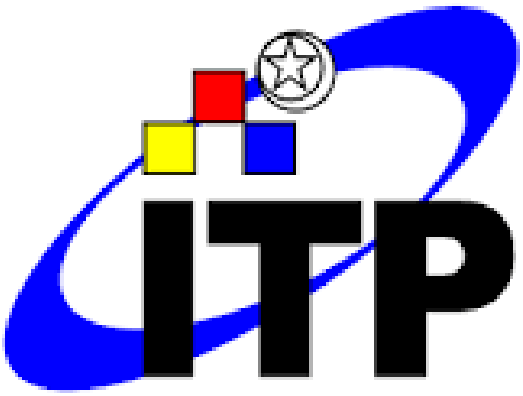
LAPORAN PEMOGRAMAN WEB



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
    }
```

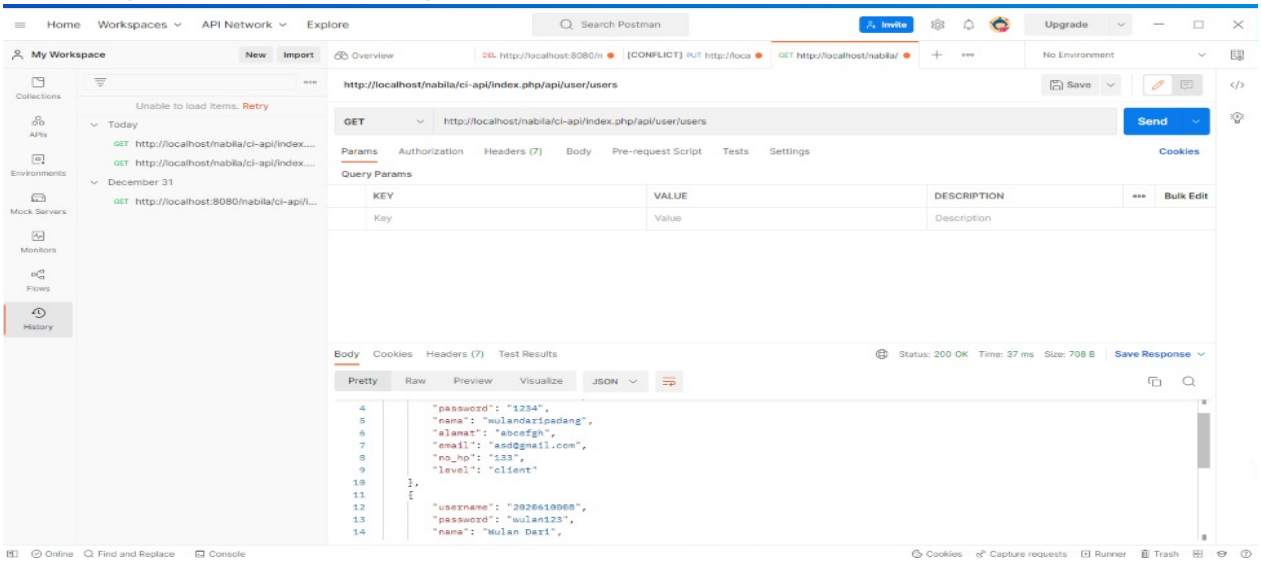
```
    public function users_get()
    {
        // Users from a data store e.g. database
        $users = [
            // ['id' => 1, 'name' => 'John', 'email' => 'john@example.com', 'fact' => 'Loves
coding'],
            // ['id' => 2, 'name' => 'Jim', 'email' => 'jim@example.com', 'fact' => 'Developed
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'),
```

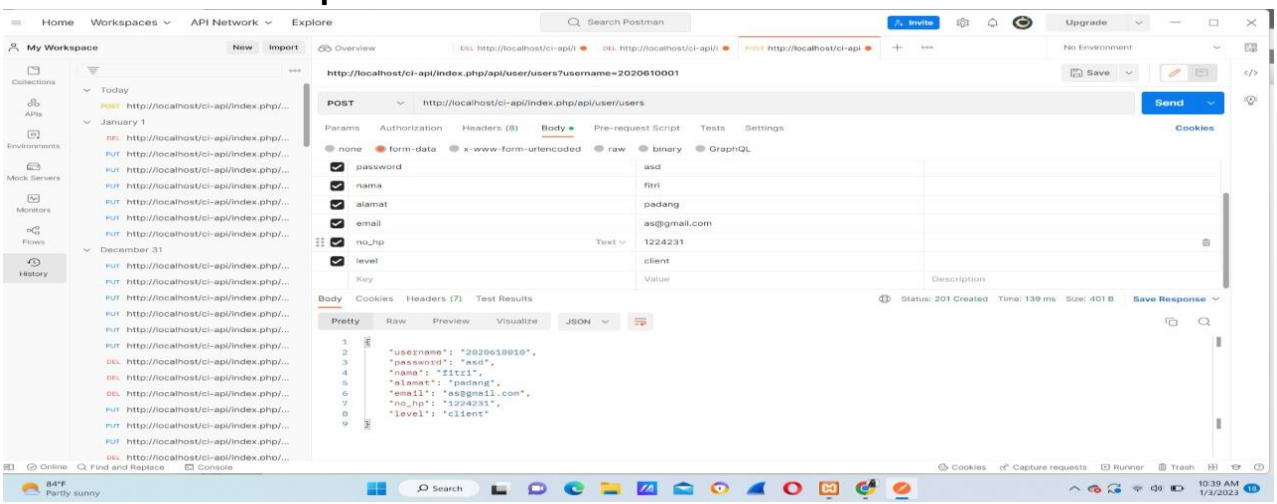
```
        ];
        $this->db->insert("user", $message);
        $this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
        $this->set_response($message, REST_Controller::HTTP_CREATED); // CREATED
(201) being the HTTP response code
    }
  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

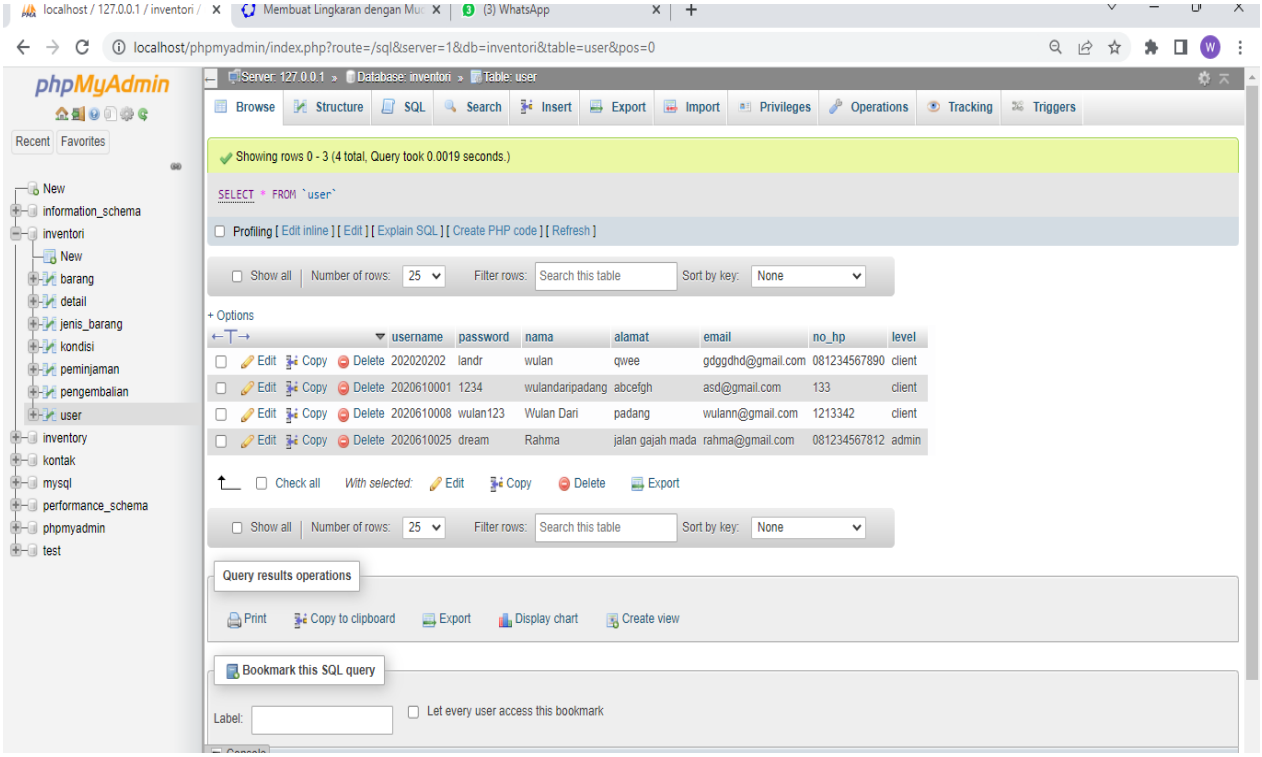
Menampilkan database ke postman



Menambahkan Dari postman ke database

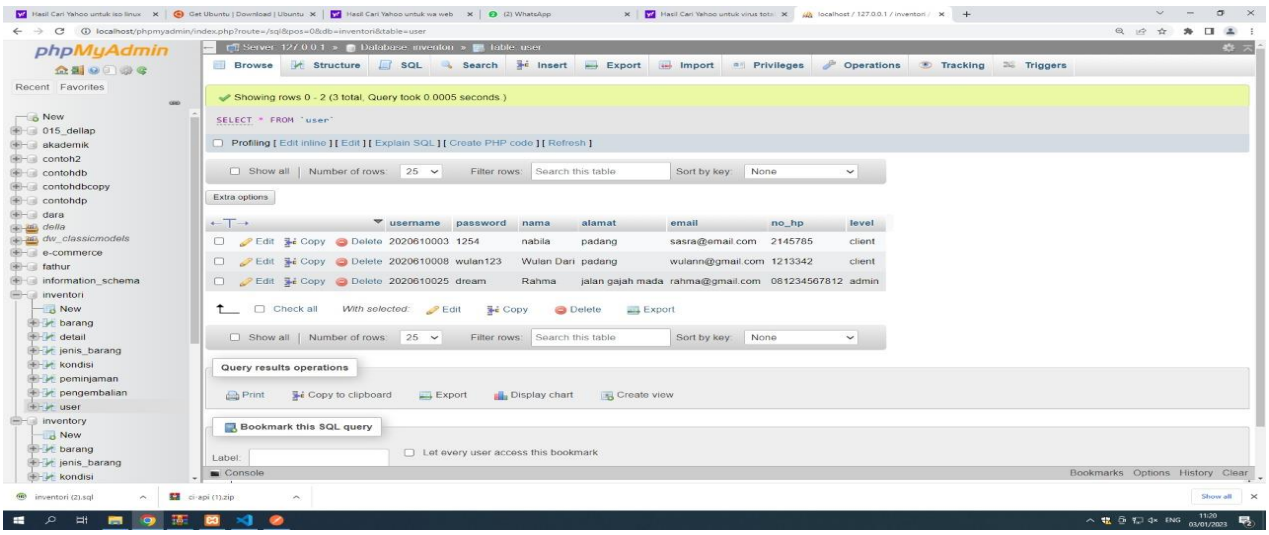


Update

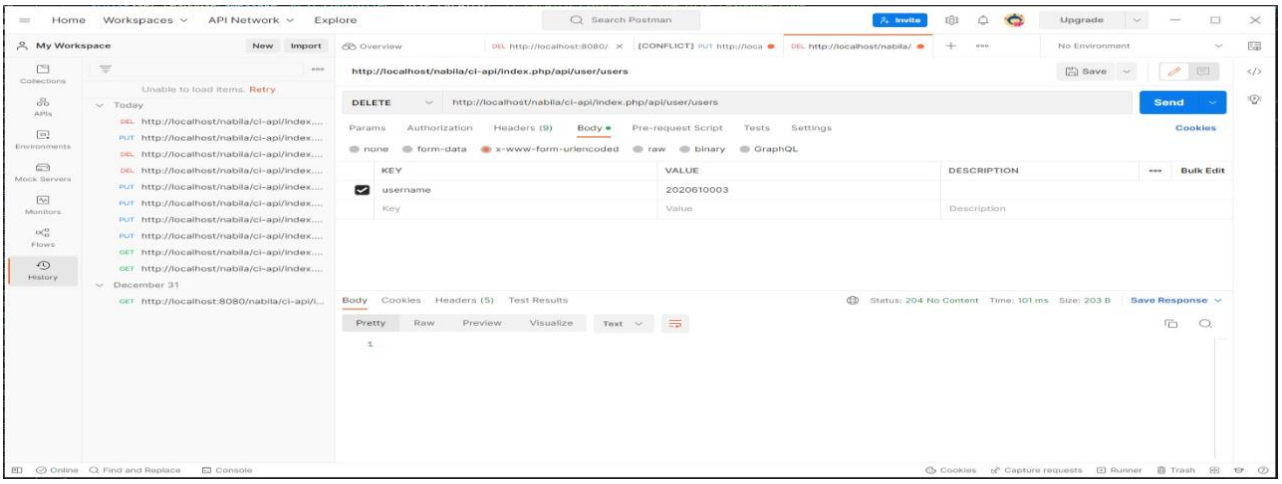


Delete

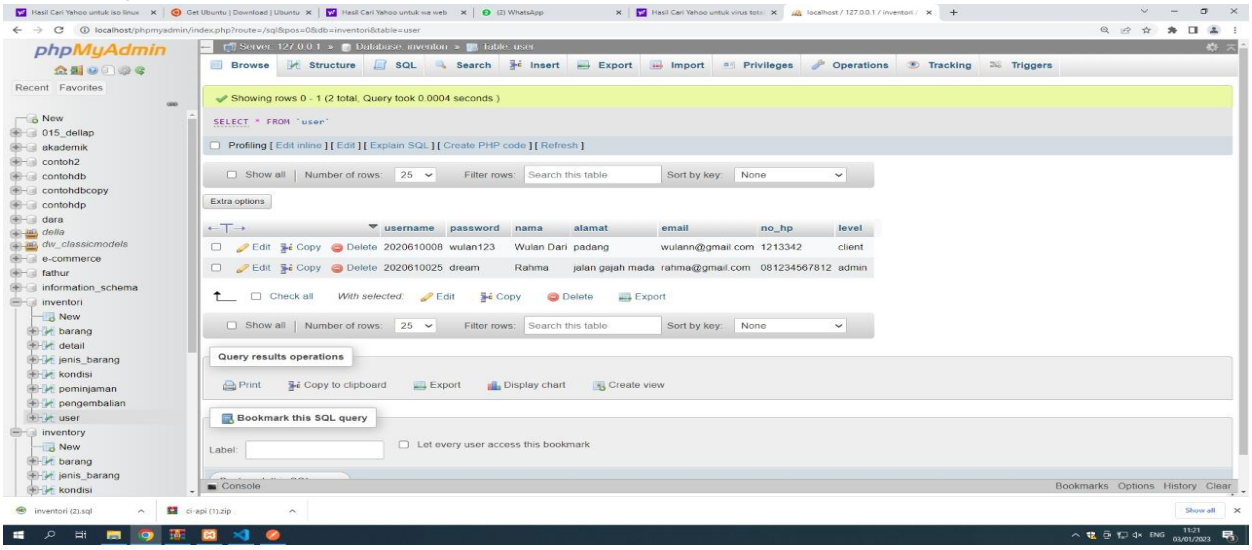
Before di database



Dari postman delete



After dipostman



- **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;
```

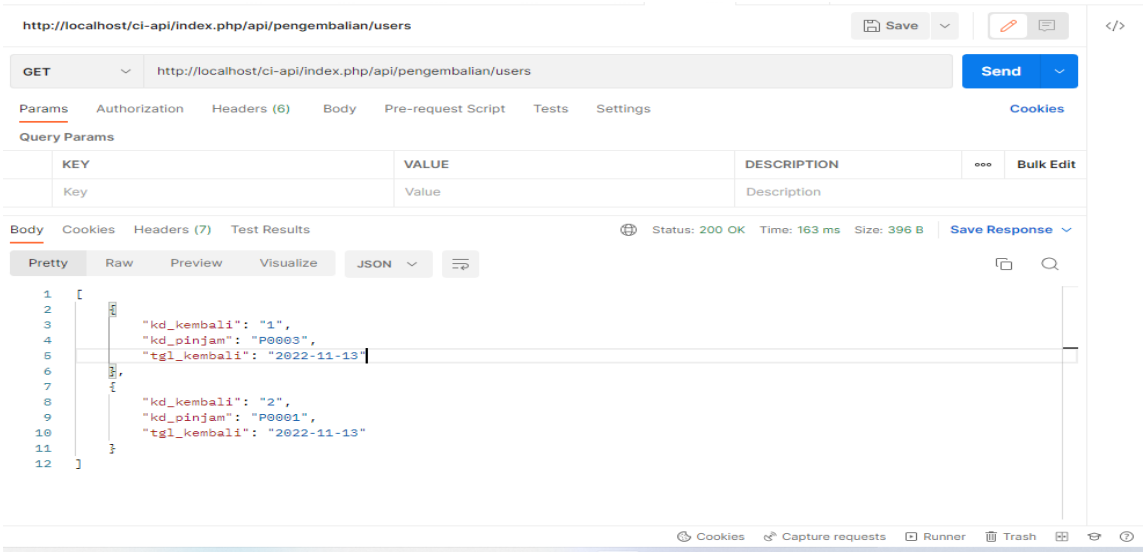
```
/**
 * This is an example of a few basic user interaction methods you could use
 * all done with a hardcoded array
 *
 * @package        CodeIgniter
 * @subpackage     Rest Server
 * @category       Controller
 * @author         Phil Sturgeon, Chris Kacerguis
 * @license        MIT
 * @link           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;
```

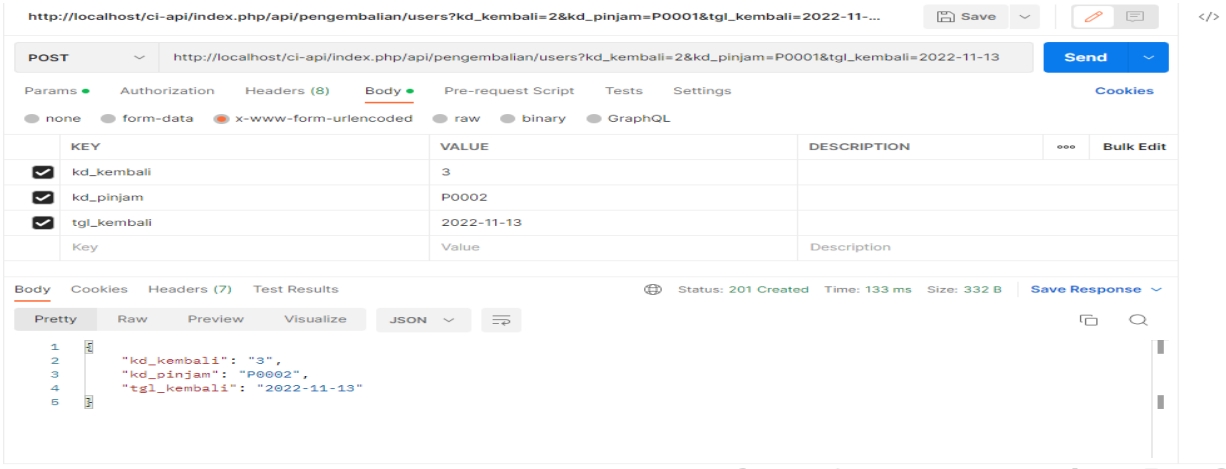
```
            }
        }
    }

    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'),
```

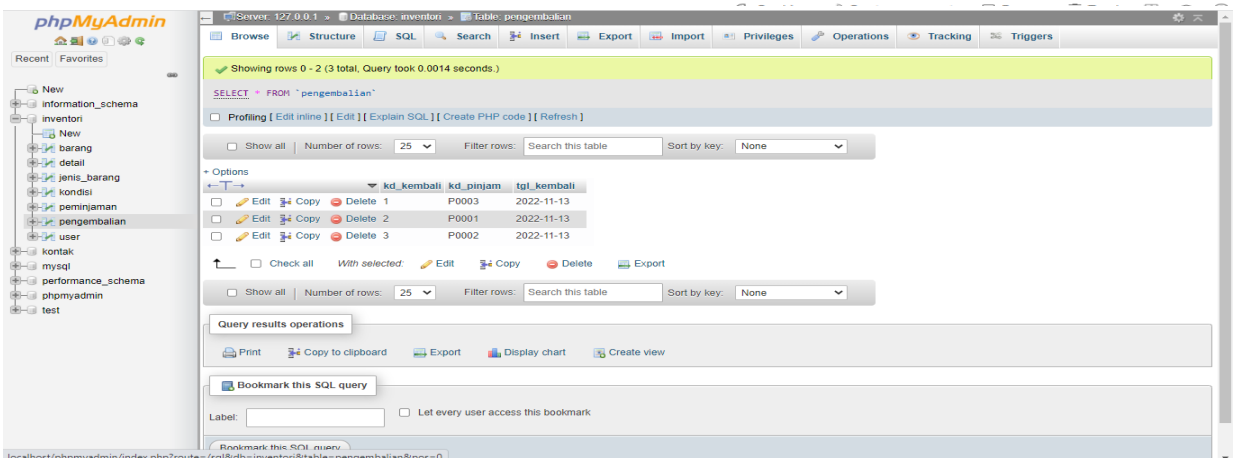
```
        ];
        $this->db->where('kd_kembali', $id);
        $update = $this->db->update('pengembalian', $message);
        if ($update) {
            $this->response($message,200);
        } else {
            $this->response(array('status' => 'fail', 502));
        }
    }

    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

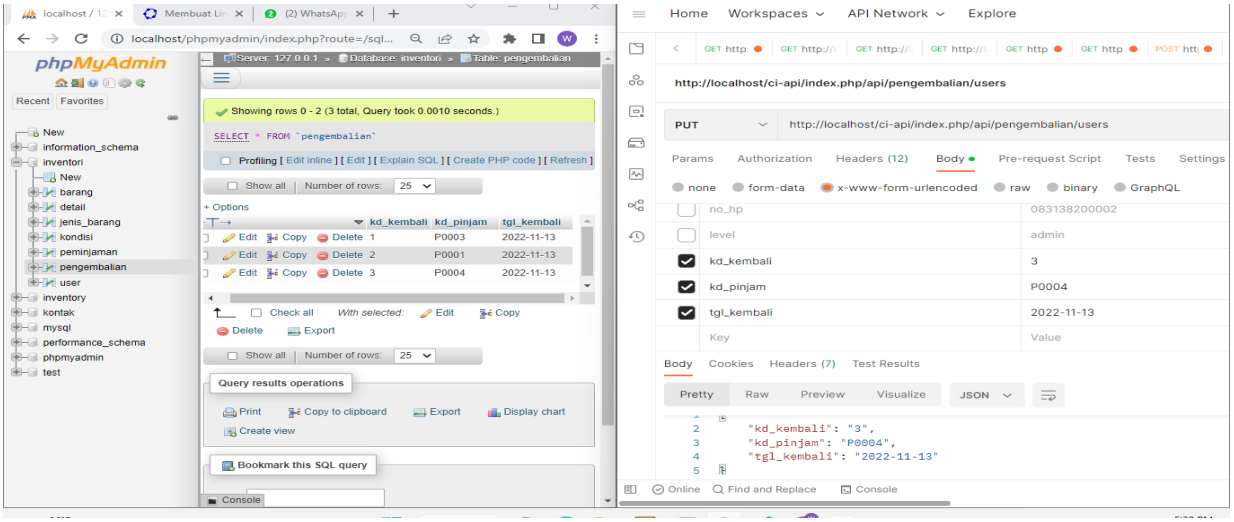


Menambahkan database dari postman



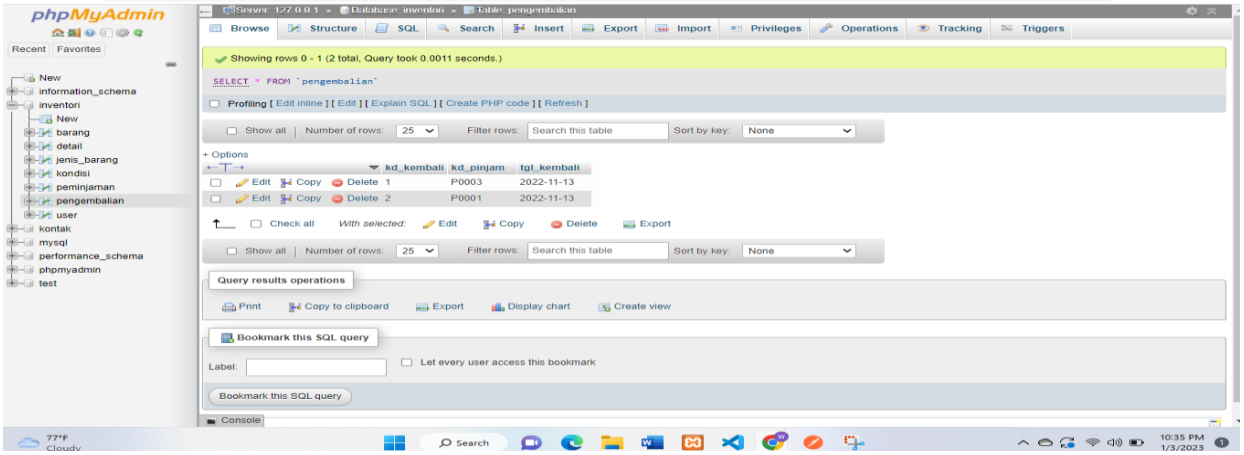


Update

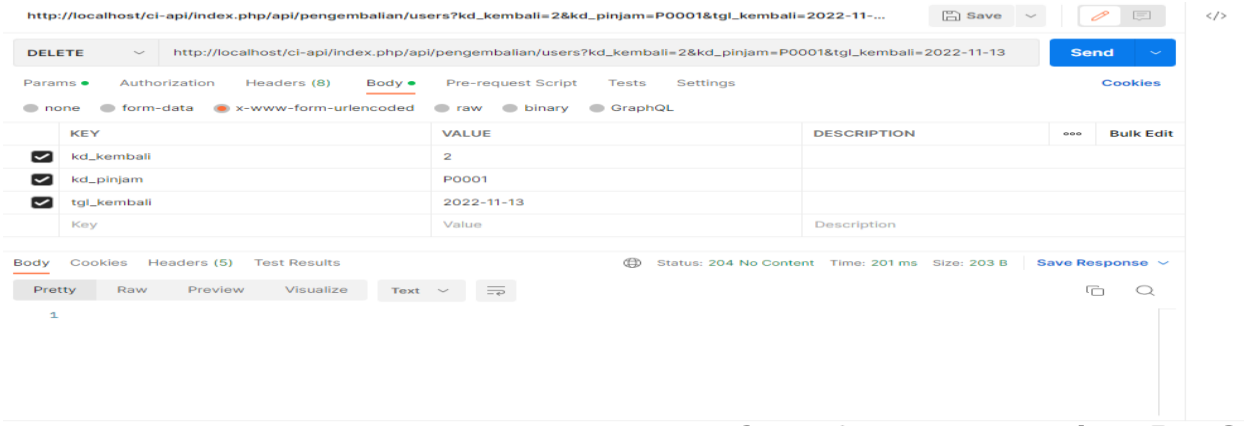


Delete

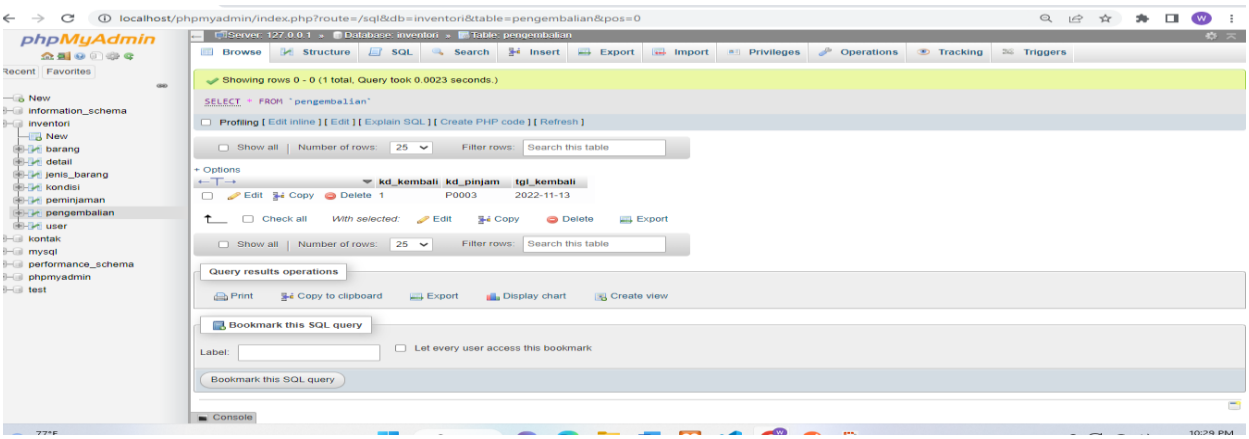
Before di database



Dari postman



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()
    {
```

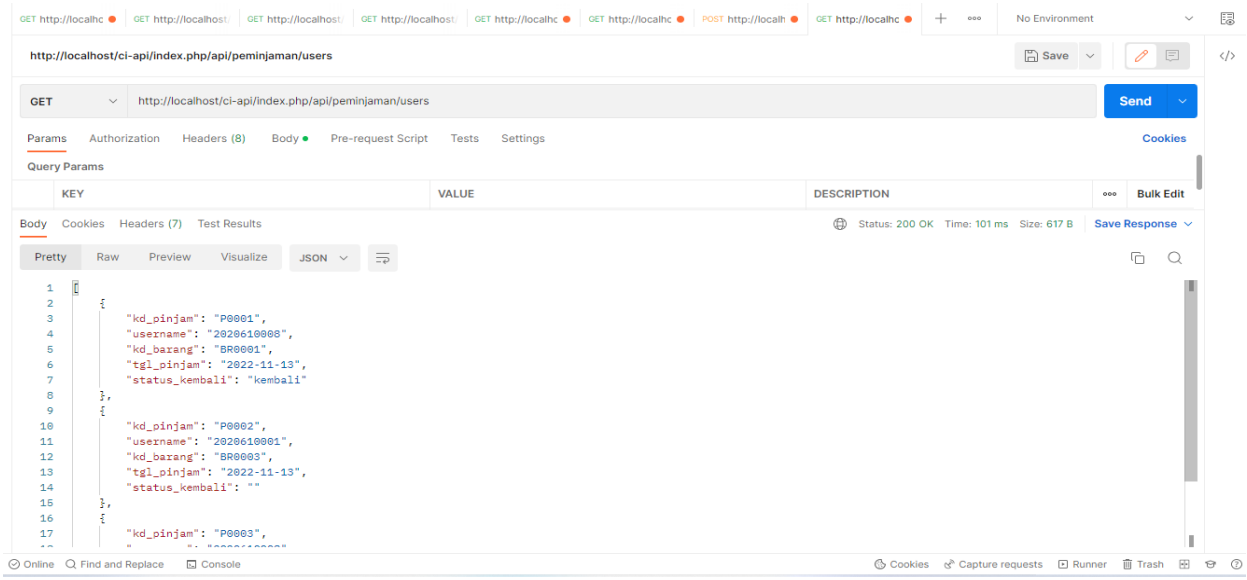
```
        // Construct the parent class
        parent::__construct();

        // 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
        }
    }
}
```

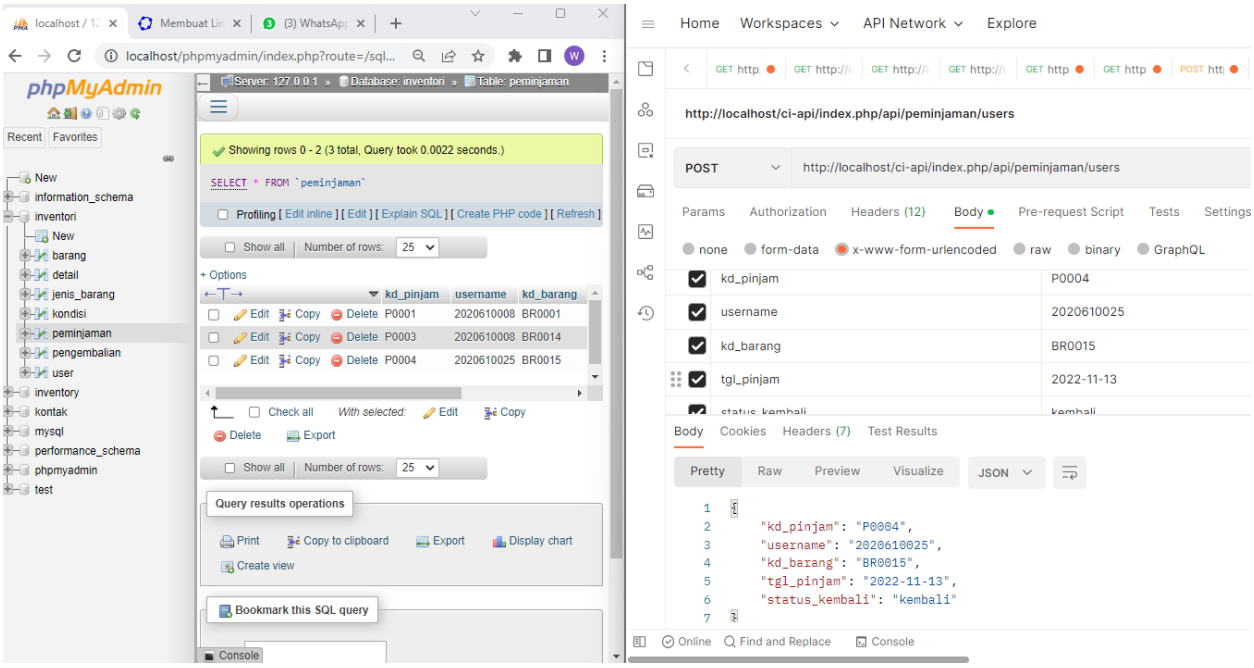
```
  public function users_post()
  {
    // $this->some_model->update_user( ... );
    $message = [
      'kd_pinjam' => $this->post('kd_pinjam'),
      'username' => $this->post('username'),
      'kd_barang' => $this->post('kd_barang'),
      'tgl pinjam' => $this->post('tgl pinjam'),
      'status_kembali'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

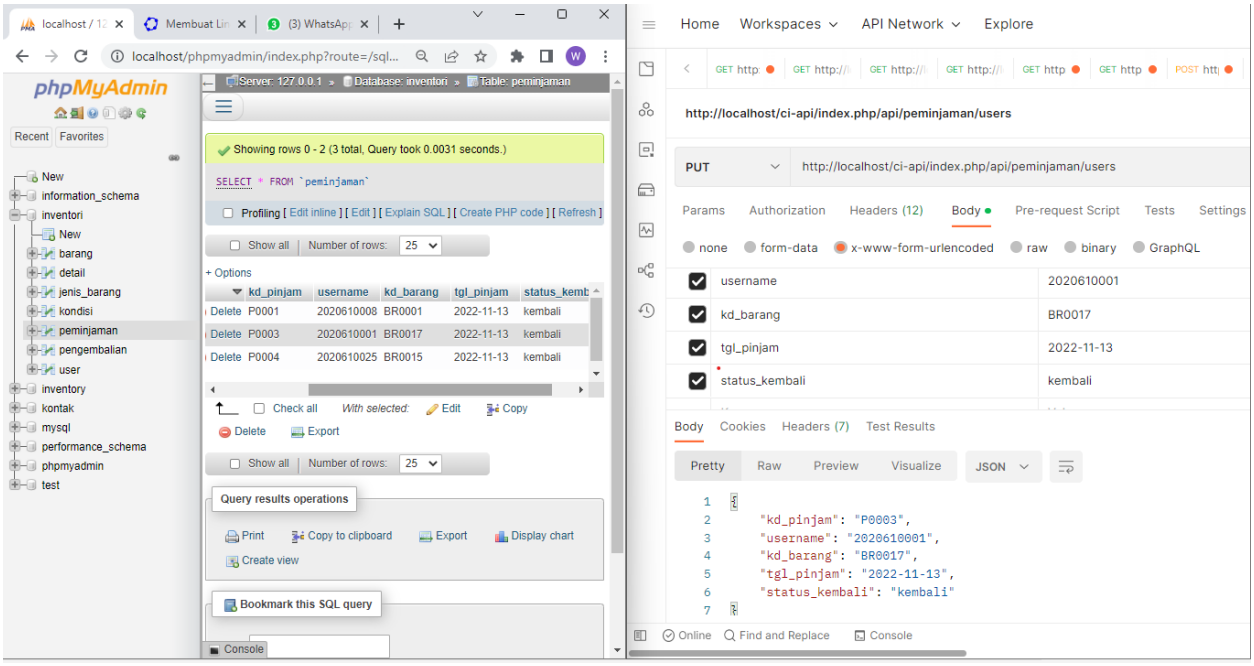
Menampilkan database ke postman



Menambahkan database dari postman

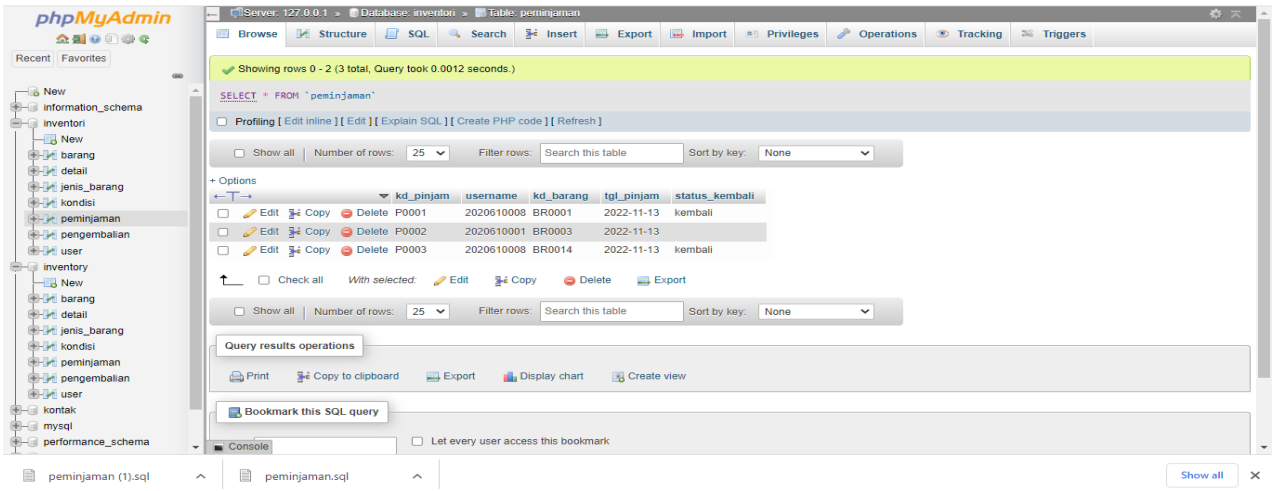


Update

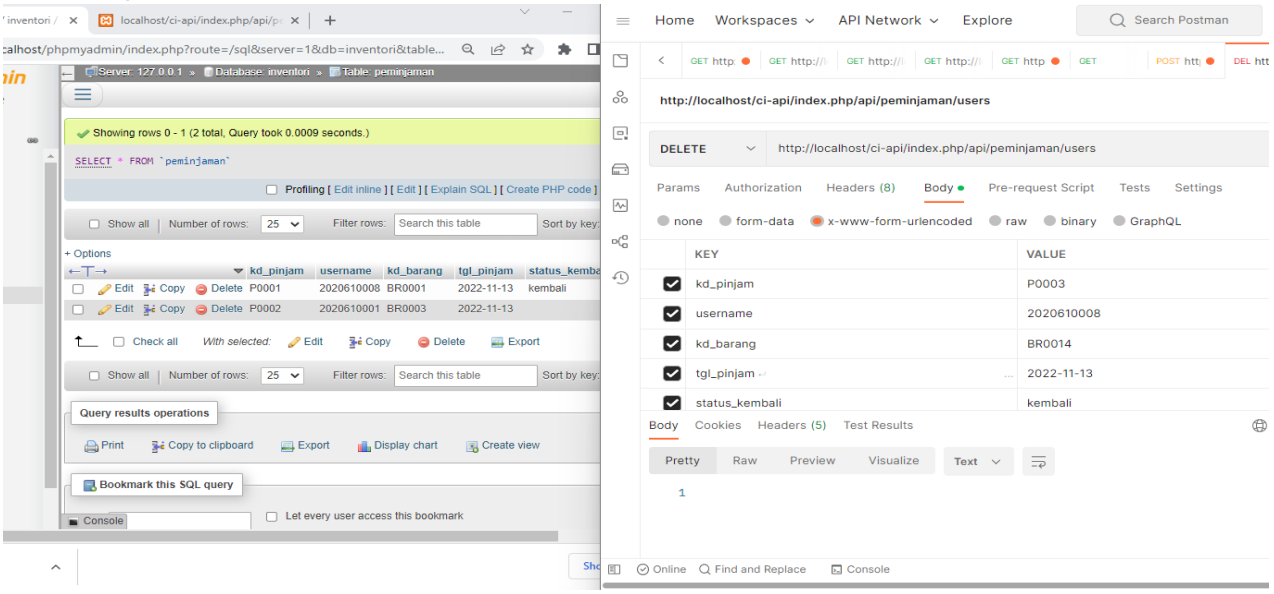


Delete

Before di database



After dipostman



- **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;
```

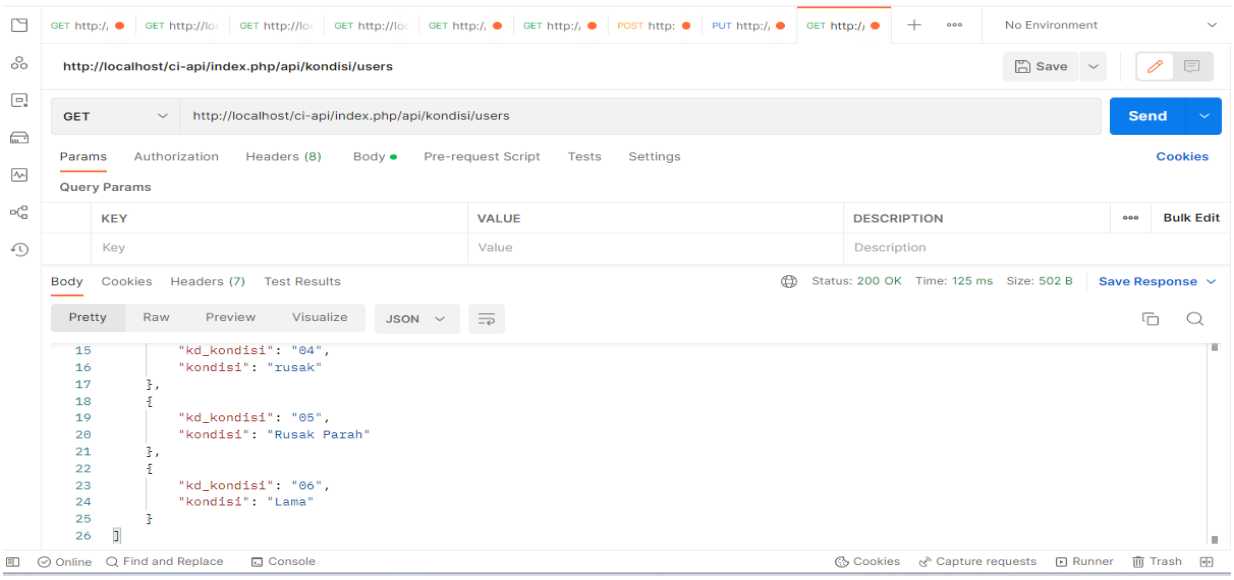
```
/**
 * This is an example of a few basic user interaction methods you could use
 * all done with a hardcoded array
 *
 * @package        CodeIgniter
 * @subpackage     Rest Server
 * @category       Controller
 * @author         Phil Sturgeon, Chris Kacerguis
 * @license        MIT
 * @link           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;
```

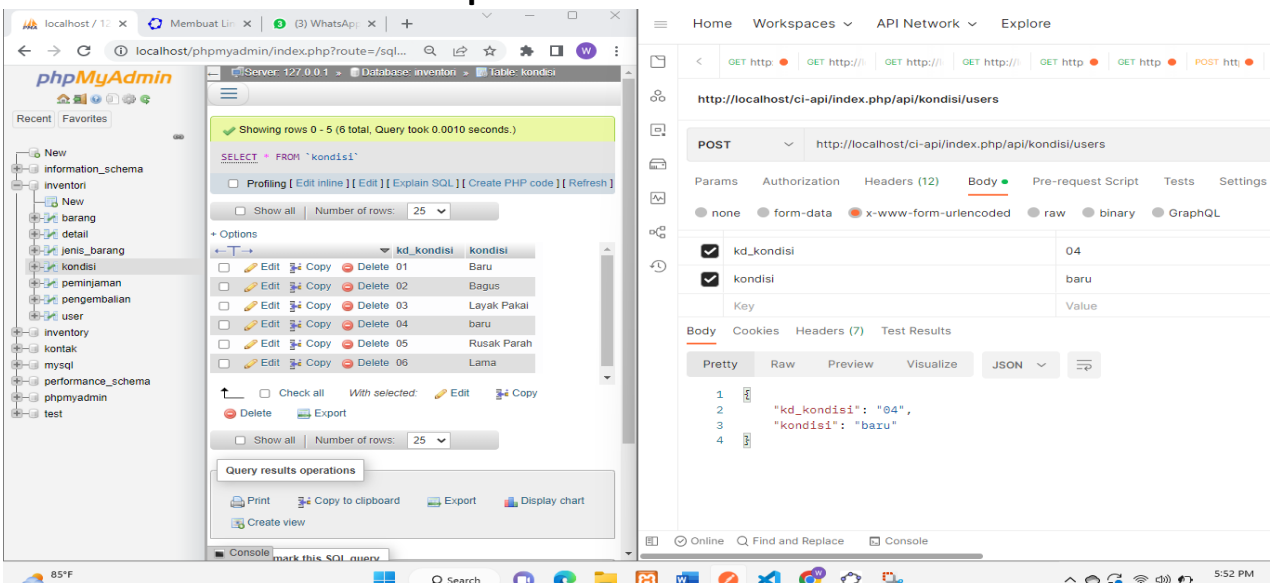
```
            }
        }
    }

    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
    }
}
```

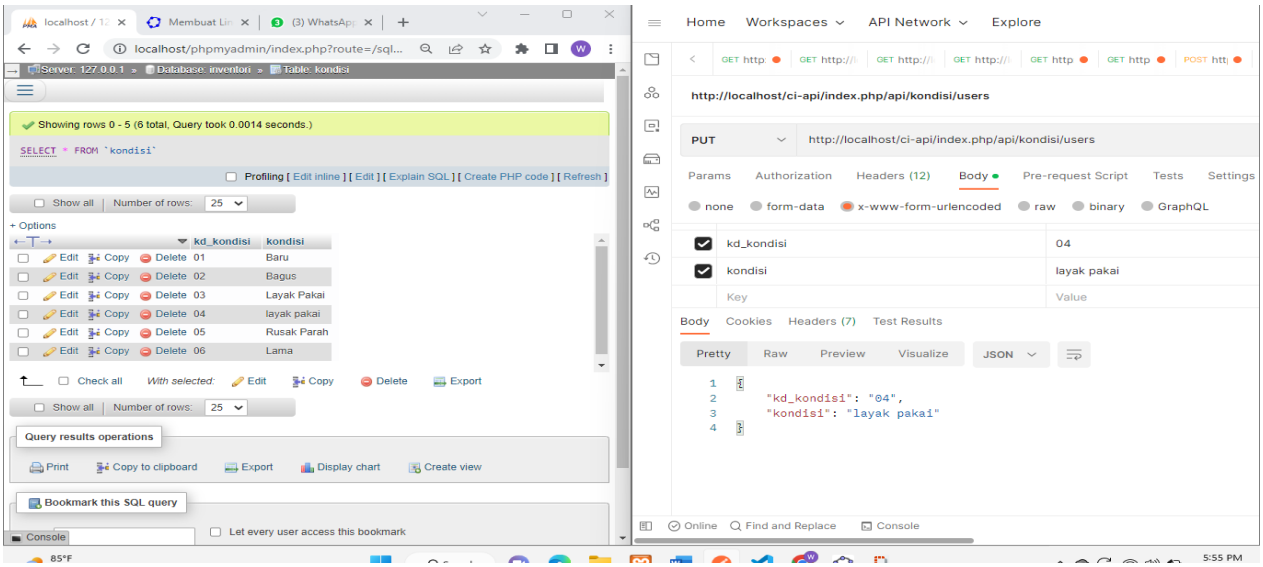
Menampilkan database ke postman



Menambahkan database dari postman

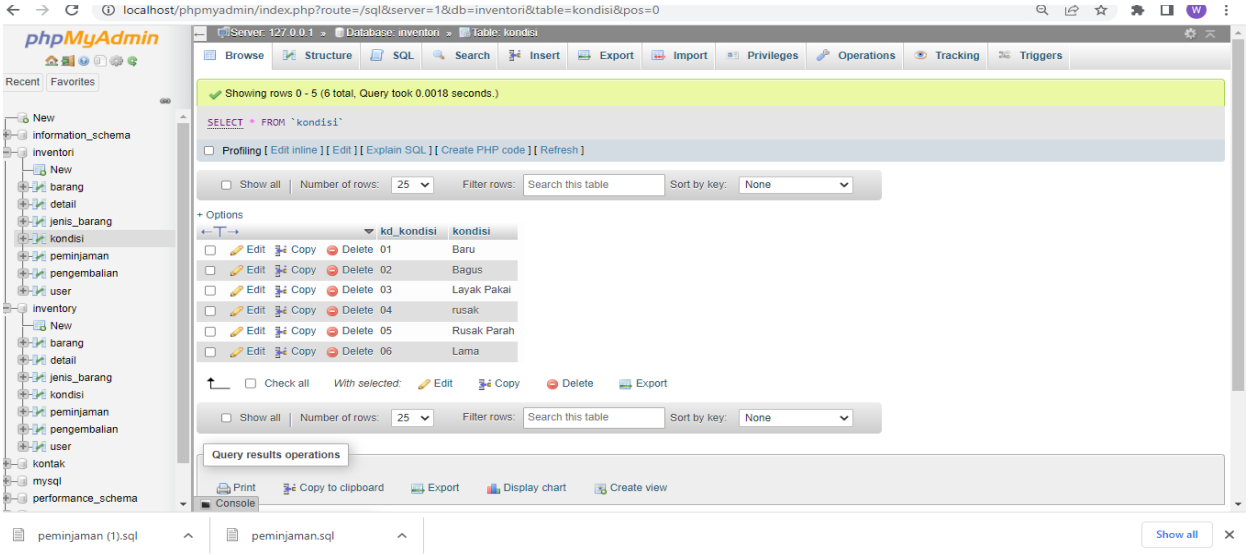


Update

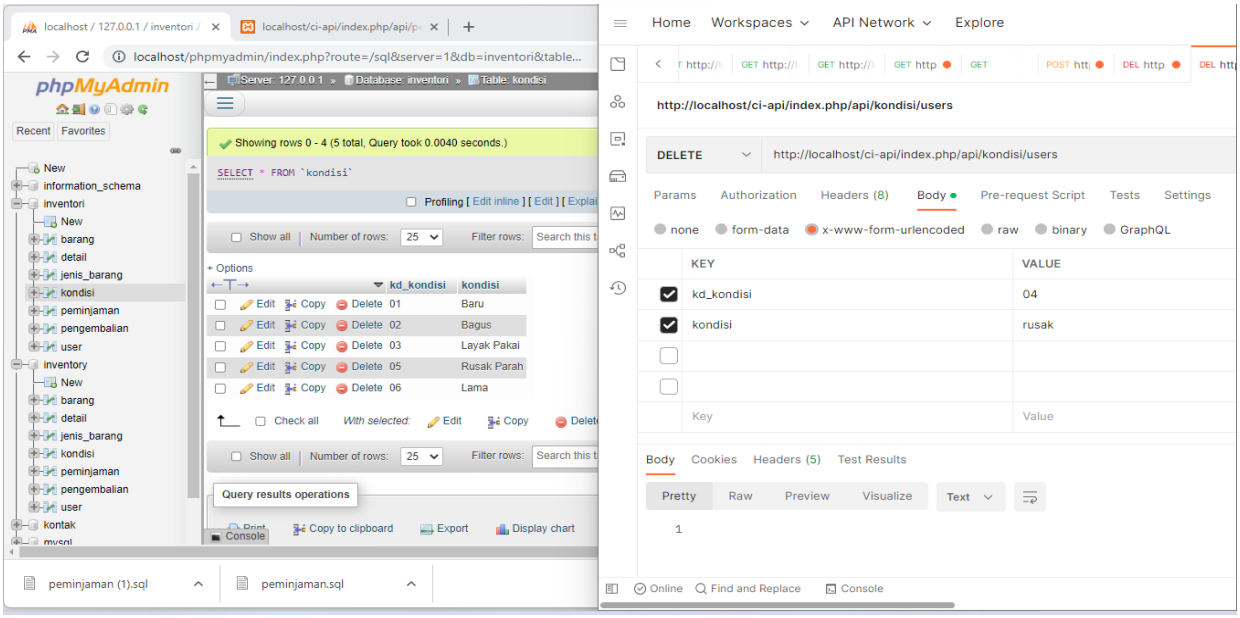


Delete

Before didatabase



After dipostman



➢ 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.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 jenis_barang extends REST_Controller
```

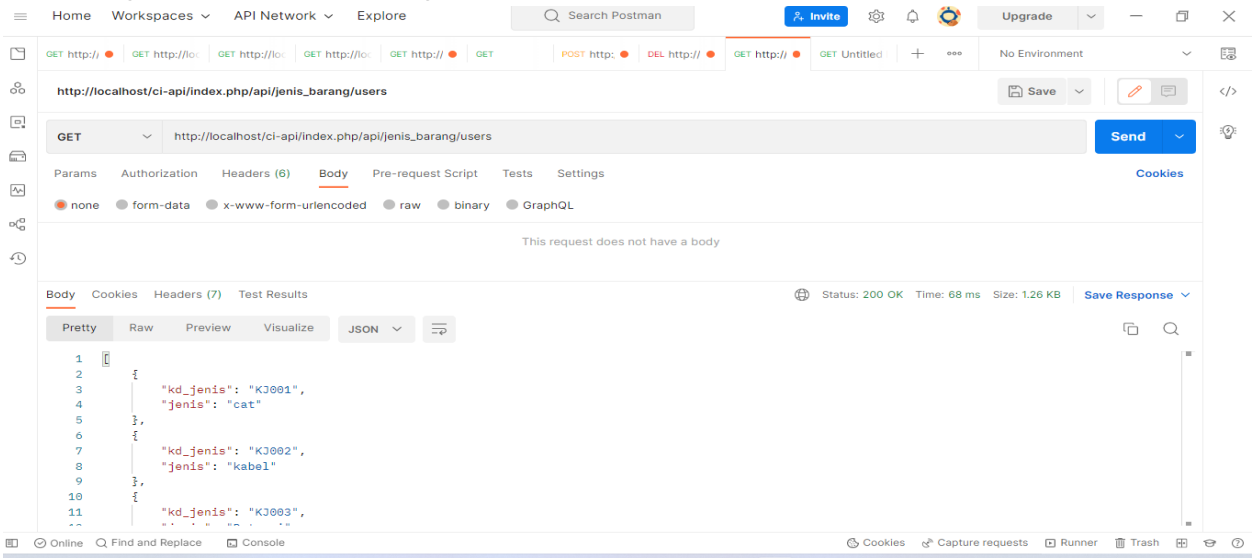
```
{

    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
```

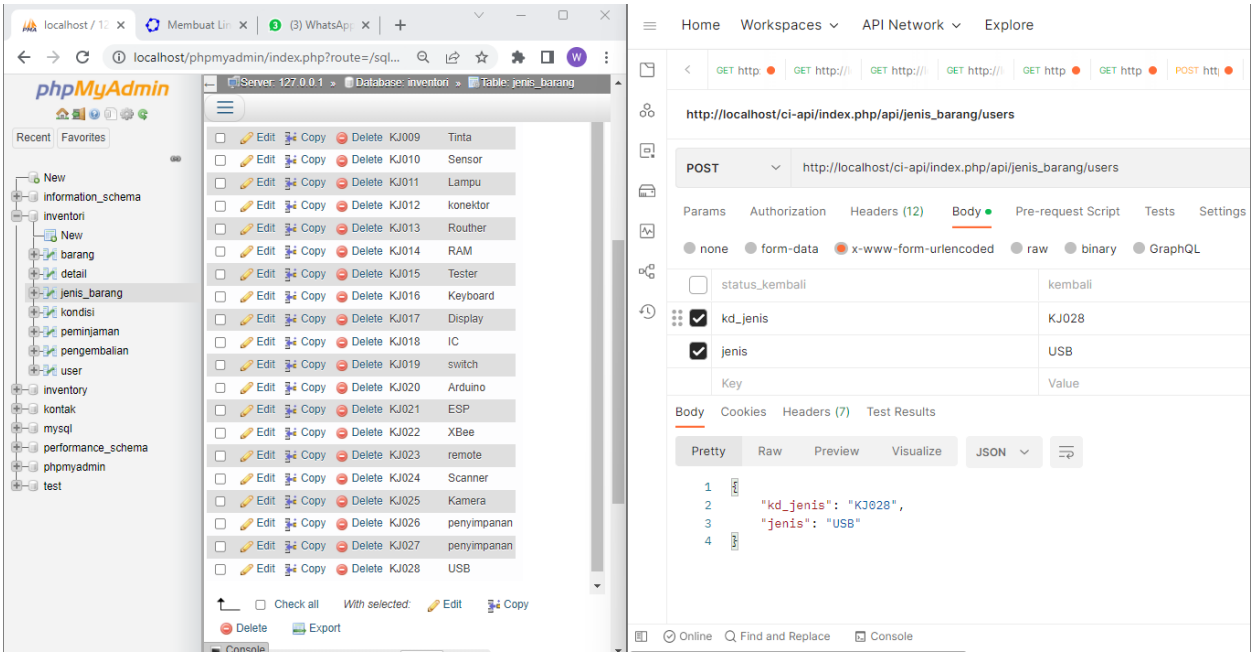
```
        }
    }
}

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
    }
}
```

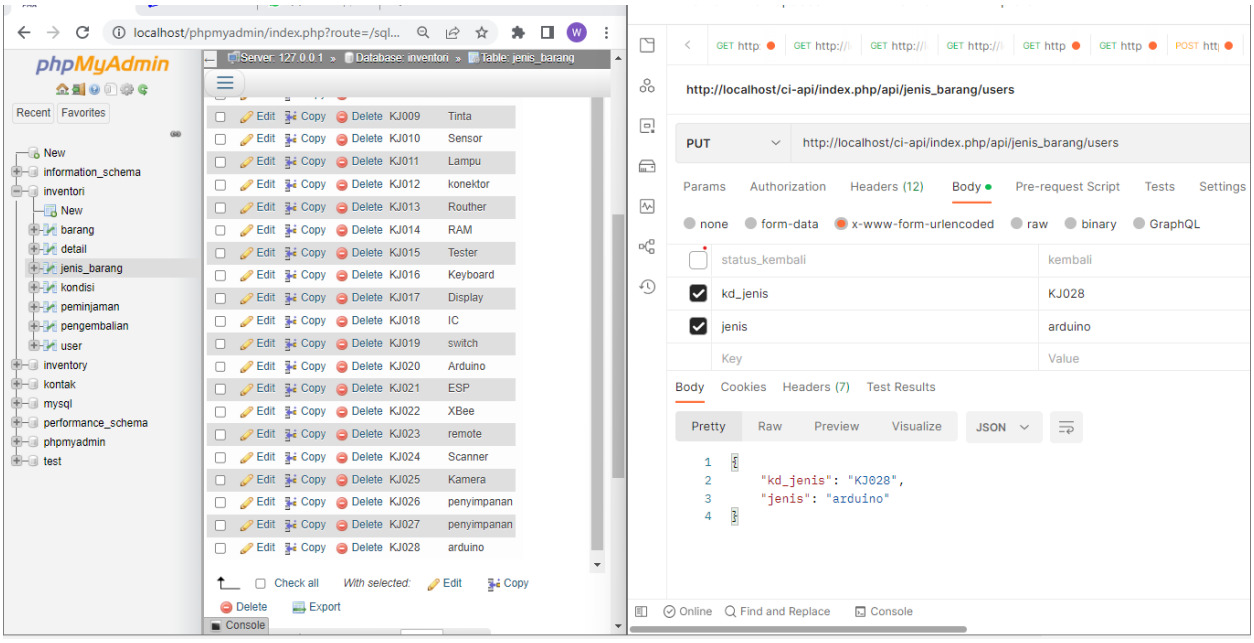
Menampilkan database ke postman



Menambahkan database dari postman

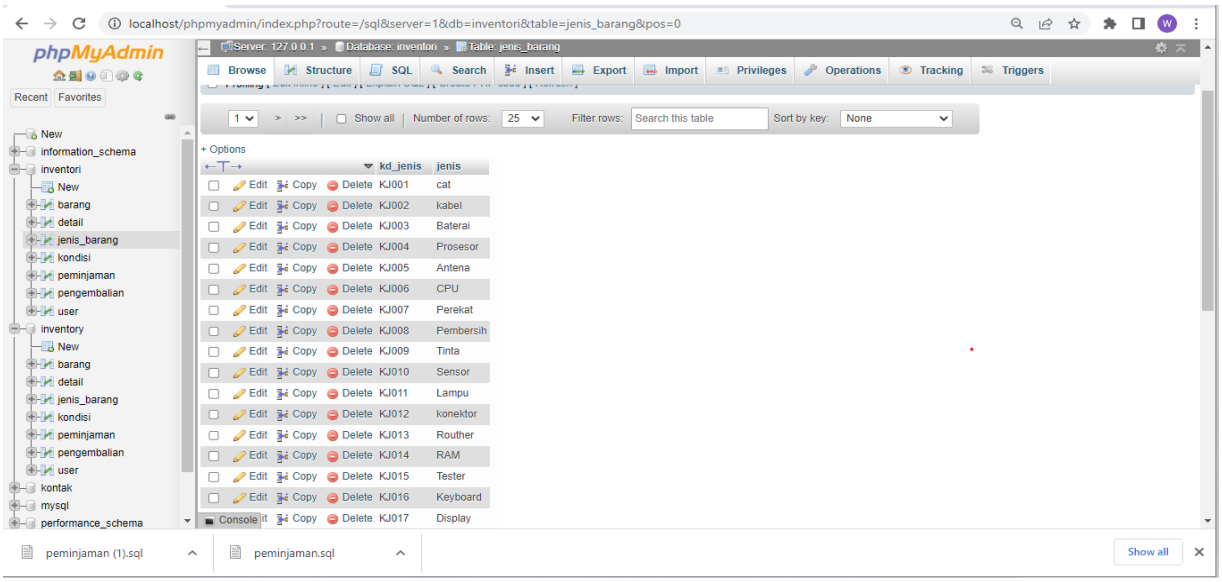


Update

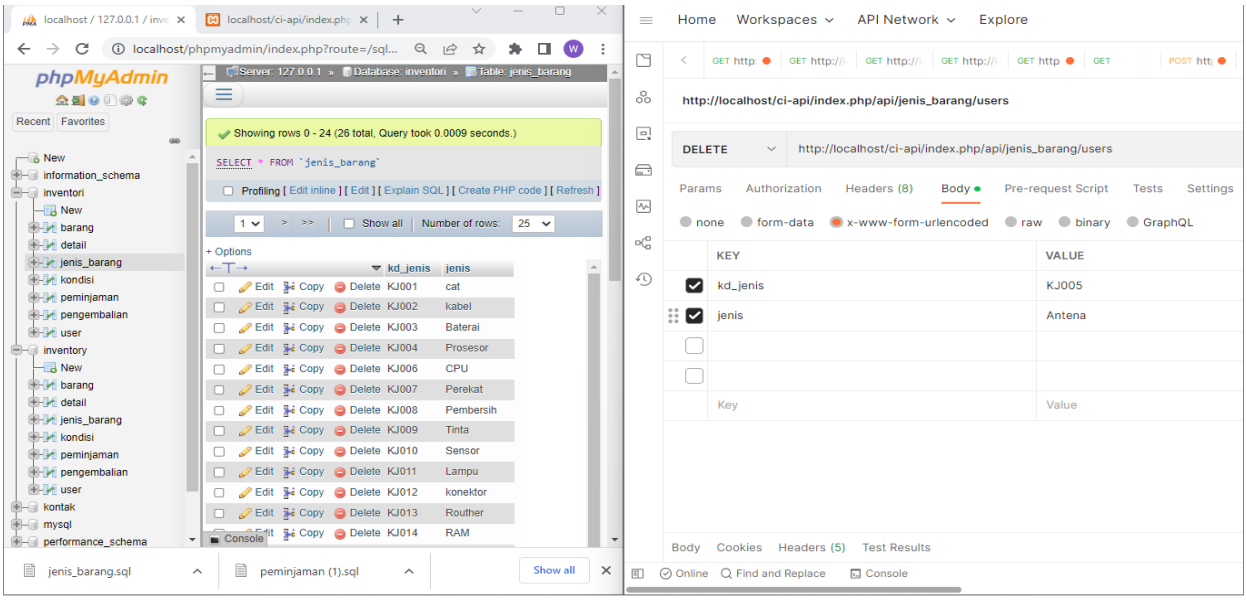


Delete

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';
```

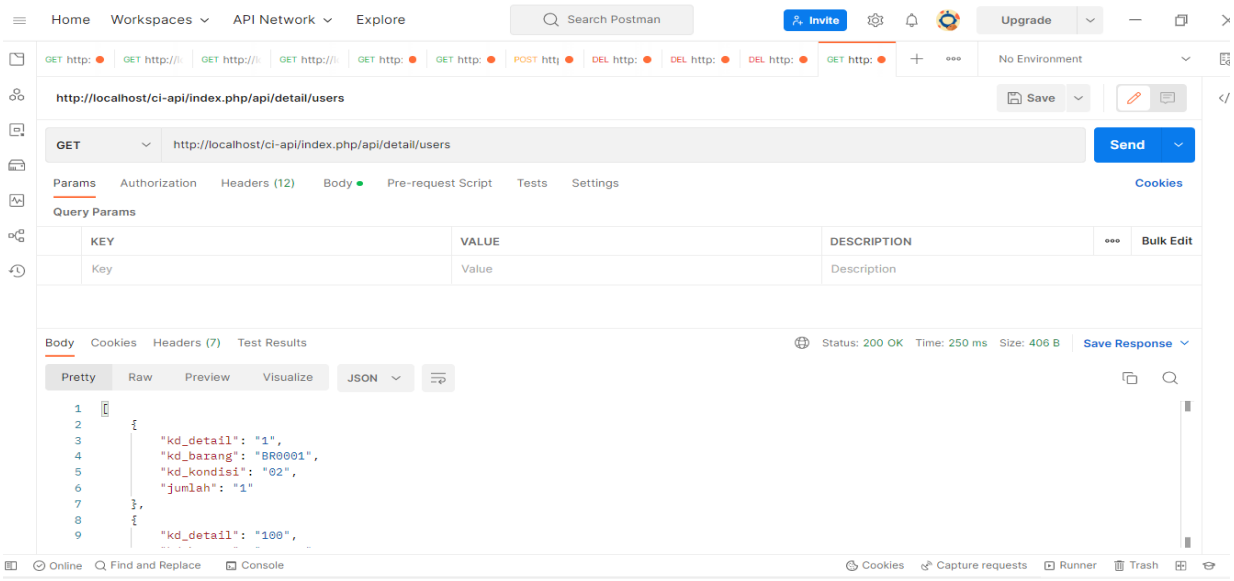
```
// use namespace
use Restserver\Libraries\REST_Controller;

/**
 * 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)) {
```

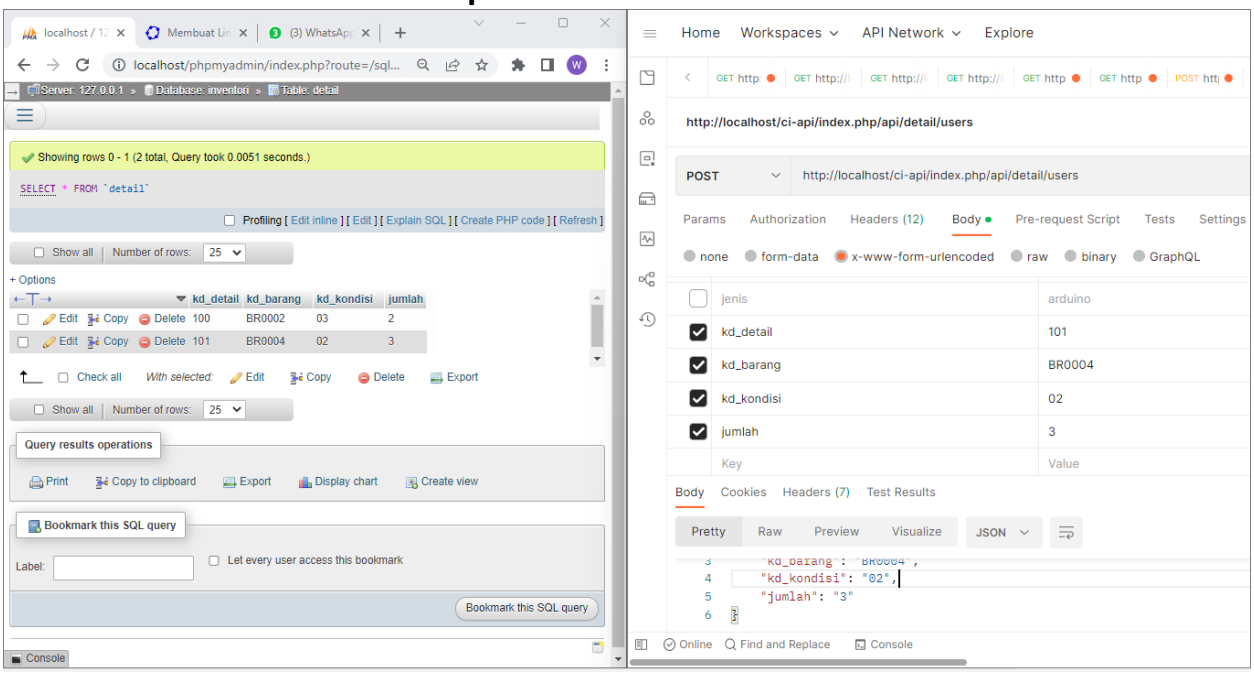
```
            foreach ($users as $key => $value) {
                if (isset($value['kd_detail']) && $value['kd_detail'] === $id) {
                    $user = $value;
                }
            }
        }

        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
    }
}
```

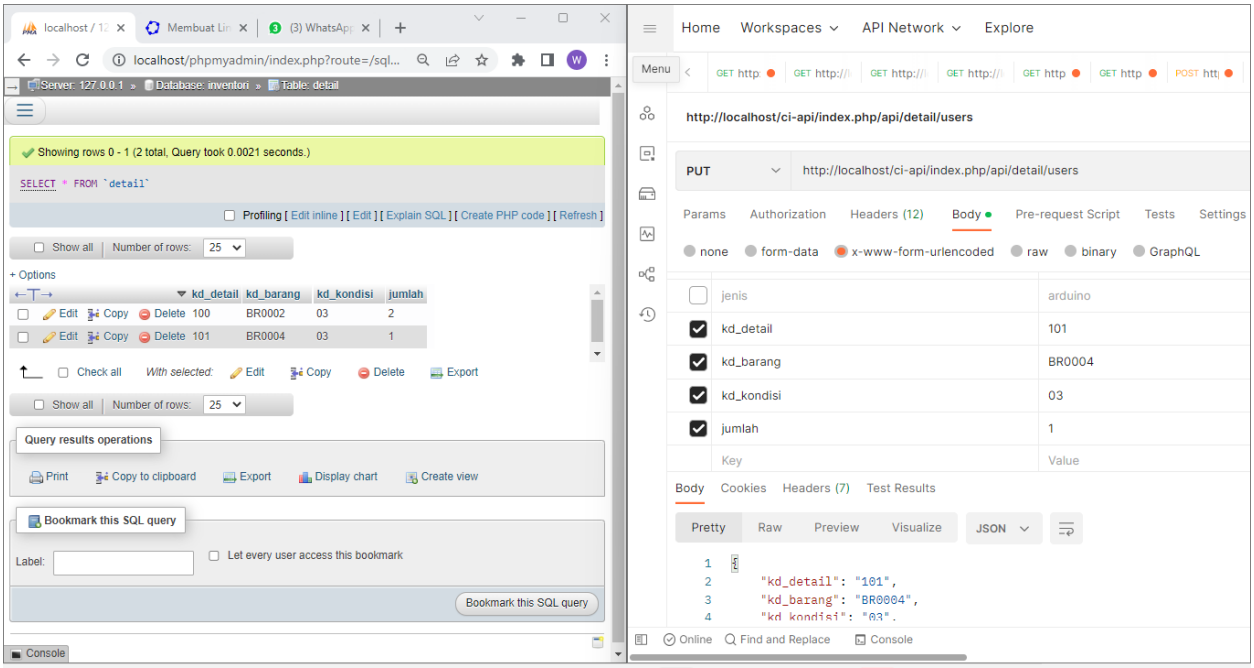
Menampilkan database ke postman



Menambahkan database dari postman

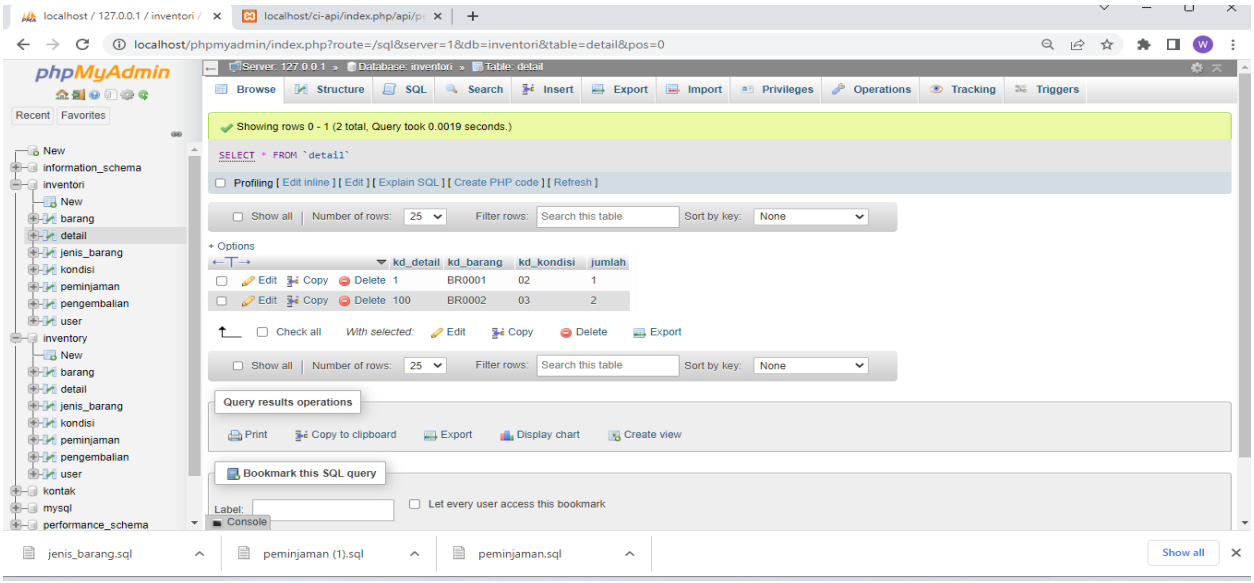


Update

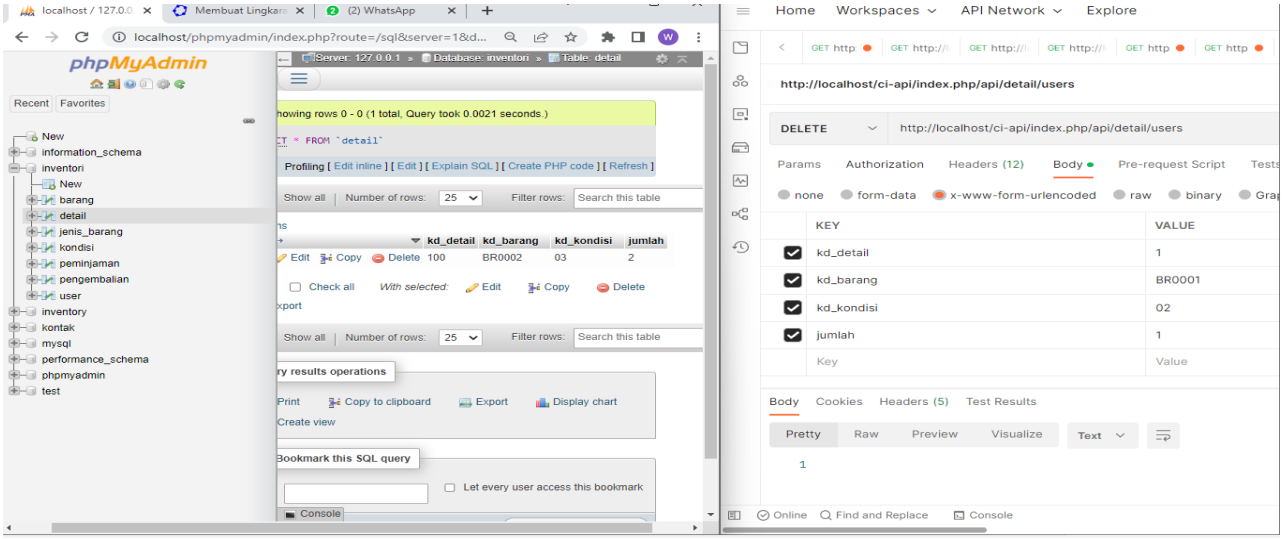


Delete

Before didatabase



After dipostman



- **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.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 barang extends REST_Controller
```

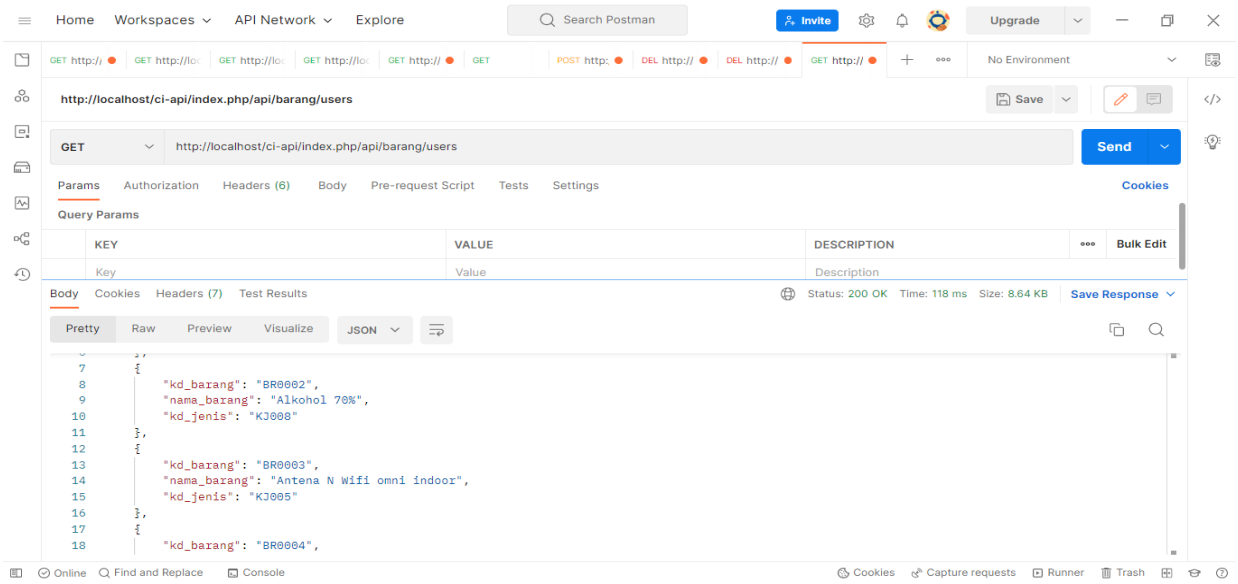
```
{

    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
```

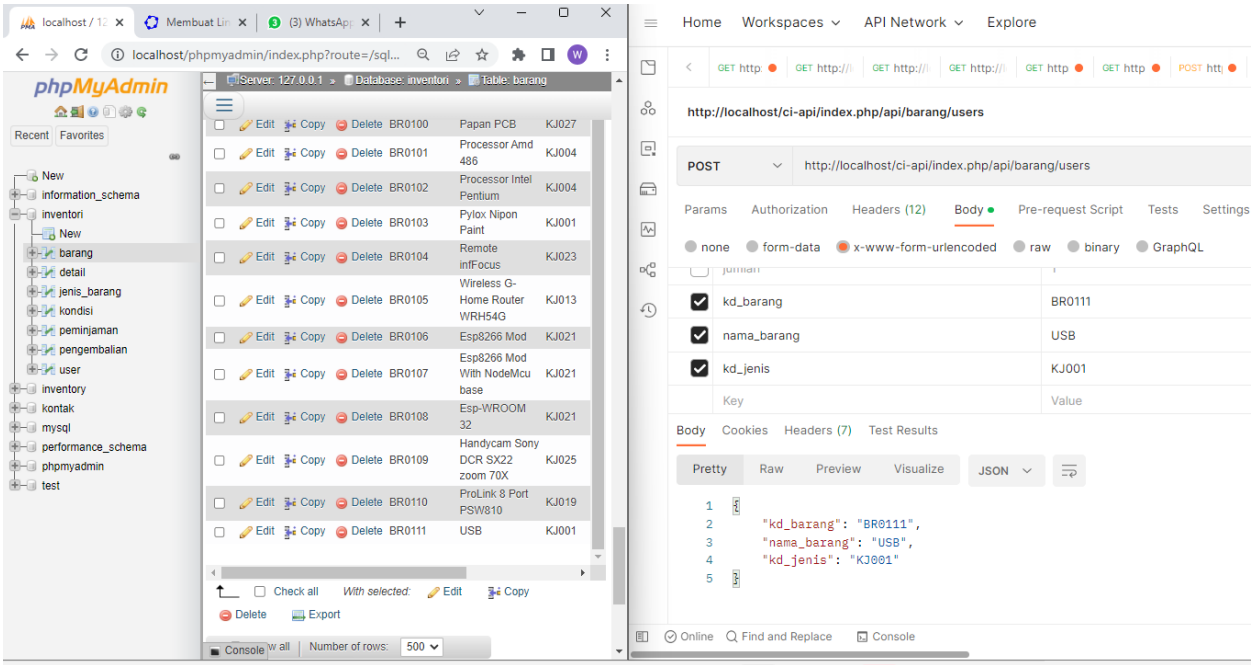
```
        }
    }
}

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
    }
}
```

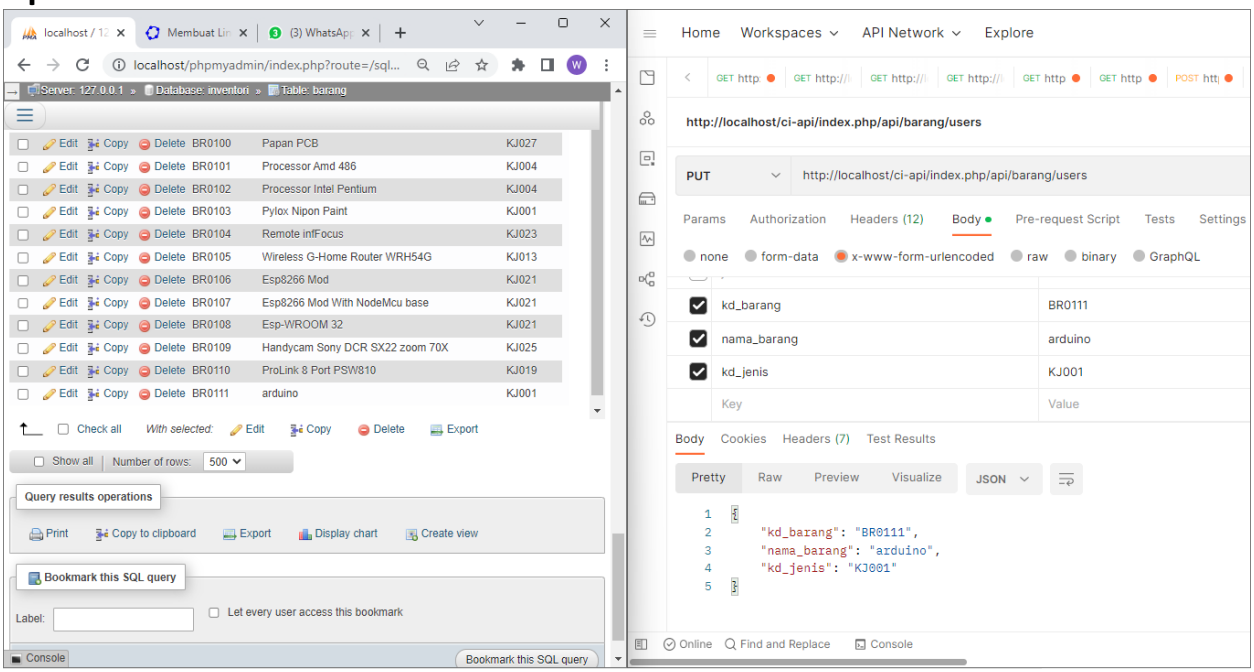
Menampilkan database ke postman



Menambahkan database dari postman

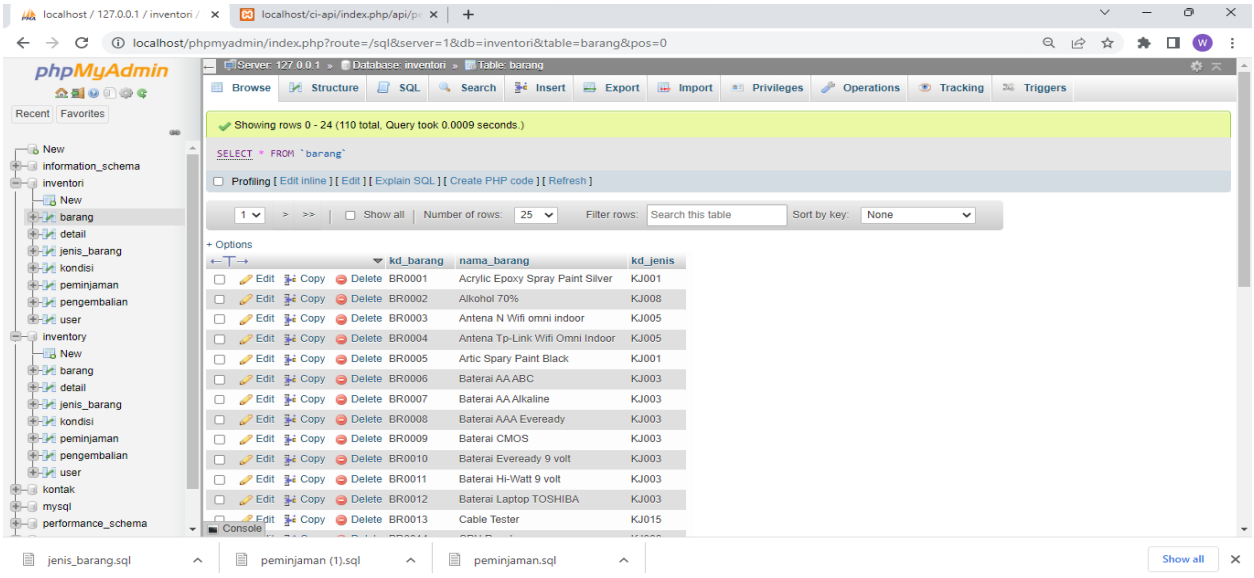


Update



Delete

Before didatabase



After dipostman