

LAPORAN TUGAS AKHIR
TEAM CYBER
MOVIE WEB BERBASIS SERVICE

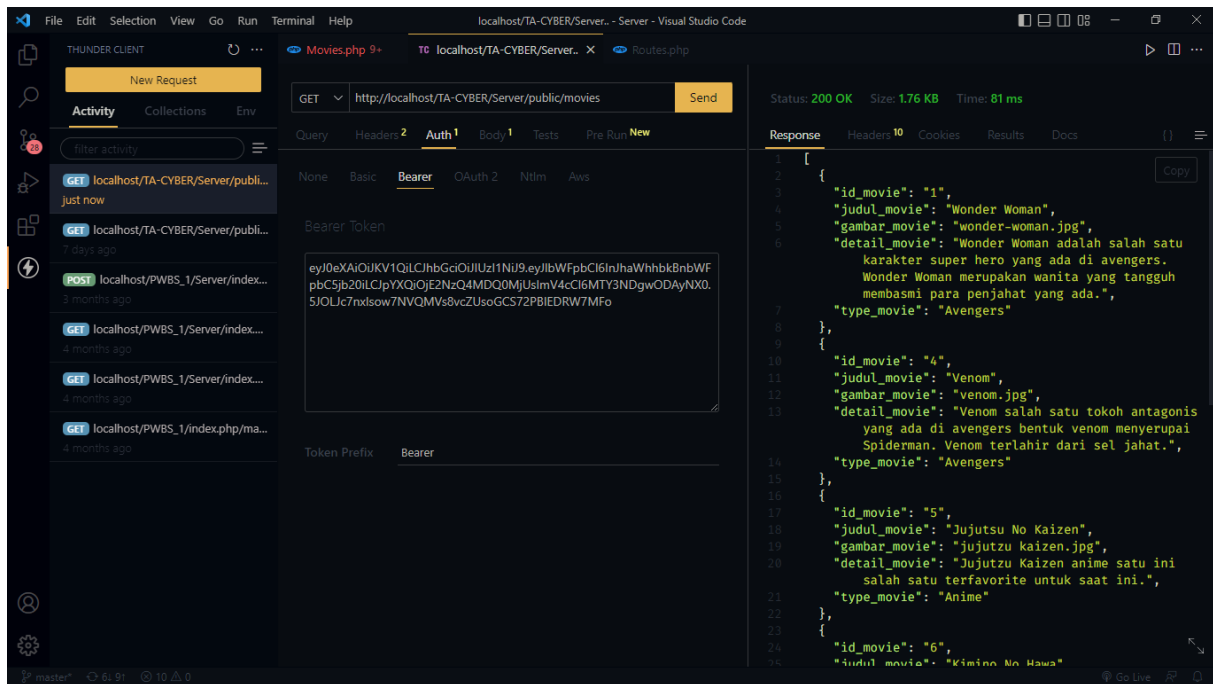
REST Server

Controller Movies :

```
1  <?php
2  namespace App\Controllers;
3
4  use CodeIgniter\API\ResponseTrait;
5  use App\Models\ModelMovie;
6
7  class Movies extends BaseController
8  {
9      use ResponseTrait;
10     function __construct()
11     {
12         $this->model = new ModelMovie();
13     }
14     public function index()
15     {
16         $data = $this->model->orderBy('id_movie', 'ASC')->findAll();
17         return $this->respond($data, 200);
18     }
```

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*.

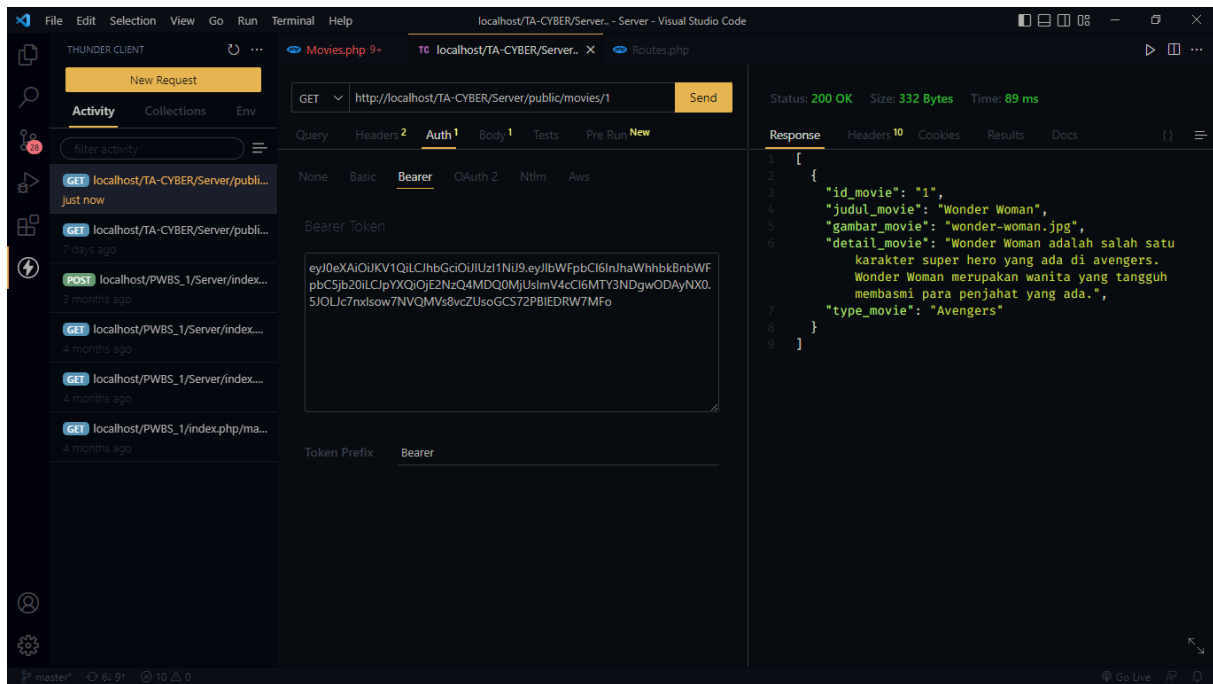
Method *index* memiliki fungsi untuk menampilkan data. Variable *data* berfungsi untuk menyimpan query yang dibuat lalu mengembalikan fungsi respond dengan kode 200. Berikut hasil dari method *index* :



```
20 public function show($id = null)
21 {
22     $data = $this->model->where("id_movie", $id)->findAll();
23     if($data) {
24         return $this->respond($data, 200);
25     } else {
26         return $this->failNotFound("Movie tidak ada untuk id : $id");
27     }
28 }
```

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 keamanan 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* :

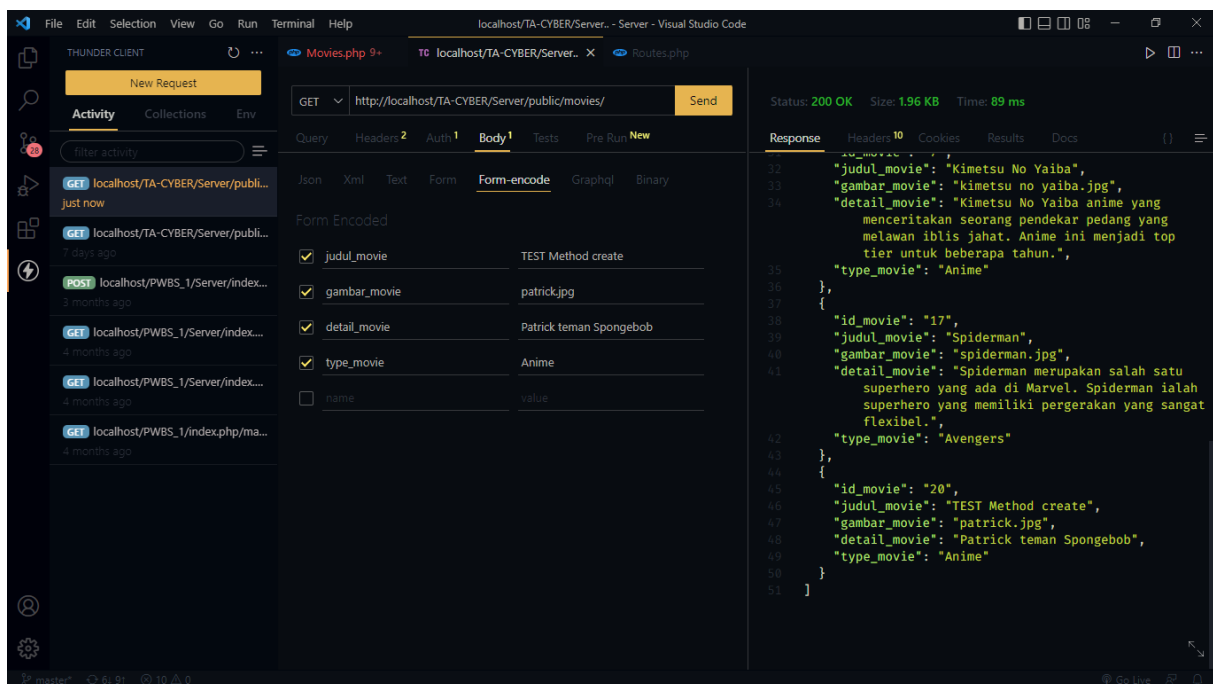
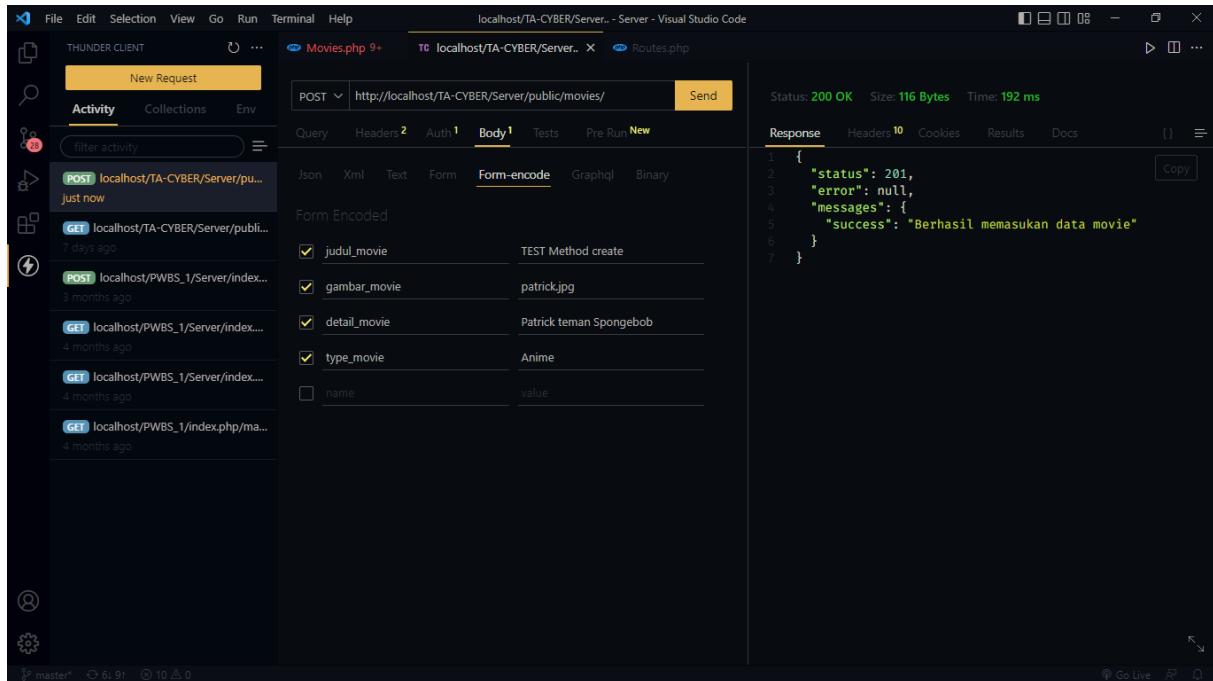


```
30 public function create()
31 {
32     $data = [
33         'judul_movie' => $this->request->getVar('judul_movie'),
34         'gambar_movie' => $this->request->getVar('gambar_movie'),
35         'detail_movie' => $this->request->getVar('detail_movie'),
36         'type_movie' => $this->request->getVar('type_movie')
37     ];
38
39     // $this->model->save($data);
40     if(!$this->model->save($data)) {
41         return $this->fail($this->model->errors());
42     }
43     $response = [
44         'status' => 201,
45         'error' => null,
46         'messages' => [
47             'success' => 'Berhasil memasukan data movie'
48         ]
49     ];
50     return $this->respond($response);
51 }
```

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* :



```

53 public function update($id = null)
54 {
55     $data = $this->request->getRawInput();
56     $data['id_movie'] = $id;
57
58     $isExist = $this->model->where('id_movie', $id)->findAll();
59     if(!$isExist) {
60         return $this->failNotFound("Movie tidak ada untuk id : $id");
61     }
62
63     if(!$this->model->save($data)) {
64         return $this->fail($this->model->errors());
65     }
66
67     $response = [
68         'status' => 200,
69         'error' => null,
70         'messages' => [
71             'success' => 'Movie berhasil di update'
72         ]
73     ];
74
75     return $this->respond($response);
76 }

```

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* :

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

PUT localhost/TACYBER/Server/public/movies/20 just now

GET localhost/TACYBER/Server/public/movies/17 7 days ago

POST localhost/PWBS_1/Server/index.php/insertMovie 3 months ago

GET localhost/PWBS_1/Server/index.php/getMovieById 4 months ago

GET localhost/PWBS_1/Server/index.php/getMovieByTitle 4 months ago

GET localhost/PWBS_1/Server/index.php/getMovieByTitle 4 months ago

GET localhost/PWBS_1/index.php/main 4 months ago

PUT http://localhost/TACYBER/Server/public/movies/20 Send

Status: 200 OK Size: 111 Bytes Time: 153 ms

Response Headers Cookies Results Docs

```
1 {
2   "status": 200,
3   "error": null,
4   "messages": {
5     "success": "Movie berhasil di update"
6   }
7 }
```

Form Encoded

<input checked="" type="checkbox"/>	judul_movie	TEST Method update
<input checked="" type="checkbox"/>	gambar_movie	patrick.jpg
<input checked="" type="checkbox"/>	detail_movie	Patrick teman Spongebob
<input checked="" type="checkbox"/>	type_movie	Anime
<input type="checkbox"/>	name	value

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

GET localhost/TACYBER/Server/public/movies/ just now

GET localhost/TACYBER/Server/public/movies/17 7 days ago

POST localhost/PWBS_1/Server/index.php/insertMovie 3 months ago

GET localhost/PWBS_1/Server/index.php/getMovieById 4 months ago

GET localhost/PWBS_1/Server/index.php/getMovieByTitle 4 months ago

GET localhost/PWBS_1/Server/index.php/getMovieByTitle 4 months ago

GET localhost/PWBS_1/index.php/main 4 months ago

GET http://localhost/TACYBER/Server/public/movies/ Send

Status: 200 OK Size: 1.96 KB Time: 86 ms

Response Headers Cookies Results Docs

```
32 [
33   {
34     "judul_movie": "Kimetsu No Yaiba",
35     "gambar_movie": "kimetsu no yaiba.jpg",
36     "detail_movie": "Kimetsu No Yaiba anime yang
37       menceritakan seorang pendekar pedang yang
38       melawan iblis jahat. Anime ini menjadi top
39       tier untuk beberapa tahun.",
40     "type_movie": "Anime"
41   },
42   {
43     "id_movie": "17",
44     "judul_movie": "Spiderman",
45     "gambar_movie": "spiderman.jpg",
46     "detail_movie": "Spiderman merupakan salah satu
47       superhero yang ada di Marvel. Spiderman ialah
48       superhero yang memiliki pergerakan yang sangat
49       flexibel.",
50     "type_movie": "Avengers"
51   },
52   {
53     "id_movie": "20",
54     "judul_movie": "TEST Method update",
55     "gambar_movie": "patrick.jpg",
56     "detail_movie": "Patrick teman Spongebob",
57     "type_movie": "Anime"
58   }
59 ]
```

Form Encoded

<input checked="" type="checkbox"/>	judul_movie	TEST Method update
<input checked="" type="checkbox"/>	gambar_movie	patrick.jpg
<input checked="" type="checkbox"/>	detail_movie	Patrick teman Spongebob
<input checked="" type="checkbox"/>	type_movie	Anime
<input type="checkbox"/>	name	value

```

78 public function delete($id = null)
79 {
80     $data = $this->model->where('id_movie', $id)->findAll();
81     if($data) {
82         $this->model->delete($id);
83         $response = [
84             'status' => 200,
85             'error' => null,
86             'messages' => [
87                 'success' => 'Movie berhasil dihapus'
88             ]
89         ];
90         return $this->respondDeleted($response);
91     } else {
92         return $this->failNotFound("Movie tidak ditemukan");
93     }
94 }
95

```

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* :

The screenshot shows the Visual Studio Code interface with a REST client extension. The request is a DELETE method to the URL `http://localhost/TA-CYBER/Server/public/movies/20`. The request body is set to 'Form-encode'. The response is a 200 OK status, 109 bytes, and 141 ms. The response body is a JSON object:

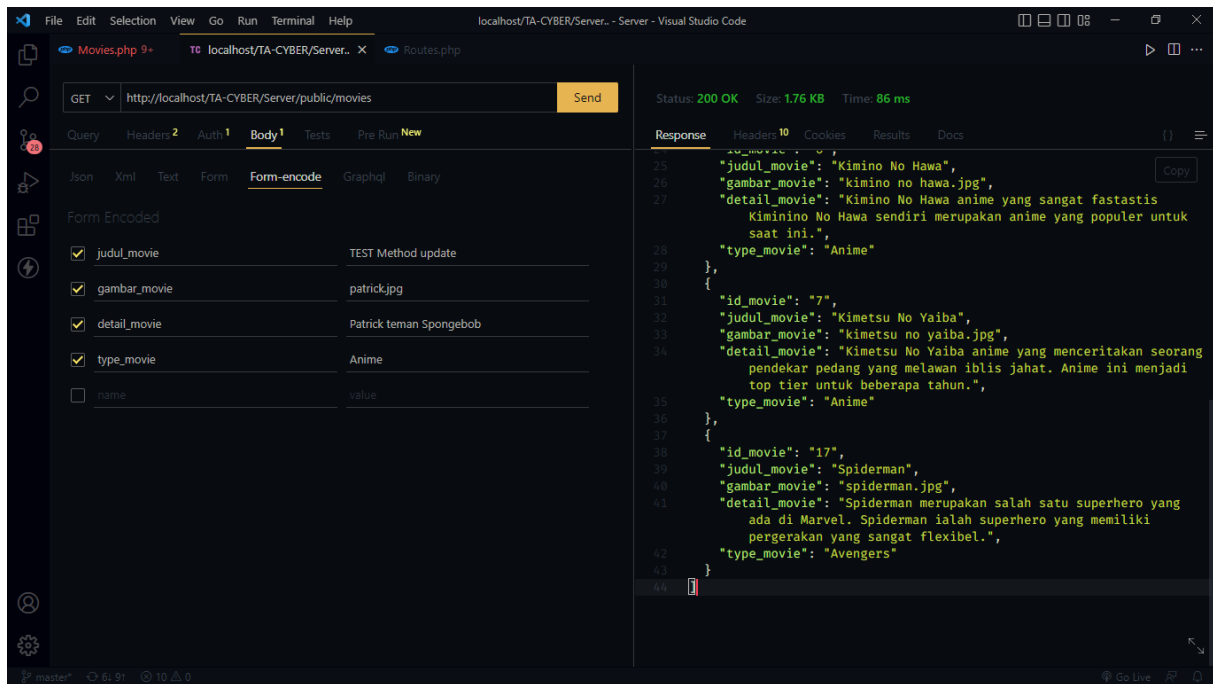
```

{
  "status": 200,
  "error": null,
  "messages": {
    "success": "Movie berhasil dihapus"
  }
}

```

The left sidebar shows the 'Form-encode' tab with the following data:

Field	Value
judul_movie	TEST Method update
gambar_movie	patrick.jpg
detail_movie	Patrick teman Spongebob
type_movie	Anime
name	value



Controller Otentikasi :

```

1  <?php
2
3  namespace App\Controllers;
4
5  use CodeIgniter\API\ResponseTrait;
6  use App\Models\ModelOtentikasi;
7
8  class Otentikasi extends BaseController
9  {
10     use ResponseTrait;
11     public function index()
12     {
13         // return view('welcome_message');
14         $validation = \Config\Services::validation();
15         $aturan = [
16             'email' => [
17                 'rules' => 'required|valid_email',
18                 'errors' => [
19                     'required' => 'Silahkan masukan email',
20                     'valid_email' => 'Silahkan masukan email yang valid'
21                 ]
22             ],

```

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.


```

30     $validation->setRules($aturan);
31     if(!$validation->withRequest($this->request)->run()) {
32         return $this->fail($validation->getErrors());
33     }
34
35     $model = new ModelOtentikasi();
36
37     $email = $this->request->getVar('email');
38     $password = $this->request->getVar('password');
39
40     $data = $model->getEmail($email);
41     if($data['password'] ≠ md5($password)) {
42         return $this->fail('Password tidak sesuai');
43     }
44
45     helper('jwt');
46     $response = [
47         'message' => 'Otentikasi Berhasil',
48         'data' => $data,
49         'access_token' => createJWT($email)
50     ];
51     return $this->respond($response);
52 }
53 }

```

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 :

```
Status: 400 Bad Request Size: 155 Bytes Time: 100 ms

Response Headers 10 Cookies Results Docs
1 {
2   "status": 400,
3   "error": 400,
4   "messages": {
5     "email": "Silahkan masukan email",
6     "password": "Silahkan masukan password"
7   }
8 }
```

Berikut hasilnya jika data yang dikirim sesuai :

```
Status: 200 OK Size: 360 Bytes Time: 87 ms

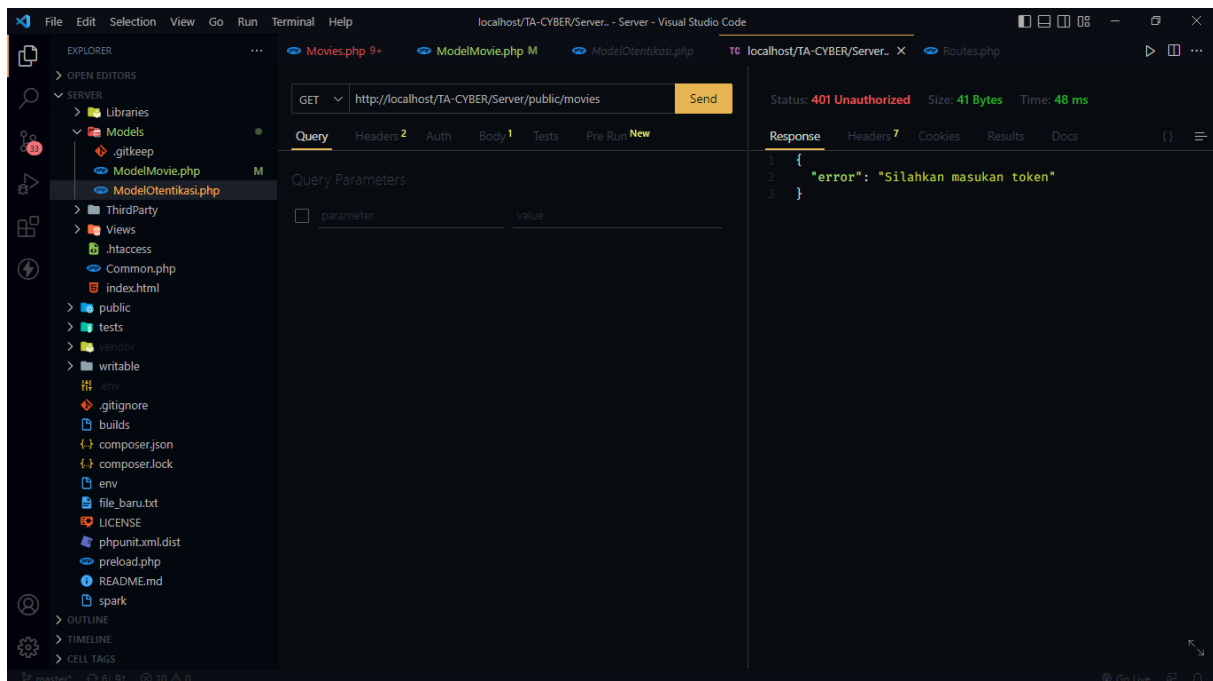
Response Headers 10 Cookies Results Docs {} ≡
1 {
2   "message": "Otentikasi Berhasil",
3   "data": {
4     "id": "4",
5     "email": "raihan@gmail.com",
6     "password": "e10adc3949ba59abbe56e057f20f883e"
7   },
8   "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlbWFpbCI6InJhaWhhbGkiLCJpYXQiOiJlY2NzQ4MDY3OTQsImV4cCI6MTY3NDgxMDM5NH0.BCOJN9JSH2rivwhgneuCAis71TT7IFFZUouIszMPvfI"
9 }
```

Jwt_helper :

```
1 <?php
2 use App\Models\ModelOtentikasi;
3 use Firebase\JWT\JWT;
4 use Firebase\JWT\Key;
5
6 function getJWT($otentikasiHeader) {
7     if(is_null($otentikasiHeader)) {
8         throw new Exception("Silahkan masukan token");
9     }
10    return explode(" ", $otentikasiHeader)[1];
11 }
12 function validateJWT($encodedToken)
13 {
14     $key = getenv('JWT_SECRET_KEY');
15     // $decodedToken = JWT::decode($encodedToken, $key, ['HS256']);
16     $decodedToken = JWT::decode($encodedToken, new Key($key, 'HS256'));
17     $modelOtentikasi = new ModelOtentikasi();
18
19     // cek email
20     $modelOtentikasi->getEmail($decodedToken->email);
21 }
```

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 :



```

12  function validateJWT($encodedToken)
13  {
14      $key = getenv('JWT_SECRET_KEY');
15      // $decodedToken = JWT::decode($encodedToken, $key, ['HS256']);
16      $decodedToken = JWT::decode($encodedToken, new Key($key, 'HS256'));
17      $modelOtentikasi = new ModelOtentikasi();
18
19      // cek email
20      $modelOtentikasi->getEmail($decodedToken->email);
21  }

```

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

```

23  function createJWT($email)
24  {
25      $waktuRequest = time();
26      $waktuToken = getenv('JWT_TIME_TO_LIVE');
27      $waktuExpired = $waktuRequest + $waktuToken;
28      $payload = [
29          'email' => $email,
30          'iat' => $waktuRequest,
31          'exp' => $waktuExpired
32      ];
33
34      $jwt = JWT::encode($payload, getenv('JWT_SECRET_KEY'), 'HS256');
35      return $jwt;
36  }

```

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 :

```

1 <?php
2
3 namespace App\Models;
4 use CodeIgniter\Model;
5 // use CodeIgniter\API\ResponseTrait;
6
7 class ModelMovie extends Model
8 {
9     protected $table = 'tbl_movie';
10    protected $primaryKey = 'id_movie';
11    protected $allowedFields = ['judul_movie', 'gambar_movie', 'detail_movie', 'type_movie'];
12
13    protected $validationRules = [
14        'judul_movie' => 'required',
15        'type_movie' => 'required'
16    ];
17
18    protected $validationMessages = [
19        'judul_movie' => [
20            'required' => 'Silahkan diisi judul movie'
21        ]
22    ];
23 }

```

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

```

1 <?php
2
3 namespace App\Models;
4 use CodeIgniter\Model;
5 use Exception;
6
7 class ModelOtentikasi extends Model {
8     protected $table = 'otentikasi';
9     protected $primaryKey = 'id';
10    protected $allowedFields = ['email', 'password'];
11
12    function getEmail($email)
13    {
14        $builder = $this->table("otentikasi");
15        $data = $builder->where("email", $email)->first();
16        if(!$data) {
17            throw new Exception("Data otentikasi tidak ditemukan!");
18        }
19        return $data;
20    }
21
22 }

```

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 Data otentikasi tidak ditemukan.

.Env :

```
14  JWT_SECRET_KEY = r41hANm0v13s
15  JWT_TIME_TO_LIVE = 3600
16
17  #-----
18  # ENVIRONMENT
19  #-----
20
21  CI_ENVIRONMENT = development
22
23  #-----
24  # APP
25  #-----
26
27  app.baseURL = 'http://localhost/TA-CYBER/Server/public/'
```

```
46  database.default.hostname = localhost
47  database.default.database = db_cyber_movie
48  database.default.username = root
49  database.default.password = ""
50  database.default.DBDriver = MySQLi
51  database.default.DBPrefix =
52  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.