LAPORAN TUGAS AKHIR

TEAM CYBER

MOVIE WEB BERBASIS SERVICE

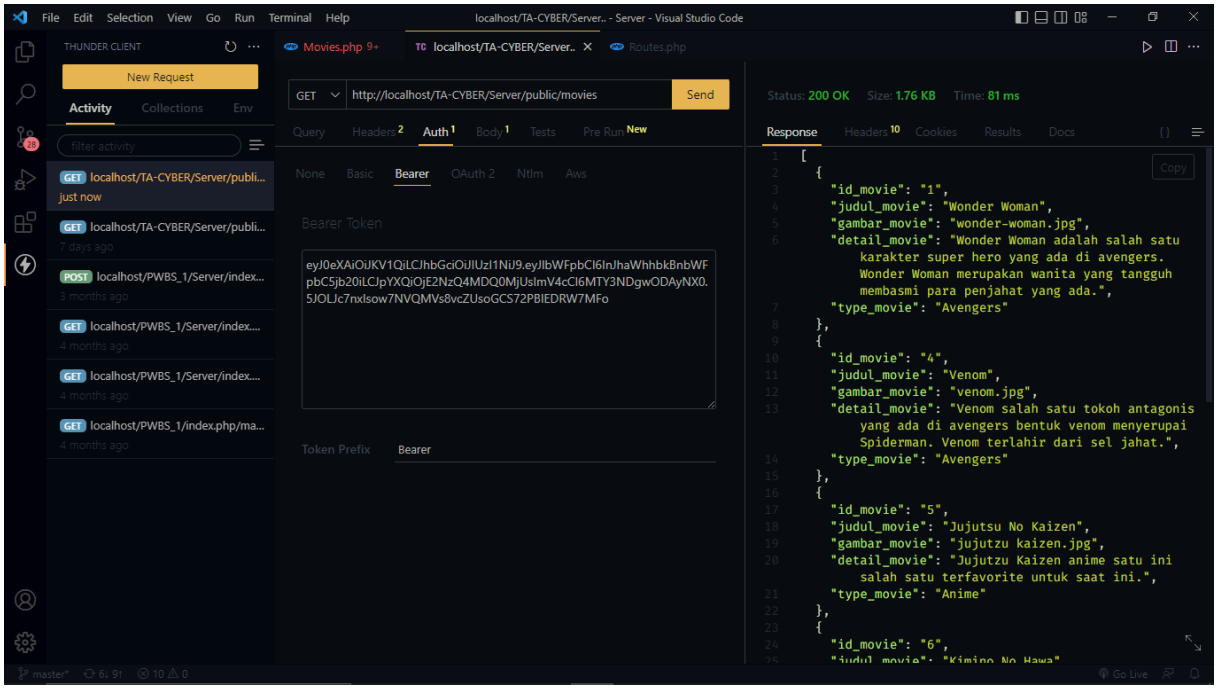
REST Server

Controller Movies :

```php
<?php
namespace App\Controllers;

use CodeIgniter\API\ResponseTrait;
use App\Models\ModelMovie;

class Movies extends BaseController
{
    use ResponseTrait;
    function __construct()
    {
        $this->model = new ModelMovie();
    }
    public function index()
    {
        $data = $this->model->orderBy('id_movie', 'ASC')->findAll();
        return $this->respond($data, 200);
    }
```

Disini saya membuat controller *movies* dimana controller ini sebagai controller utama dari REST Server. Didalam controller *movies* ini terdapat beberapa *method* atau *function* salah satunya ialah index.

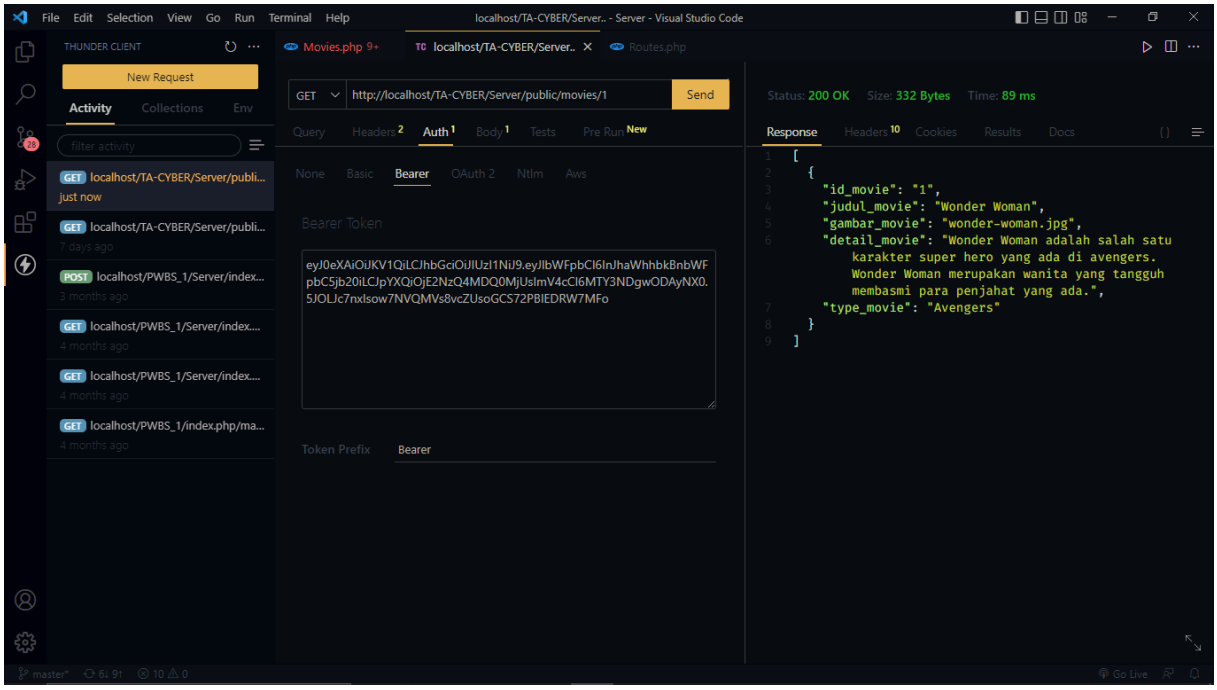
Method *index* memiliki fungsi untuk menampilkan data. Variable *data* berfungsi untuk menyimpan query yang dibuat lalu mengembalikan fungsi respond dengan kode 200. Berikut hasil dari method index :




```php
public function show($id = null)
{
    $data = $this->model->where("id_movie", $id)->findAll();
    if($data) {
        return $this->respond($data, 200);
    } else {
        return $this->failNotFound("Movie tidak ada untuk id : $id");
    }
}
```

Method *show* berfungsi untuk menampilkan data berdasarkan dari id_movie. Variable *data* berfungsi untuk menyimpan query dimana query ini memfilter data dari parameter *id*. Kemudian diberikan kondisi untuk kemanan jika keadaan terpenuhi mengembalikan respond dengan kode 200 yang artinya datanya ditemukan, selain dari itu Movie tidak ada dari *id* tersebut.

Berikut hasil dari method *show* :

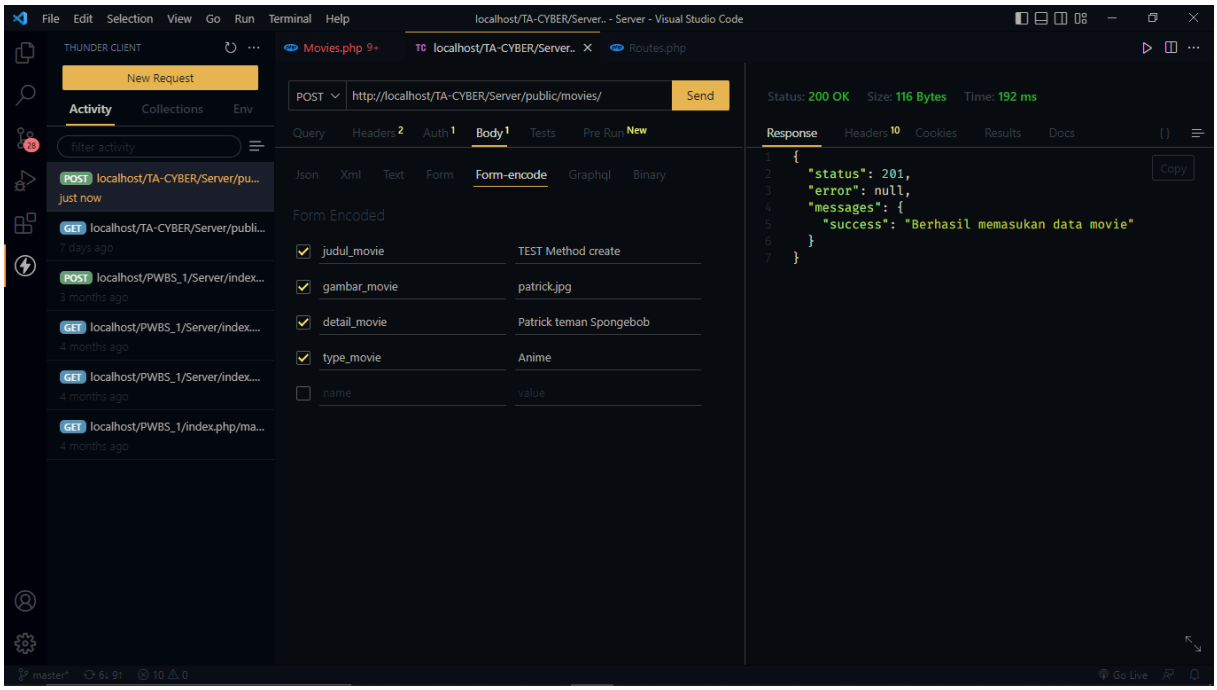


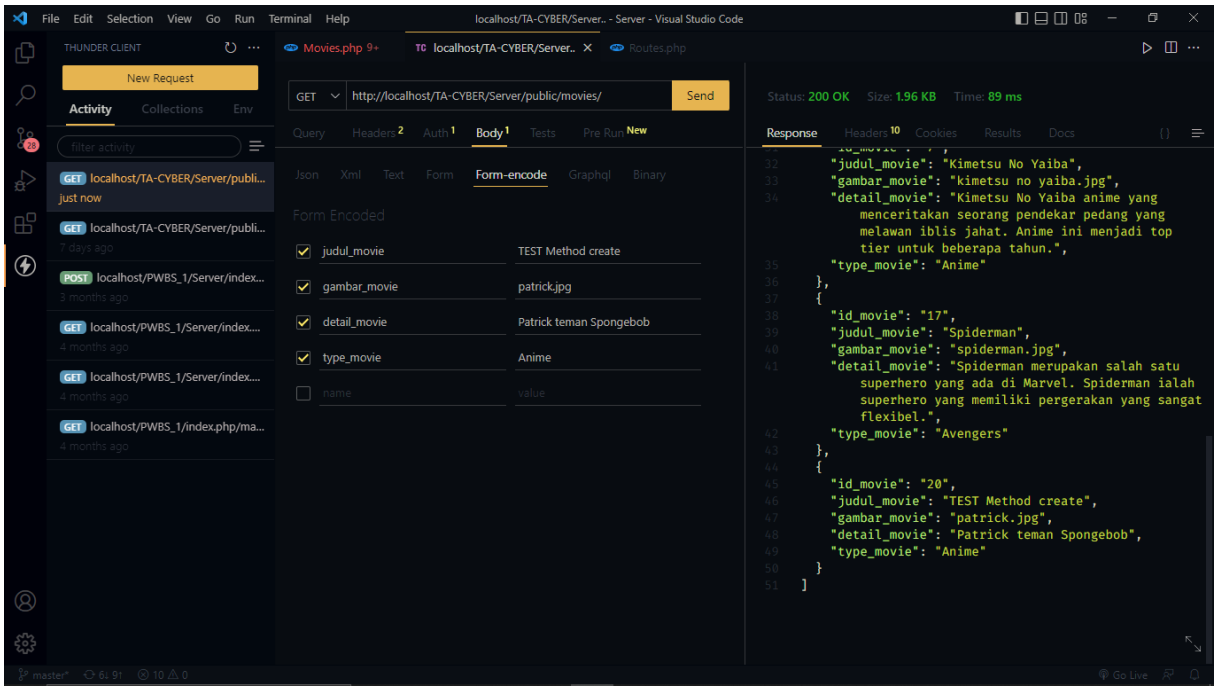

```php
public function create()
{
    $data = [
        'judul_movie' => $this->request->getVar('judul_movie'),
        'gambar_movie' => $this->request->getVar('gambar_movie'),
        'detail_movie' => $this->request->getVar('detail_movie'),
        'type_movie' => $this->request->getVar('type_movie')
    ];

    // $this->model->save($data);
    if(!$this->model->save($data)) {
        return $this->fail($this->model->errors());
    }
    $response = [
        'status' => 201,
        'error' => null,
        'messages' => [
            'success' => 'Berhasil memasukan data movie'
        ]
    ];
    return $this->respond($response);
}
```

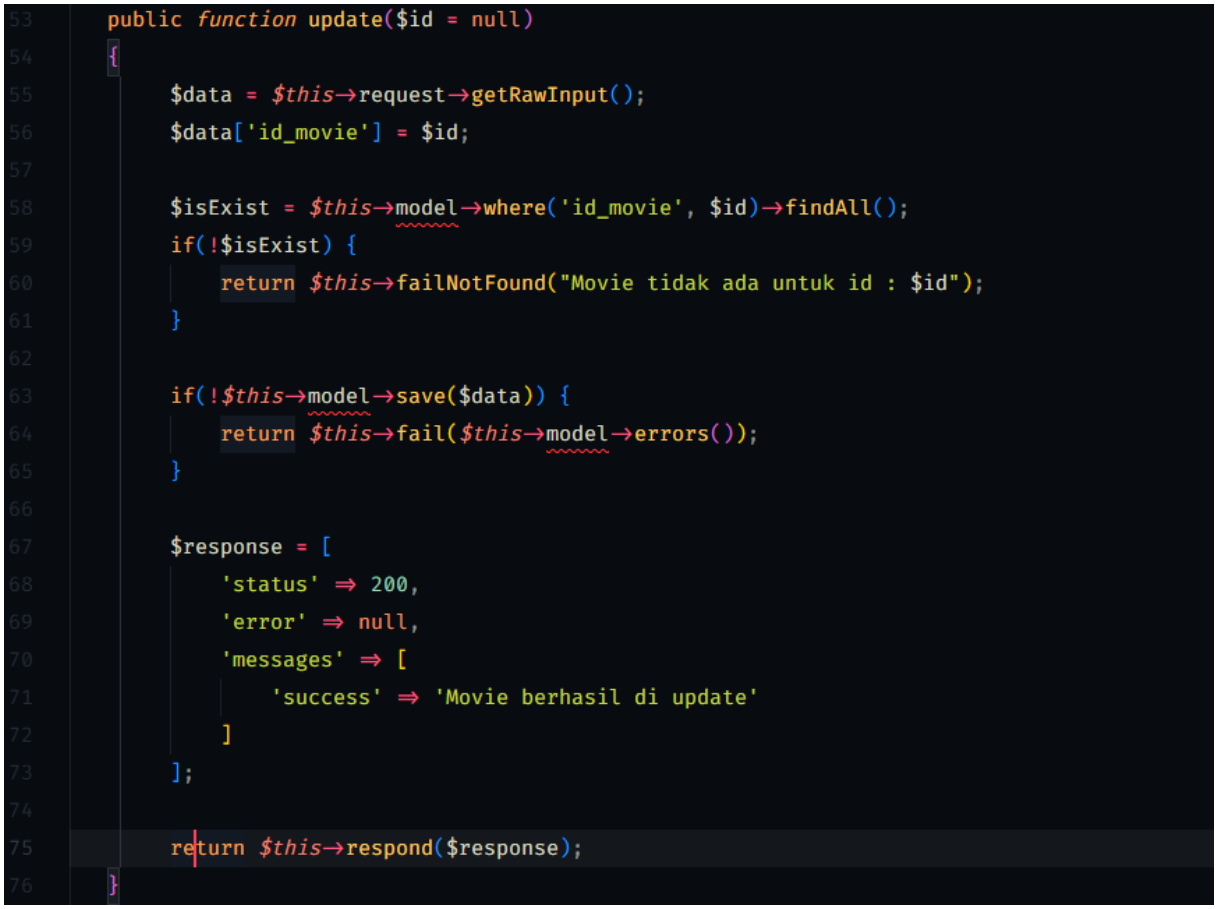
Method *create* berfungsi untuk menambahkan data baru pada REST Server. Variable *data* merupakan array yang berfungsi untuk menangkap data yang dikirim oleh client. Setelah data ditangkap kemudian

dicek apakah terjadi error atau tidak jika error maka akan mengembalikan pesan error jika tidak error mengembalikan respond Berhasil memasukan data movie dengan status kode 201.

Berikut hasil dari method *create* :



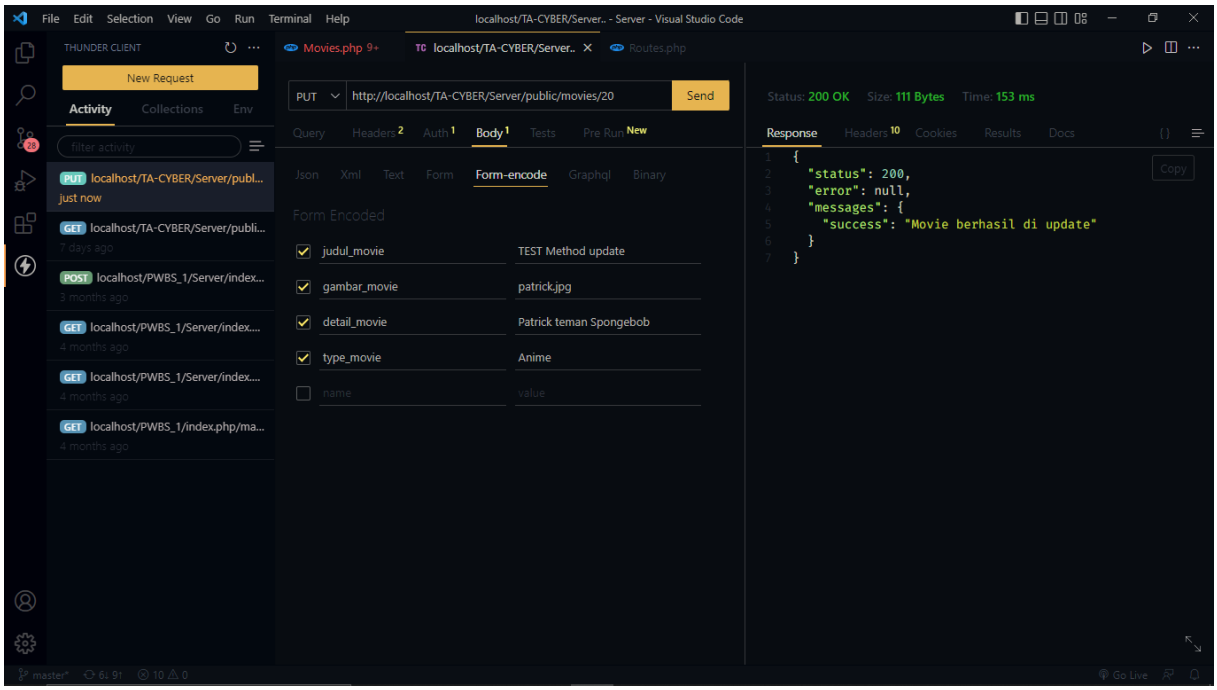


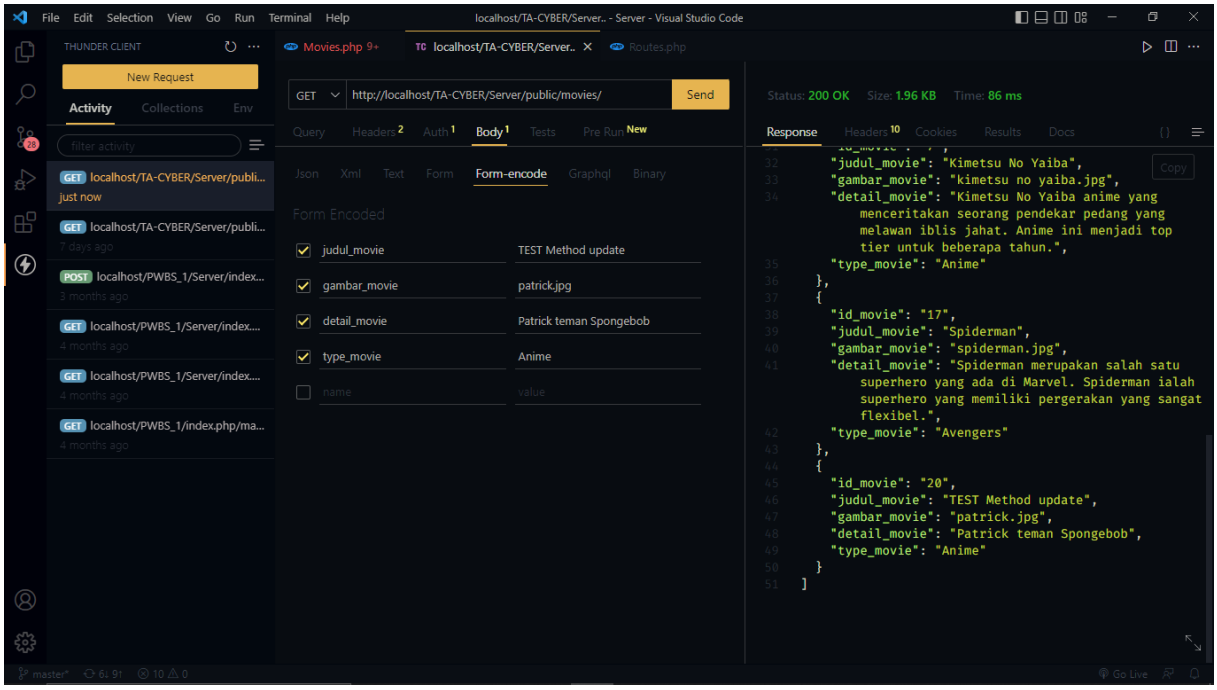


```php
public function update($id = null)
{
    $data = $this->request->getRawInput();
    $data['id_movie'] = $id;

    $isExist = $this->model->where('id_movie', $id)->findAll();
    if(!$isExist) {
        return $this->failNotFound("Movie tidak ada untuk id : $id");
    }

    if(!$this->model->save($data)) {
        return $this->fail($this->model->errors());
    }

    $response = [
        'status' => 200,
        'error' => null,
        'messages' => [
            'success' => 'Movie berhasil di update'
        ]
    ];

    return $this->respond($response);
}
```

Method *update* berfungsi untuk mengupdate data dari REST Server. Pada metodh *update* memiliki parameter *id* dimana parameter tersebut sebagai acuan ketika melakukan update data. Variable *data* digunakan untuk menampung data yang di request dari client. Variable *isExist* berfungsi untuk menampung data berdasarkan *id_movie*. Fungsi *if* untuk mengecek kondisi dari variable *isExist* jika variable atau datanya tidak ada maka akan mengembalikan Movie tidak ada untuk id dan jika data terdapat kesalahan akan mengembalikan fungsi fail dimana menandakan file gagal di update. Jika lolos pengecekan maka akan mengembalikan fungsi respond dengan status kode 200 yang artinya data berhasil di update.

Berikut hasil method *update* :

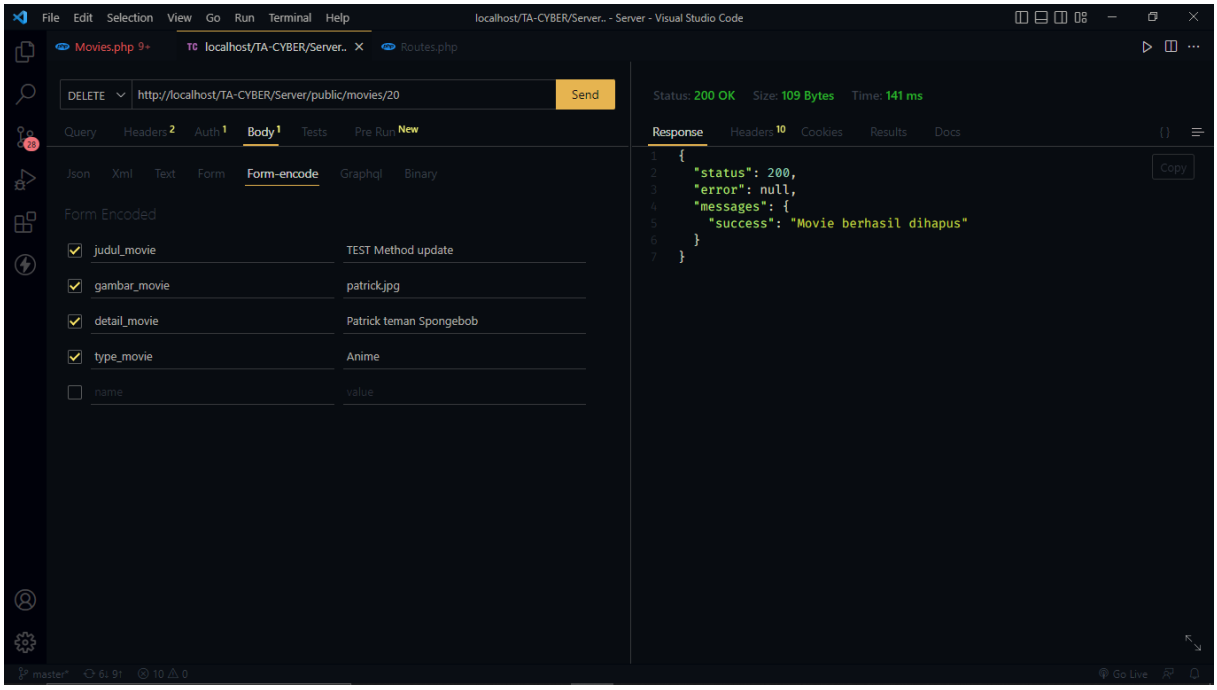


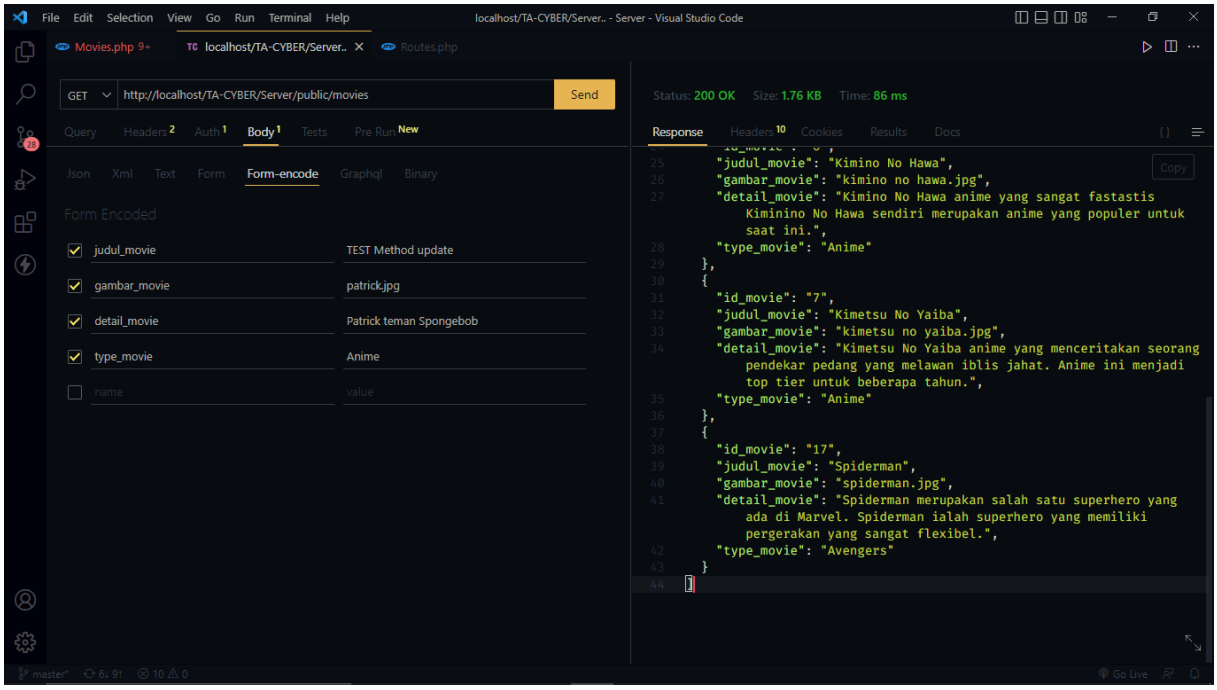


```php
public function delete($id = null)
{
    $data = $this->model->where('id_movie', $id)->findAll();
    if($data) {
        $this->model->delete($id);
        $response = [
            'status' => 200,
            'error' => null,
            'messages' => [
                'success' => 'Movie berhasil dihapus'
            ]
        ];
        return $this->respondDeleted($response);
    } else {
        return $this->failNotFound("Movie tidak ditemukan");
    }
}
}
```

Method *delete* berfungsi untuk menghapus data dari REST Server variable *data* digunakan untuk menampung query berdasarkan id_movie. *If* berfungsi untuk memeriksa jika variable data bernilai true maka data dari REST Server berhasil dihapus jika bernilai false Movie tidak ditemukan.

Berikut hasil method *delete* :





Controller Otentikasi :

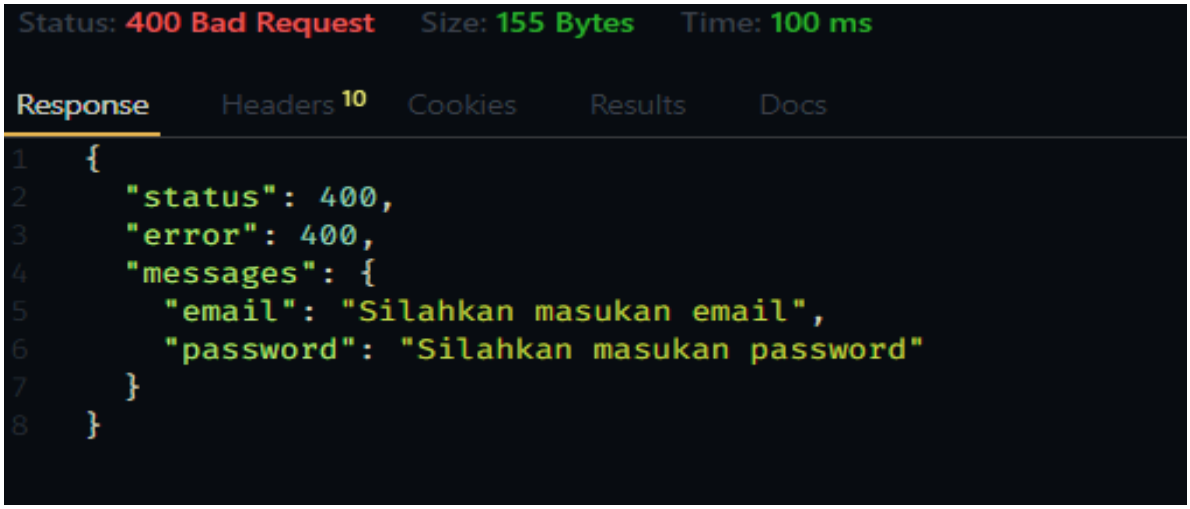
```php
<?php

namespace App\Controllers;

use CodeIgniter\API\ResponseTrait;
use App\Models\ModelOtentikasi;

class Otentikasi extends BaseController
{
    use ResponseTrait;
    public function index()
    {
        // return view('welcome_message');
        $validation = \Config\Services::validation();
        $aturan = [
            'email' => [
            'rules' => 'required|valid_email',
                'errors' =>[
                    'required' => 'Silahkan masukan email',
                    'valid_email' => 'Silahkan masukan email yang valid'
                ]
            ],
```

Controller Otentikasi ialah controller yang berfungsi untuk memeriksa apakah email dan password ada didalam database, jika data ada maka akan memunculkan token jwt dimana token ini berfungsi sebagai kunci agar kita sebagai client bisa mengakses data dari REST Server. Berikut code yang ada didalam controller *otentikasi* variable aturan adalah variable yang berisi aturan apa saja yang akan diterapkan pada REST Server.

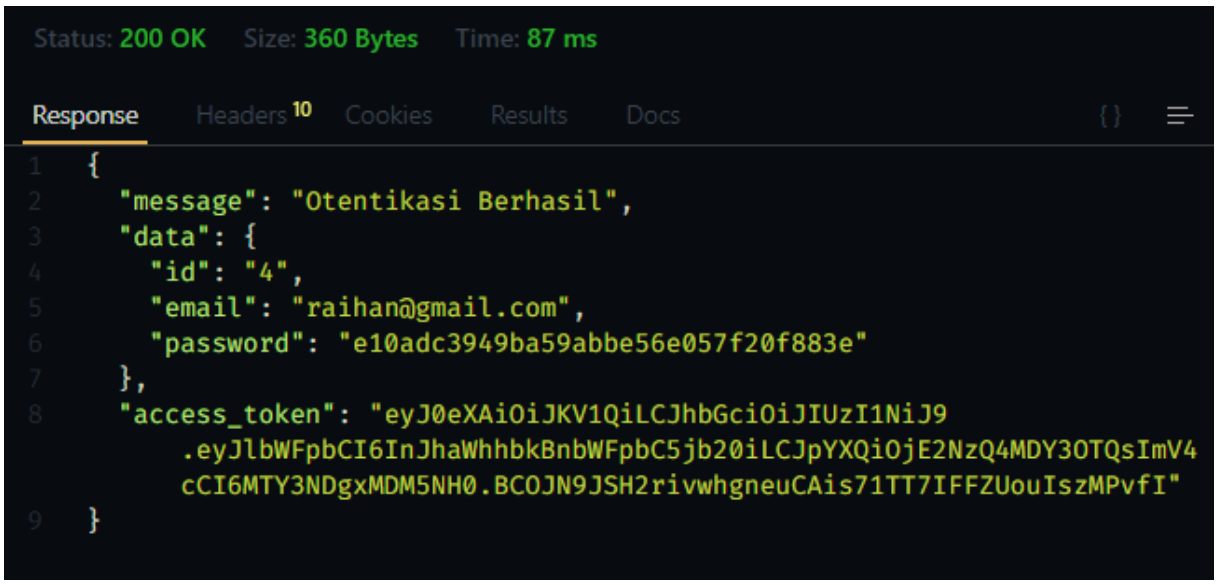
```
        $validation->setRules($aturan);
        if(!$validation->withRequest($this->request)->run()) {
            return $this->fail($validation->getErrors());
        }

        $model = new ModelOtentikasi();

        $email = $this->request->getVar('email');
        $password = $this->request->getVar('password');

        $data = $model->getEmail($email);
        if($data['password'] != md5($password)) {
            return $this->fail('Password tidak sesuai');
        }

        helper('jwt');
        $response = [
            'message' => 'Otentikasi Berhasil',
            'data' => $data,
            'access_token' => createJWT($email)
        ];
        return $this->respond($response);
    }
}
```

Code berikut merupakan validasi yang diterapkan untuk controller *otentikasi* Jika data berhasil lolos dari tahap pengecekan maka akan mengembalikan fungsi respond dimana Otentikasi Berhasil dan memberikan access_token.

Berikut hasil dari validasi otentikasi :



Berikut hasilnya jika data yang dikirim sesuai :

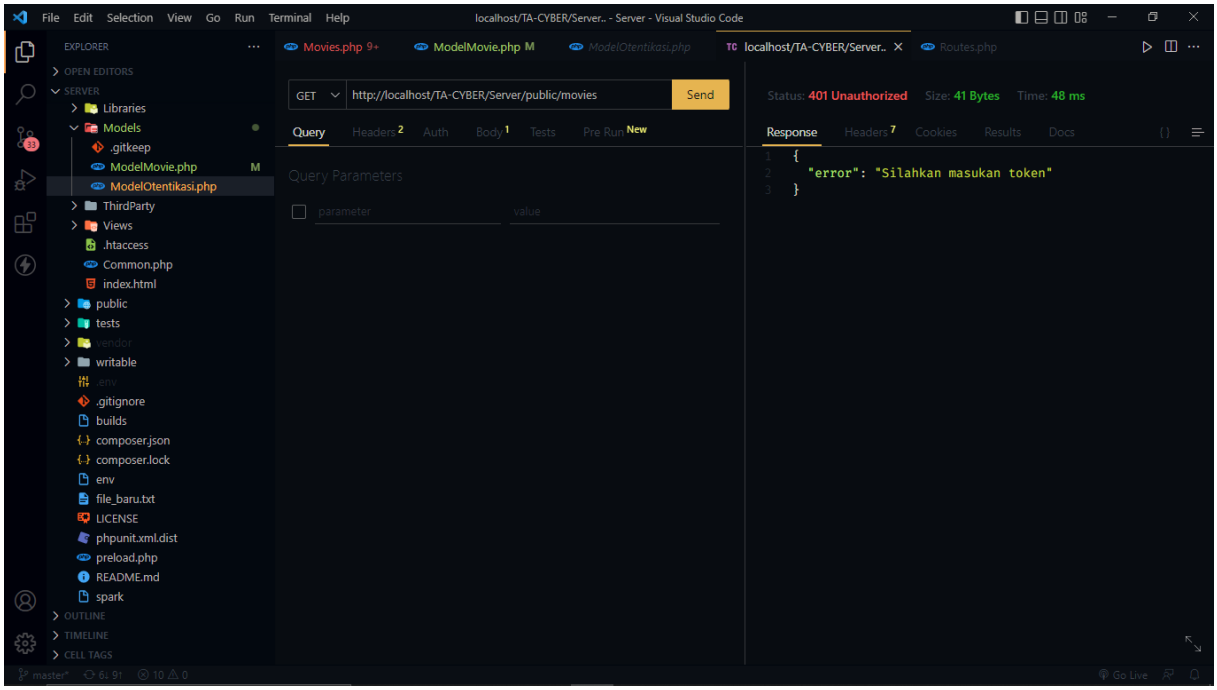


Jwt_helper :

```php
<?php
use App\Models\ModelOtentikasi;
use Firebase\JWT\JWT;
use Firebase\JWT\Key;

function getJWT($otentikasiHeader) {
    if(is_null($otentikasiHeader)) {
        throw new Exception("Silahkan masukan token");
    }
    return explode(" ", $otentikasiHeader)[1];
}
function validateJWT($encodedToken)
{
    $key = getenv('JWT_SECRET_KEY');
    // $decodedToken = JWT::decode($encodedToken, $key, ['HS256']);
    $decodedToken = JWT::decode($encodedToken, new Key($key, 'HS256'));
    $modelOtentikasi = new ModelOtentikasi();

    // cek email
    $modelOtentikasi->getEmail($decodedToken->email);
}
```

Helper ini memiliki beberapa method diantaranya getJWT dimana berfungsi untuk memeriksa headers apakah terdapat token atau tidak ketika headers request tidak menyertakan token maka akan mengeluarkan pesan Silahkan masukan token jika memiliki token maka data dari REST Server akan di tampilkan.

Berikut hasilnya :



```php
function validateJWT($encodedToken)
{
    $key = getenv('JWT_SECRET_KEY');
    // $decodedToken = JWT::decode($encodedToken, $key, ['HS256']);
    $decodedToken = JWT::decode($encodedToken, new Key($key, 'HS256'));
    $modelOtentikasi = new ModelOtentikasi();

    // cek email
    $modelOtentikasi->getEmail($decodedToken->email);
}
```

Method validateJWT memiliki satu parameter encodedToken dimana method ini berfungsi untuk memvalidasi token JWT.

```php
function createJWT($email)
{
    $waktuRequest = time();
    $waktuToken = getenv('JWT_TIME_TO_LIVE');
    $waktuExpired = $waktuRequest + $waktuToken;
    $payload = [
        'email' => $email,
        'iat' => $waktuRequest,
        'exp' => $waktuExpired
    ];

    $jwt = JWT::encode($payload, getenv('JWT_SECRET_KEY'), 'HS256');
    return $jwt;
}
```

Method createJWT berfungsi untuk membuat token *JWT* dimana token tersebut memiliki waktu expired. Setelah token dibuat akan membuat payload yang terdiri dari email, iat dan exp dan kemudian mengembalikan *jwt token*.

Model Movie :

```php
<?php

namespace App\Models;
use CodeIgniter\Model;
// use CodeIgniter\API\ResponseTrait;

class ModelMovie extends Model
{
    protected $table = 'tbl_movie';
    protected $primaryKey = 'id_movie';
    protected $allowedFields = ['judul_movie', 'gambar_movie', 'detail_movie', 'type_movie'];

    protected $validationRules = [
        'judul_movie' => 'required',
        'type_movie' => 'required'
    ];

    protected $validationMessages = [
        'judul_movie' => [
            'required' => 'Silahkan diisi judul movie'
        ]
        ];
}
```

ModelMovie berfungsi untuk memberikan beberapa rules atau aturan serta fields yang boleh di edit pada table yang diterapkan pada REST Server.

```php
<?php

namespace App\Models;
use CodeIgniter\Model;
use Exception;

class ModelOtentikasi extends Model {
    protected $table = 'otentikasi';
    protected $primaryKey = 'id';
    protected $allowedFields = ['email', 'password'];

    function getEmail($email)
    {
        $builder = $this->table("otentikasi");
        $data = $builder->where("email", $email)->first();
        if(!$data) {
            throw new Exception("Data otentikasi tidak ditemukan!");
        }
        return $data;
    }

}
```

ModelOtentikasi berfungsi menerapkan model atau database ini pada REST Server yang dan mengizinkan fields yang di edit pada table. Selain beberapa rules atau aturan model ini juga memiliki

method getEmail dimana berfungsi ketika client ingin meminta otentikasi tidak sesuai email yang ada di database maka akan mengeluarkan pesan Dara otentikasi tidak ditemukan.

.Env :

```
JWT_SECRET_KEY = r41hANm0v13s
JWT_TIME_TO_LIVE = 3600

#--------------------------------------------------------------------
# ENVIRONMENT
#--------------------------------------------------------------------

CI_ENVIRONMENT = development

#--------------------------------------------------------------------
# APP
#--------------------------------------------------------------------

app.baseURL = 'http://localhost/TA-CYBER/Server/public/'
```

```
database.default.hostname = localhost
database.default.database = db_cyber_movie
database.default.username = root
database.default.password = ""
database.default.DBDriver = MySQLi
database.default.DBPrefix =
database.default.port = 3306
```

File .env merupakan file yang sangat penting dimana file ini berisi beberapa konfigurasi yang diterapkan seperti JWT_SECRET_KEY, JWT_TIME_TO_LIVE dan database yang digunakan serta baseURL.