('No direct script access allowed');
```

```
// This can be removed if you use __autoload() in config.php OR use Modular Extensions
```

```
/** @noinspection PhpIncludeInspection */
```

```
require APPPATH . '/libraries/REST_Controller.php';
```

```
// use namespace
```

```
use Restserver\Libraries\REST_Controller;
```

```
/**
```

```
* This is an example of a few basic user interaction methods you could use
```

```
* all done with a hardcoded array
```

```
*
```

```
* @package    CodeIgniter
```

```
* @subpackage Rest Server
```

```
* @category   Controller
```

```
* @author    Phil Sturgeon, Chris Kacerguis
```

```
* @license   MIT
```

```
* @link      https://github.com/chriskacerguis/codeigniter-restserver
```

```
*/
```

```
class peminjaman extends REST_Controller
```

```
{
```

```
function __construct()
```

```
{
```

```

// Construct the parent class
parent::__construct();

// Configure limits on our controller methods
// Ensure you have created the 'limits' table and enabled 'limits' within
application/config/rest.php
$this->methods['users_get']['limit'] = 500; // 500 requests per hour per user/key
$this->methods['users_post']['limit'] = 100; // 100 requests per hour per user/key
$this->methods['users_delete']['limit'] = 50; // 50 requests per hour per user/key
}

public function users_get()
{
    // Users from a data store e.g. database
    $users = [
        // ['id' => 1, 'name' => 'John', 'email' => 'john@example.com', 'fact' => 'Loves
        coding'],
        // ['id' => 2, 'name' => 'Jim', 'email' => 'jim@example.com', 'fact' => 'Developed
        on CodeIgniter'],
        // ['id' => 3, 'name' => 'Jane', 'email' => 'jane@example.com', 'fact' => 'Lives in the
        USA', ['hobbies' => ['guitar', 'cycling']]],
    ];
}

$id = $this->get('kd_pinjam');

// If the id parameter doesn't exist return all the users

if ($id === NULL) {
    $users = $this->db->get("peminjaman")->result_array();
    // Check if the users data store contains users (in case the database result returns
    NULL)
    if ($users) {
        // Set the response and exit
        $this->response($users, REST_Controller::HTTP_OK); // OK (200) being the
        HTTP response code
    } else {
        // Set the response and exit
        $this->response([
            'status' => FALSE,
            'message' => 'No users were found'
        ]);
    }
}

```

```

        ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
response code
    }
}

// Find and return a single record for a particular user.
else {
    $id = (int) $id;

    // Validate the id.
    if ($id <= 0) {
        // Invalid id, set the response and exit.
        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); //
BAD_REQUEST (400) being the HTTP response code
    }

    // Get the user from the array, using the id as key for retrieval.
    // Usually a model is to be used for this.

    $user = NULL;

    if (!empty($users)) {
        foreach ($users as $key => $value) {
            if (isset($value['kd_pinjam']) && $value['kd_pinjam'] === $id) {
                $user = $value;
            }
        }
    }

    if (!empty($user)) {
        $this->set_response($user, REST_Controller::HTTP_OK); // OK (200) being the
HTTP response code
    } else {
        $this->set_response([
            'status' => FALSE,
            'message' => 'User could not be found'
        ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
response code
    }
}

```

```

public function users_post()
{
    // $this->some_model->update_user( ... );
    $message = [
        'kd_pinjam' => $this->post('kd_pinjam'),
        'username' => $this->post('username'),
        'kd_barang' => $this->post('kd_barang'),
        'tgl pinjam' => $this->post('tgl pinjam'),
        'status_kembali' => $this->post('status_kembali'),
    ];

    $this->db->insert("peminjaman", $message);
    $this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
    (201) being the HTTP response code
    $this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
    (201) being the HTTP response code
}

function users_put()
{
    $id = $this->put('kd_pinjam');
    $message = [
        'kd_pinjam' => $this->put('kd_pinjam'),
        'username' => $this->put('username'),
        'kd_barang' => $this->put('kd_barang'),
        'tgl pinjam' => $this->put('tgl pinjam'),
        'status_kembali' => $this->put('status_kembali'),
    ];

    $this->db->where('kd_pinjam', $id);
    $update = $this->db->update('peminjaman', $message);
    if ($update) {
        $this->response($message, 200);
    } else {
        $this->response(array('status' => 'fail', 502));
    }
}

public function users_delete()
{
}

```

```

$id = $this->delete('kd_pinjam');
$this->db->where('kd_pinjam', $id);
$delete = $this->db->delete('peminjaman');

// Validate the id.
if ($delete) {
    // Set the response and exit
    $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); // BAD_REQUEST
(400) being the HTTP response code
}

// $this->some_model->delete_something($id);
$message = [
    'kd_pinjam' => $id,
    'message' => 'Deleted the resource'
];

$this->set_response($message, REST_Controller::HTTP_NO_CONTENT); // NO_CONTENT (204) being the HTTP response code
}
}

```

Menampilkan database ke postman

The screenshot shows the Postman application interface. At the top, there is a header bar with several tabs and a 'No Environment' button. Below the header, the URL 'http://localhost:8000/api/peminjaman/users' is entered into the address bar. To the right of the address bar are buttons for 'Save', 'Edit', and 'Send'. The 'Send' button is highlighted in blue. Underneath the address bar, there are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Body' tab is currently selected. In the 'Body' section, there is a table with columns 'KEY', 'VALUE', 'DESCRIPTION', and 'Bulk Edit'. Below the table, there are tabs for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. The JSON response is displayed in a code editor-like area:

```

1 [ ]
2 {
3     "kd_pinjam": "P0001",
4     "username": "2026100008",
5     "kd_barang": "BR0001",
6     "tgl_pinjam": "2022-11-13",
7     "status_kembali": "kembali"
8 },
9 {
10     "kd_pinjam": "P0002",
11     "username": "2026100001",
12     "kd_barang": "BR0003",
13     "tgl_pinjam": "2022-11-13",
14     "status_kembali": ""
15 },
16 {
17     "kd_pinjam": "P0003",
18     ...
19 }

```

At the bottom of the interface, there are buttons for 'Online', 'Find and Replace', 'Console', 'Cookies', 'Capture requests', 'Runner', 'Trash', and a help icon.

Menambahkan database dari postman

The screenshot shows two windows side-by-side. On the left is the phpMyAdmin interface for the 'inventor' database, specifically the 'peminjaman' table. It displays three rows of data with columns: kd_pinjam, username, and kd_barang. On the right is the Postman application. In the 'POST' tab, a request is being made to `http://localhost/ci-api/index.php/api/peminjaman/users`. The 'Body' tab is selected, showing a JSON payload:

```
1 "kd_pinjam": "P0004",
2 "username": "2020610025",
3 "kd_barang": "BR0015",
4 "tgl_pinjam": "2022-11-13",
5 "status_kembali": "kembali"
```

Update

The screenshot shows two windows side-by-side. On the left is the phpMyAdmin interface for the 'inventor' database, specifically the 'peminjaman' table. It displays three rows of data with columns: kd_pinjam, username, kd_barang, tgl_pinjam, and status_kemb. On the right is the Postman application. In the 'PUT' tab, a request is being made to `http://localhost/ci-api/index.php/api/peminjaman/users`. The 'Body' tab is selected, showing a JSON payload:

```
1 "username": "2020610001",
2 "kd_barang": "BR0017",
3 "tgl_pinjam": "2022-11-13",
4 "status_kembali": "kembali"
```

Delete Before di database

After dipostman

localhost / localhost/ci-api/index.php/api/peminjaman

Server: 127.0.0.1 > Database: inventori > Table: peminjaman

Showing rows 0 - 2 (3 total, Query took 0.0012 seconds.)

SELECT * FROM `peminjaman`

| | kd_pinjam | username | kd_barang | tgl_pinjam | status_kembali | |
|--------------------------|-----------|----------|------------|------------|----------------|---------|
| <input type="checkbox"/> | Edit | P0001 | 2020610008 | BR0001 | 2022-11-13 | kembali |
| <input type="checkbox"/> | Edit | P0002 | 2020610001 | BR0003 | 2022-11-13 | |
| <input type="checkbox"/> | Edit | P0003 | 2020610008 | BR0014 | 2022-11-13 | kembali |

Query results operations

Bookmark this SQL query

Console

Let every user access this bookmark

localhost / localhost/ci-api/index.php/api/peminjaman/users

DELETE http://localhost/ci-api/index.php/api/peminjaman/users

Params Authorization Headers (8) Body Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

| KEY | VALUE |
|----------------|------------|
| kd_pinjam | P0003 |
| username | 2020610008 |
| kd_barang | BR0014 |
| tgl_pinjam | 2022-11-13 |
| status_kembali | kembali |

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

1

➤ Table kondisi

Scriptnya:

<?php

```
defined('BASEPATH') or exit('No direct script access allowed');
```

```
// This can be removed if you use __autoload() in config.php OR use Modular Extensions
/** @noinspection PhpIncludeInspection */
require APPPATH . '/libraries/REST_Controller.php';
```

```
// use namespace
use Restserver\Libraries\REST_Controller;
```

```

/**
 * This is an example of a few basic user interaction methods you could use
 * all done with a hardcoded array
 *
 * @package    CodeIgniter
 * @subpackage Rest Server
 * @category   Controller
 * @author     Phil Sturgeon, Chris Kacerguis
 * @license    MIT
 * @link       https://github.com/chriskacerguis/codeigniter-restserver
 */
class kondisi extends REST_Controller
{

    function __construct()
    {
        // Construct the parent class
        parent::__construct();

        // Configure limits on our controller methods
        // Ensure you have created the 'limits' table and enabled 'limits' within
        application/config/rest.php
        $this->methods['users_get']['limit'] = 500; // 500 requests per hour per user/key
        $this->methods['users_post']['limit'] = 100; // 100 requests per hour per user/key
        $this->methods['users_delete']['limit'] = 50; // 50 requests per hour per user/key
    }

    public function users_get()
    {
        // Users from a data store e.g. database
        $users = [
            // ['id' => 1, 'name' => 'John', 'email' => 'john@example.com', 'fact' => 'Loves
            coding'],
            // ['id' => 2, 'name' => 'Jim', 'email' => 'jim@example.com', 'fact' => 'Developed
            on CodeIgniter'],
            // ['id' => 3, 'name' => 'Jane', 'email' => 'jane@example.com', 'fact' => 'Lives in the
            USA', ['hobbies' => ['guitar', 'cycling']]],
        ];
    }

    $id = $this->get('kd_kondisi');
}

```

```

// If the id parameter doesn't exist return all the users

if ($id === NULL) {
    $users = $this->db->get("kondisi")->result_array();
    // Check if the users data store contains users (in case the database result returns
    NULL)
    if ($users) {
        // Set the response and exit
        $this->response($users, REST_Controller::HTTP_OK); // OK (200) being the
        HTTP response code
    } else {
        // Set the response and exit
        $this->response([
            'status' => FALSE,
            'message' => 'No users were found'
        ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
        response code
    }
}

// Find and return a single record for a particular user.
else {
    $id = (int) $id;

    // Validate the id.
    if ($id <= 0) {
        // Invalid id, set the response and exit.
        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); //
        BAD_REQUEST (400) being the HTTP response code
    }

    // Get the user from the array, using the id as key for retrieval.
    // Usually a model is to be used for this.

    $user = NULL;

    if (!empty($users)) {
        foreach ($users as $key => $value) {
            if (isset($value['kd_kondisi']) && $value['kd_kondisi'] === $id) {
                $user = $value;
            }
        }
    }
}

```

```

        }
    }
}

if (!empty($user)) {
    $this->set_response($user, REST_Controller::HTTP_OK); // OK (200) being the
HTTP response code
} else {
    $this->set_response([
        'status' => FALSE,
        'message' => 'User could not be found'
    ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
response code
}
}

public function users_post()
{
// $this->some_model->update_user( ... );
$message = [
    'kd_kondisi' => $this->post('kd_kondisi'),
    'kondisi' => $this->post('kondisi'),

];
$this->db->insert("kondisi", $message);
$this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
$this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
}

function users_put()
{
$id = $this->put('kd_kondisi');
$message = [
    'kd_kondisi' => $this->put('kd_kondisi'),
    'kondisi' => $this->put('kondisi'),

];
$this->db->where('kd_kondisi', $id);

```

```

$update = $this->db->update('kondisi', $message);
if ($update) {
    $this->response($message,200);
} else {
    $this->response(array('status' => 'fail', 502));
}
}

public function users_delete()
{
    $id = $this->delete('kd_kondisi');
    $this->db->where('kd_kondisi', $id);
    $delete = $this->db->delete('kondisi');

    // Validate the id.
    if ($delete) {
        // Set the response and exit
        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); // BAD_REQUEST
(400) being the HTTP response code
    }

    // $this->some_model->delete_something($id);
    $message = [
        'kd_kondisi' => $id,
        'message' => 'Deleted the resource'
    ];

    $this->set_response($message, REST_Controller::HTTP_NO_CONTENT); // NO_CONTENT (204) being the HTTP response code
}
}

```

Menampilkan database ke postman

http://localhost/ci-api/index.php/api/kondisi/users

GET http://localhost/ci-api/index.php/api/kondisi/users

| KEY | VALUE | DESCRIPTION | Bulk Edit |
|-----|-------|-------------|-----------|
| Key | Value | Description | |

```

15      "kd_kondisi": "04",
16      "kondisi": "zusak"
17      ,
18      :
19      "kd_kondisi": "05",
20      "kondisi": "Rusak Parah"
21      ,
22      :
23      "kd_kondisi": "06",
24      "kondisi": "Lama"
25      ,
26      :

```

Menambahkan database dari postman

localhost / T... Membuat Lin... (3) WhatsApp +

localhost/phpmyadmin/index.php?route=/sql...

Server: 127.0.0.1 Database: inventori Table: kondisi

Showing rows 0 - 5 (6 total, Query took 0.0010 seconds.)

SELECT * FROM `kondisi`

| | kd_kondisi | kondisi |
|----|-------------|---------|
| 01 | Baru | |
| 02 | Bagus | |
| 03 | Layak Pakai | |
| 04 | baru | |
| 05 | Rusak Parah | |
| 06 | Lama | |

POST http://localhost/ci-api/index.php/api/kondisi/users

| Params | Authorization | Headers (12) | Body | Pre-request Script | Tests | Settings |
|--|---------------|-----------------------|------|--------------------|---------|----------|
| none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | |
| <input checked="" type="checkbox"/> kd_kondisi | 04 | | | | | |
| <input checked="" type="checkbox"/> kondisi | baru | | | | | |
| Key | Value | | | | | |

```

1      "kd_kondisi": "04",
2      "kondisi": "baru"
3      ,
4      :

```

Update

Showing rows 0 - 5 (6 total). Query took 0.0014 seconds.

```
SELECT * FROM `kondisi`
```

| | kd_kondisi | kondisi | | | |
|--------------------------|------------|---------|--------|----|-------------|
| <input type="checkbox"/> | Edit | Copy | Delete | 01 | Baru |
| <input type="checkbox"/> | Edit | Copy | Delete | 02 | Bagus |
| <input type="checkbox"/> | Edit | Copy | Delete | 03 | Layak Pakai |
| <input type="checkbox"/> | Edit | Copy | Delete | 04 | layak pakai |
| <input type="checkbox"/> | Edit | Copy | Delete | 05 | Rusak Parah |
| <input type="checkbox"/> | Edit | Copy | Delete | 06 | Lama |

PUT http://localhost/ci-api/index.php/api/kondisi/users

Params Authorization Headers (12) Body Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

kd_kondisi 04
 kondisi layak pakai

Key Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```

1: "kd_kondisi": "04",
2: "kondisi": "layak pakai"

```

Delete

Before didatabase

Showing rows 0 - 5 (6 total). Query took 0.0018 seconds.

```
SELECT * FROM `kondisi`
```

| | kd_kondisi | kondisi | | | |
|--------------------------|------------|---------|--------|----|-------------|
| <input type="checkbox"/> | Edit | Copy | Delete | 01 | Baru |
| <input type="checkbox"/> | Edit | Copy | Delete | 02 | Bagus |
| <input type="checkbox"/> | Edit | Copy | Delete | 03 | Layak Pakai |
| <input type="checkbox"/> | Edit | Copy | Delete | 04 | rusak |
| <input type="checkbox"/> | Edit | Copy | Delete | 05 | Rusak Parah |
| <input type="checkbox"/> | Edit | Copy | Delete | 06 | Lama |

After dipostman

The screenshot shows two windows side-by-side. On the left, the phpMyAdmin interface displays the database structure for 'inventori'. A table named 'kondisi' is selected, showing six rows with columns 'kd_kondisi' and 'kondisi'. The rows are: 01 Baru, 02 Bagus, 03 Layak Pakai, 04 Rusak Parah, 05 Rusak Parah, and 06 Lama. On the right, a Postman window shows a DELETE request to 'http://localhost/ci-api/index.php/api/kondisi/users'. The 'Body' tab is selected, showing a key-value pair: 'kd_kondisi' with value '04' and 'kondisi' with value 'rusak'.

➤ Table jenis_barang

Scriptnya :

```
<?php
```

```
defined('BASEPATH') or exit('No direct script access allowed');
```

```
// This can be removed if you use __autoload() in config.php OR use Modular Extensions
/** @noinspection PhpIncludeInspection */
require APPPATH . '/libraries/REST_Controller';
```

```
// use namespace
use Restserver\Libraries\REST_Controller;
```

```
/**
 * This is an example of a few basic user interaction methods you could use
 * all done with a hardcoded array
 *
 * @package    CodeIgniter
 * @subpackage Rest Server
 * @category   Controller
 * @author    Phil Sturgeon, Chris Kacerguis
 * @license   MIT
 * @link      https://github.com/chriskacerguis/codeigniter-restserver
 */
class jenis_barang extends REST_Controller
```

```

{

function __construct()
{
    // Construct the parent class
    parent::__construct();

    // Configure limits on our controller methods
    // Ensure you have created the 'limits' table and enabled 'limits' within
    application/config/rest.php
    $this->methods['users_get']['limit'] = 500; // 500 requests per hour per user/key
    $this->methods['users_post']['limit'] = 100; // 100 requests per hour per user/key
    $this->methods['users_delete']['limit'] = 50; // 50 requests per hour per user/key
}

public function users_get()
{
    // Users from a data store e.g. database
    $users = [
        // ['id' => 1, 'name' => 'John', 'email' => 'john@example.com', 'fact' => 'Loves
        coding'],
        // ['id' => 2, 'name' => 'Jim', 'email' => 'jim@example.com', 'fact' => 'Developed
        on CodeIgniter'],
        // ['id' => 3, 'name' => 'Jane', 'email' => 'jane@example.com', 'fact' => 'Lives in the
        USA', ['hobbies' => ['guitar', 'cycling']]],
    ];

    $id = $this->get('kd_jenis');

    // If the id parameter doesn't exist return all the users

    if ($id === NULL) {
        $users = $this->db->get("jenis_barang")->result_array();
        // Check if the users data store contains users (in case the database result returns
        NULL)
        if ($users) {
            // Set the response and exit
            $this->response($users, REST_Controller::HTTP_OK); // OK (200) being the
            HTTP response code
        } else {
            // Set the response and exit

```

```

$this->response([
    'status' => FALSE,
    'message' => 'No users were found'
], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
response code
}

}

// Find and return a single record for a particular user.
else {
    $id = (int) $id;

    // Validate the id.
    if ($id <= 0) {
        // Invalid id, set the response and exit.
        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); //
BAD_REQUEST (400) being the HTTP response code
    }

    // Get the user from the array, using the id as key for retrieval.
    // Usually a model is to be used for this.

    $user = NULL;

    if (!empty($users)) {
        foreach ($users as $key => $value) {
            if (isset($value['kd_jenis']) && $value['kd_jenis'] === $id) {
                $user = $value;
            }
        }
    }
}

if (!empty($user)) {
    $this->set_response($user, REST_Controller::HTTP_OK); // OK (200) being the
HTTP response code
} else {
    $this->set_response([
        'status' => FALSE,
        'message' => 'User could not be found'
], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
response code
}

```

```

        }
    }
}

public function users_post()
{
    // $this->some_model->update_user( ... );
    $message = [
        'kd_jenis' => $this->post('kd_jenis'),
        'jenis' => $this->post('jenis'),
    ];

    $this->db->insert("jenis_barang", $message);
    $this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
    $this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
}

function users_put()
{
    $id = $this->put('kd_jenis');
    $message = [
        'kd_jenis' => $this->put('kd_jenis'),
        'jenis' => $this->put('jenis'),
    ];
    $this->db->where('kd_jenis', $id);
    $update = $this->db->update('jenis_barang', $message);
    if ($update) {
        $this->response($message, 200);
    } else {
        $this->response(array('status' => 'fail', 502));
    }
}

public function users_delete()
{
    $id = $this->delete('kd_jenis');
    $this->db->where('kd_jenis', $id);
    $delete = $this->db->delete('jenis_barang');
}

```

```

// Validate the id.
if ($delete) {
    // Set the response and exit
    $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); // BAD_REQUEST
    (400) being the HTTP response code
}

// $this->some_model->delete_something($id);
$message = [
    'kd_jenis' => $id,
    'message' => 'Deleted the resource'
];

$this->set_response($message, REST_Controller::HTTP_NO_CONTENT); // NO_CONTENT (204) being the HTTP response code
}
}

```

Menampilkan database ke postman

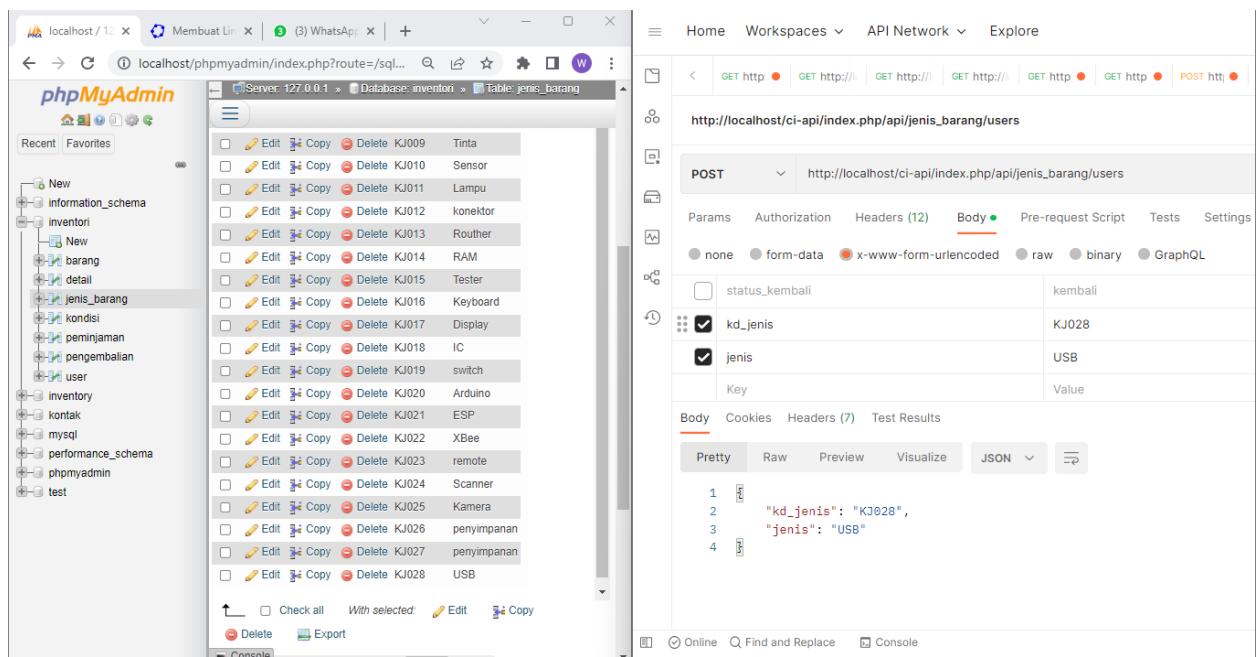
The screenshot shows the Postman application interface. A GET request is made to `http://localhost/ci-api/index.php/api/jenis_barang/users`. The response status is 200 OK, and the response body is displayed in Pretty JSON format:

```

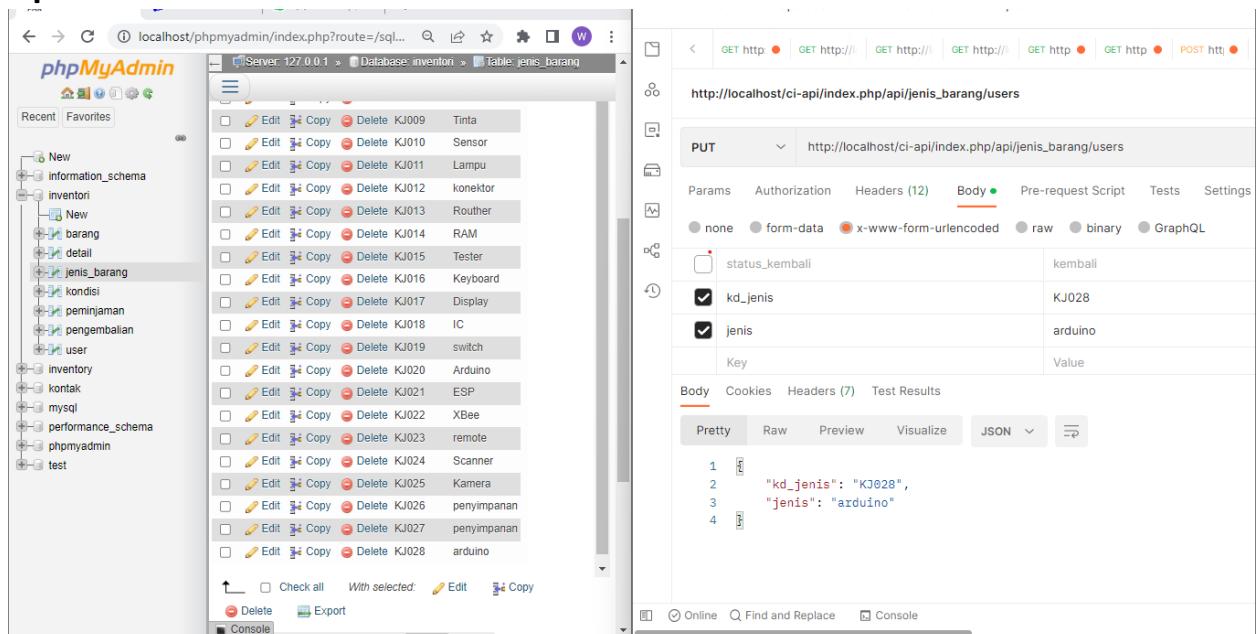
1  [
2   {
3     "kd_jenis": "KJ001",
4     "jenis": "cat"
5   },
6   {
7     "kd_jenis": "KJ002",
8     "jenis": "kabel"
9   },
10  {
11    "kd_jenis": "KJ003",
12    ...
13  }
]

```

Menambahkan database dari postman



Update



Delete Before didatabase

After dipostman

➤ Table detail

Scriptnya :

```
<?php
```

```
defined('BASEPATH') or exit('No direct script access allowed');
```

```
// This can be removed if you use __autoload() in config.php OR use Modular Extensions
/** @noinspection PhpIncludeInspection */
require APPPATH . '/libraries/REST_Controller.php';
```

```

// use namespace
use Restserver\Libraries\REST_Controller;

/**
 * This is an example of a few basic user interaction methods you could use
 * all done with a hardcoded array
 *
 * @package    CodeIgniter
 * @subpackage Rest Server
 * @category   Controller
 * @author     Phil Sturgeon, Chris Kacerguis
 * @license    MIT
 * @link       https://github.com/chriskacerguis/codeigniter-restserver
 */
class detail extends REST_Controller
{

    function __construct()
    {
        // Construct the parent class
        parent::__construct();

        // Configure limits on our controller methods
        // Ensure you have created the 'limits' table and enabled 'limits' within
        application/config/rest.php
        $this->methods['users_get']['limit'] = 500; // 500 requests per hour per user/key
        $this->methods['users_post']['limit'] = 100; // 100 requests per hour per user/key
        $this->methods['users_delete']['limit'] = 50; // 50 requests per hour per user/key
    }

    public function users_get()
    {
        // Users from a data store e.g. database
        $users = [
            // ['id' => 1, 'name' => 'John', 'email' => 'john@example.com', 'fact' => 'Loves
            coding'],
            // ['id' => 2, 'name' => 'Jim', 'email' => 'jim@example.com', 'fact' => 'Developed
            on CodeIgniter'],
            // ['id' => 3, 'name' => 'Jane', 'email' => 'jane@example.com', 'fact' => 'Lives in the
            USA', ['hobbies' => ['guitar', 'cycling']]],
    }
}

```

```

];
$id = $this->get('kd_detail');

// If the id parameter doesn't exist return all the users

if ($id === NULL) {
    $users = $this->db->get("detail")->result_array();
    // Check if the users data store contains users (in case the database result returns
    NULL)
    if ($users) {
        // Set the response and exit
        $this->response($users, REST_Controller::HTTP_OK); // OK (200) being the
        HTTP response code
    } else {
        // Set the response and exit
        $this->response([
            'status' => FALSE,
            'message' => 'No users were found'
        ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
        response code
    }
}

// Find and return a single record for a particular user.
else {
    $id = (int) $id;

    // Validate the id.
    if ($id <= 0) {
        // Invalid id, set the response and exit.
        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); //
        BAD_REQUEST (400) being the HTTP response code
    }

    // Get the user from the array, using the id as key for retrieval.
    // Usually a model is to be used for this.

    $user = NULL;

    if (!empty($users)) {

```

```

foreach ($users as $key => $value) {
    if (isset($value['kd_detail']) && $value['kd_detail'] === $id) {
        $user = $value;
    }
}
}

if (!empty($user)) {
    $this->set_response($user, REST_Controller::HTTP_OK); // OK (200) being the
HTTP response code
} else {
    $this->set_response([
        'status' => FALSE,
        'message' => 'User could not be found'
    ], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
response code
}

public function users_post()
{
// $this->some_model->update_user( ... );
$message = [
    'kd_detail' => 100,
    'kd_barang' => $this->post('kd_barang'),
    'kd_kondisi' => $this->post('kd_kondisi'),
    'jumlah' => $this->post('jumlah'),


];
$this->db->insert("detail", $message);
$this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
$this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
}

function users_put()
{
$id = $this->put('kd_detail');
$message = [

```

```

'kd_detail' => 100,
'kd_barang' => $this->put('kd_barang'),
'kd_kondisi' => $this->put('kd_kondisi'),
'jumlah' => $this->put('jumlah'),

];
$this->db->where('kd_detail', $id);
$update = $this->db->update('detail', $message);
if ($update) {
    $this->response($message,200);
} else {
    $this->response(array('status' => 'fail', 502));
}
}

public function users_delete()
{
    $id = $this->delete('kd_detail');
    $this->db->where('kd_detail', $id);
    $delete = $this->db->delete('detail');

    // Validate the id.
    if ($delete) {
        // Set the response and exit
        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); // BAD_REQUEST
        (400) being the HTTP response code
    }

    // $this->some_model->delete_something($id);
    $message = [
        'kd_detail' => $id,
        'message' => 'Deleted the resource'
    ];

    $this->set_response($message, REST_Controller::HTTP_NO_CONTENT); // NO_CONTENT (204) being the HTTP response code
}
}

```

Menampilkan database ke postman

http://localhost/ci-api/index.php/api/detail/users

GET http://localhost/ci-api/index.php/api/detail/users

| KEY | VALUE | DESCRIPTION |
|-----|-------|-------------|
| Key | Value | Description |

```

1 [
2   {
3     "kd_detail": "1",
4     "kd_barang": "BR0001",
5     "kd_kondisi": "02",
6     "jumlah": "1"
7   },
8   {
9     "kd_detail": "101",
10    ...
11  }
12 ]
  
```

Status: 200 OK Time: 250 ms Size: 406 B Save Response

Menambahkan database dari postman

localhost / 12 Membuat Lin WhatsApp +

localhost/phpmyadmin/index.php?route=sq... Server: 127.0.0.1 Database: inventon Table: detail

Showing rows 0 - 1 (2 total, Query took 0.0051 seconds.)

SELECT * FROM `detail`

POST http://localhost/ci-api/index.php/api/detail/users

| Body | Params | Authorization | Headers (12) | Body | Pre-request Script | Tests | Settings |
|---|--------|---------------|--------------|---|--------------------|-------|----------|
| <input checked="" type="radio"/> none <input type="radio"/> form-data <input checked="" type="radio"/> x-www-form-urlencoded <input type="radio"/> raw <input type="radio"/> binary <input type="radio"/> GraphQL | | | | <input type="checkbox"/> jenis: arduino <input checked="" type="checkbox"/> kd_detail: 101 <input checked="" type="checkbox"/> kd_barang: BR0004 <input checked="" type="checkbox"/> kd_kondisi: 02 <input checked="" type="checkbox"/> jumlah: 3 | | | |

```

3   "kd_detail": "101",
4   "kd_barang": "BR0004",
5   "kd_kondisi": "02",
6   "jumlah": "3"
  
```

Update

The screenshot shows a dual-pane interface. On the left, a browser window displays the PHPMyAdmin 'Table: detail' page with a query result table:

| | kd_detail | kd_barang | kd_kondisi | jumlah |
|--------------------------|-----------|-----------|------------|--------|
| <input type="checkbox"/> | Edit | Copy | Delete | 100 |
| <input type="checkbox"/> | Edit | Copy | Delete | 101 |

On the right, a terminal window shows a cURL command with a JSON body:

```
PUT http://localhost/ci-api/index.php/api/detail/users
Body
{
    "jenis": "arduino",
    "kd_detail": "101",
    "kd_barang": "BR0004",
    "kd_kondisi": "03",
    "jumlah": "1"
}
```

Delete Before didatabase

This screenshot shows the 'before didatabase' state of the database. The left pane shows the database structure with the 'inventori' schema selected. The 'detail' table is visible under 'inventori'. The right pane shows the contents of the 'detail' table:

| | kd_detail | kd_barang | kd_kondisi | jumlah |
|--------------------------|-----------|-----------|------------|--------|
| <input type="checkbox"/> | Edit | Copy | Delete | 1 |
| <input type="checkbox"/> | Edit | Copy | Delete | 100 |

After dipostman

The screenshot shows two windows side-by-side. On the left is the phpMyAdmin interface, displaying a database structure with tables like 'barang', 'detail', 'jenis_barang', 'kondisi', 'peminjaman', 'pengembalian', and 'user'. A specific row from the 'detail' table is selected, showing fields: kd_detail (100), kd_barang (BR0002), kd_kondisi (03), and jumlah (2). On the right is the Postman application, showing a DELETE request to 'http://localhost/ci-api/index.php/api/detail/users'. The request body contains four parameters: kd_detail (1), kd_barang (BR0001), kd_kondisi (02), and jumlah (1). The 'Body' tab is selected in Postman.

➤ Table barang

Scriptnya :

```
<?php
```

```
defined('BASEPATH') or exit('No direct script access allowed');
```

```
// This can be removed if you use __autoload() in config.php OR use Modular Extensions
/** @noinspection PhpIncludeInspection */
require APPPATH . '/libraries/REST_Controller';
```

```
// use namespace
```

```
use Restserver\Libraries\REST_Controller;
```

```
/**
```

```
* This is an example of a few basic user interaction methods you could use
```

```
* all done with a hardcoded array
```

```
*
```

```
* @package    CodeIgniter
```

```
* @subpackage Rest Server
```

```
* @category   Controller
```

```
* @author    Phil Sturgeon, Chris Kacerguis
```

```
* @license    MIT
```

```
* @link      https://github.com/chriskacerguis/codeigniter-restserver
```

```
*/
```

```
class barang extends REST_Controller
```

```

{

function __construct()
{
    // Construct the parent class
    parent::__construct();

    // Configure limits on our controller methods
    // Ensure you have created the 'limits' table and enabled 'limits' within
    application/config/rest.php
    $this->methods['users_get']['limit'] = 500; // 500 requests per hour per user/key
    $this->methods['users_post']['limit'] = 100; // 100 requests per hour per user/key
    $this->methods['users_delete']['limit'] = 50; // 50 requests per hour per user/key
}

public function users_get()
{
    // Users from a data store e.g. database
    $users = [
        // ['id' => 1, 'name' => 'John', 'email' => 'john@example.com', 'fact' => 'Loves
        coding'],
        // ['id' => 2, 'name' => 'Jim', 'email' => 'jim@example.com', 'fact' => 'Developed
        on CodeIgniter'],
        // ['id' => 3, 'name' => 'Jane', 'email' => 'jane@example.com', 'fact' => 'Lives in the
        USA', ['hobbies' => ['guitar', 'cycling']]],
    ];
}

$id = $this->get('kd_barang');

// If the id parameter doesn't exist return all the users

if ($id === NULL) {
    $users = $this->db->get("barang")->result_array();
    // Check if the users data store contains users (in case the database result returns
    NULL)
    if ($users) {
        // Set the response and exit
        $this->response($users, REST_Controller::HTTP_OK); // OK (200) being the
        HTTP response code
    } else {
        // Set the response and exit
    }
}

```

```

$this->response([
    'status' => FALSE,
    'message' => 'No users were found'
], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
response code
}

}

// Find and return a single record for a particular user.
else {
    $id = (int) $id;

    // Validate the id.
    if ($id <= 0) {
        // Invalid id, set the response and exit.
        $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); //
BAD_REQUEST (400) being the HTTP response code
    }

    // Get the user from the array, using the id as key for retrieval.
    // Usually a model is to be used for this.

    $user = NULL;

    if (!empty($users)) {
        foreach ($users as $key => $value) {
            if (isset($value['kd_barang']) && $value['kd_barang'] === $id) {
                $user = $value;
            }
        }
    }
}

if (!empty($user)) {
    $this->set_response($user, REST_Controller::HTTP_OK); // OK (200) being the
HTTP response code
} else {
    $this->set_response([
        'status' => FALSE,
        'message' => 'User could not be found'
], REST_Controller::HTTP_NOT_FOUND); // NOT_FOUND (404) being the HTTP
response code
}

```

```

        }
    }
}

public function users_post()
{
    // $this->some_model->update_user( ... );
    $message = [
        'kd_barang' => $this->post('kd_barang'),
        'barang' => $this->post('barang'),
        'kd_jenis' => $this->post('kd_jenis'),
    ];

    $this->db->insert("barang", $message);
    $this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
    (201) being the HTTP response code
    $this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
    (201) being the HTTP response code
}

function users_put()
{
    $id = $this->put('kd_barang');
    $message = [
        'kd_barang' => $this->put('kd_barang'),
        'barang' => $this->put('barang'),
        'kd_jenis' => $this->put('kd_jenis'),
    ];
    $this->db->where('kd_barang', $id);
    $update = $this->db->update('barang', $message);
    if ($update) {
        $this->response($message, 200);
    } else {
        $this->response(array('status' => 'fail', 502));
    }
}

public function users_delete()
{
    $id = $this->delete('kd_barang');
}

```

```

$this->db->where('kd_barang', $id);
$delete = $this->db->delete('barang');

// Validate the id.
if ($delete) {
    // Set the response and exit
    $this->response(NULL, REST_Controller::HTTP_BAD_REQUEST); // BAD_REQUEST
(400) being the HTTP response code
}

// $this->some_model->delete_something($id);
$message = [
    'kd_barang' => $id,
    'message' => 'Deleted the resource'
];

$this->set_response($message, REST_Controller::HTTP_NO_CONTENT); // NO_CONTENT (204) being the HTTP response code
}
}

```

Menampilkan database ke postman

The screenshot shows the Postman application interface. A GET request is made to `http://localhost/ci-api/index.php/api/barang/users`. The response status is 200 OK, and the response body is displayed in JSON format:

```

[
    {
        "kd_barang": "BR0002",
        "nama_barang": "Alkohol 70%",
        "kd_jenis": "KJ008"
    },
    {
        "kd_barang": "BR0003",
        "nama_barang": "Antena N Wifi omni indoor",
        "kd_jenis": "KJ005"
    },
    {
        "kd_barang": "BR0004",
        ...
    }
]

```

Menambahkan database dari postman

phpMyAdmin

localhost / 12 Membuat Lin (3) WhatsApp +

localhost/phpmyadmin/index.php?route=/sql... Home Workspaces API Network Explore

Server: 127.0.0.1 Database: inventori Table: barang

Information_schema inventori barang detail jenis_barang kondisi peminjaman pengembalian user inventory kontak mysql performance_schema phpmyadmin test

BR0100 Papan PCB KJ027
BR0101 Processor Amd 486 KJ004
BR0102 Processor Intel Pentium KJ004
BR0103 Pylox Nipon Paint KJ001
BR0104 Remote InFocus KJ023
WRH54G Wireless G-Home Router KJ013
BR0105 Home Router WRH54G
BR0106 Esp8266 Mod KJ021
BR0107 Esp8266 Mod With NodeMcu base KJ021
BR0108 Esp-WROOM 32 KJ021
Handycam Sony DCR SX22 zoom 70X KJ025
BR0109 ProLink 8 Port PSW810 KJ019
BR0110 USB KJ001

POST http://localhost/ci-api/index.php/api/barang/users

Params Authorization Headers (12) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

kd_barang BR0111
nama_barang USB
kd_jenis KJ001

Key Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```

1 "kd_barang": "BR0111",
2 "nama_barang": "USB",
3 "kd_jenis": "KJ001"
4
5

```

Online Find and Replace Console

Update

localhost / 12 Membuat Lin (3) WhatsApp +

localhost/phpmyadmin/index.php?route=/sql... Home Workspaces API Network Explore

Server: 127.0.0.1 Database: inventori Table: barang

Information_schema inventori barang detail jenis_barang kondisi peminjaman pengembalian user inventory kontak mysql performance_schema phpmyadmin test

BR0100 Papan PCB KJ027
BR0101 Processor Amd 486 KJ004
BR0102 Processor Intel Pentium KJ004
BR0103 Pylox Nipon Paint KJ001
BR0104 Remote InFocus KJ023
WRH54G Wireless G-Home Router KJ013
BR0105 Home Router WRH54G
BR0106 Esp8266 Mod KJ021
BR0107 Esp8266 Mod With NodeMcu base KJ021
BR0108 Esp-WROOM 32 KJ021
Handycam Sony DCR SX22 zoom 70X KJ025
BR0109 ProLink 8 Port PSW810 KJ019
BR0110 USB KJ001
BR0111 arduino KJ001

Query results operations

Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query

Label: Let every user access this bookmark

POST http://localhost/ci-api/index.php/api/barang/users

PUT http://localhost/ci-api/index.php/api/barang/users

Params Authorization Headers (12) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

kd_barang BR0111
nama_barang arduino
kd_jenis KJ001

Key Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```

1 "kd_barang": "BR0111",
2 "nama_barang": "arduino",
3 "kd_jenis": "KJ001"
4
5

```

Online Find and Replace Console

Delete Before didatabase

localhost / 127.0.0.1 / inventon / localhost/ci-api/index.php/api/p... +

localhost/phpmyadmin/index.php?route=/sql&server=1&db=inventon&table=barang&pos=0

phpMyAdmin

Recent : Favorites

Server: 127.0.0.1 > Database: inventon > Table: barang

Browse Structure SQL Search Insert Export Import Privileges Operations Tracking Triggers

Showing rows 0 - 24 (110 total, Query took 0.0009 seconds.)

SELECT * FROM `barang`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Options kd_barang nama_barang kd_jenis

| | kd_barang | nama_barang | kd_jenis |
|--------------------------|-----------|----------------------------------|----------|
| <input type="checkbox"/> | BR0001 | Acrylic Epoxy Spray Paint Silver | KJ001 |
| <input type="checkbox"/> | BR0002 | Alkohol 70% | KJ008 |
| <input type="checkbox"/> | BR0003 | Antena N Wifi omni indoor | KJ005 |
| <input type="checkbox"/> | BR0004 | Antena Tp-Link Wifi Omni Indoor | KJ005 |
| <input type="checkbox"/> | BR0005 | Artic Spary Paint Black | KJ001 |
| <input type="checkbox"/> | BR0006 | Baterai AA ABC | KJ003 |
| <input type="checkbox"/> | BR0007 | Baterai AA Alkaline | KJ003 |
| <input type="checkbox"/> | BR0008 | Baterai AAA Eveready | KJ003 |
| <input type="checkbox"/> | BR0009 | Baterai CMOS | KJ003 |
| <input type="checkbox"/> | BR0010 | Baterai Eveready 9 volt | KJ003 |
| <input type="checkbox"/> | BR0011 | Baterai HH-Watt 9 volt | KJ003 |
| <input type="checkbox"/> | BR0012 | Baterai Laptop TOSHIBA | KJ003 |
| <input type="checkbox"/> | BR0013 | Cable Tester | KJ015 |

Console

jenis_barang.sql peminjaman (1).sql peminjaman.sql

Show all

After dipostman

localhost / 127.0.0.1 / inventon / localhost/ci-api/index.php... +

localhost/phpmyadmin/index.php?route=/sql... Home Workspaces API Network Explore

http://localhost/ci-api/index.php/api/barang/users

DELETE http://localhost/ci-api/index.php/api/barang/users

Params Authorization Headers (8) Body Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

| KEY | VALUE |
|---|-------------------------|
| <input checked="" type="checkbox"/> kd_barang | BR0005 |
| <input checked="" type="checkbox"/> nama_barang | Artic Spary Paint Black |
| <input checked="" type="checkbox"/> kd_jenis | KJ001 |
| Key | Value |

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

1

Online Find and Replace Console