# Pertemuan 13

## Autentikasi dengan RESTFUL API

## Pendahuluan

Terkadang ketika kita membuat website dengan Javascript/Typescript kita membutuhkan data dari server, dan server itu tadi meminta kredensi user agar bisa melakukan pengolahan data, hal ini masih bisa dilakukan di RESTFUL API tetapi dengan menggunakan sistem Token Based Authentication (Autentikasi berbasis token), Jadi ketika si user melakukan login Javascript/Typescript kita mengirimkan data login tadi ke server lalu setelah autentikasi berhasil server akan memberikan response access token yang nanti bisa digunakan untuk akses di server yang memerlukan autentikasi.

Token Based Authentication ini beda dengan JSON Web Token karena Token Based Authentication si server juga menyimpan token tadi, kalau pengguna github pasti kalau nge-cek di settings ada nama-nya "personal access token" nah Token Based Authentication mirip seperti itu.

Cara membuat token untuk Token Based Authentication itu sangat bermacam-macam hal yang paling mudah ialah membuat token menggunakan versi hash username si user jika username user itu unik dan tidak ada yang sama di dalam database, karena ini token di simpan di server jadi kita harus juga menyiapkan tabel yang digunakan untuk menyimpan token tersebut, dan memiliki relasi dengan tabel user.

Untuk pengiriman token ini di Javascript kita menggunakan Authorization Header lalu kita isi Bearer {{ TOKEN }}, lalu di server harus melakukan parsing dulu di Authorization Header request-nya tadi lalu ngambil token itu tadi, lalu di verifikasi kalau itu memang betul token-nya.

Untuk contoh kali ini kita akan membuat REST Server menggunakan PHP, karena ini contoh jadi bisa dikembangkan lagi, karena ini hanya fokus di kode server-nya jadi saya sedikit mengabaikan dibagian frontend-nya, jadi di-sarankan untuk meng-install postman atau extensi REST Client atau Thunder Client untuk melakukan request do API yang kita buat, untuk function fetch di Javascript bisa ditambah.

```
fetch(url, {
      ...
      headers: {
            'Authorization': `Bearer ${token}`
      }
      ...
});
```

# Desain

Sebelum kita membuat REST Server kita, kita harus men-desain terlebih dahulu, yang paling penting database

```
CREATE TABLE IF NOT EXISTS users (
      userId INT PRIMARY KEY AUTO_INCREMENT,
      username VARCHAR(255) UNIQUE NOT NULL,
      password TEXT
);

CREATE TABLE IF NOT EXISTS tokens (
      tokenId INT PRIMARY KEY AUTO_INCREMENT,
      user_id INT,
      token TEXT,
      FOREIGN KEY (user_Id) REFERENCES users(id)
);
```

Nah dari sql di-atas kita membuat tabel users dan tokens, lalu di tabel tokens itu memiliki relasi dengan tabel user, kalau belum tau cara buat relasi di phpmyadmin bisa cek di materi pertemuan 07, kalau malas bisa langsung masuk di tab sql lalu copy paste kode sql di atas.

Setelah itu untuk demonstrasi penggunaan token itu tadi kita bisa menggunakan tabel satu lagi yaitu tabel posts, dan juga kita akan mengikuti standar RESTFUL API sepenuh-nya.

```
CREATE TABLE IF NOT EXISTS posts (
      postId INT PRIMARY KEY AUTO_INCREMENT,
      user_id INT,
      judul VARCHAR(255),
      isi TEXT,
      FOREIGN KEY (user_id) REFERENCES users(id)
);
```

# Schema

Schema yang nanti kita buat RESTFUL Server, untuk {{ BASE_URI }} bisa menyesuaikan, dan {{ TOKEN }} menyesuaikan dengan token yang diberikan oleh server.

Untuk melakukan registrasi.
POST http://{{ BASE_URI }}/api/register.php
Request :

```
{
      “name”: “Zagar”,
      “password”: “password”
}
```

Response :

```
{
	“code”: 201,
	“user”: {
		“id”; xxx,
		“username”: “Zagar”
	},
	“token”: “xxx”
}
```

Untuk melakukan login.
POST http://{{ BASE_URI }}/api/login.php
Request :

```
{
	“name”: “Zagar”,
	“password”: “password”
}
```

Response :

```
{
	“code”: 200,
	“token”: “xxx”
}
```

Untuk Logout.
POST http://{{ BASE_URI }}/api/logout.php
Authorization: Bearer {{ TOKEN }}
Response :

```
{
	“code”: 200,
	“message”: “Logout berhasil!”
}
```

Untuk penambahan post.
POST http://{{ BASE_URI }}/api/post.php
Authorization: Bearer {{ TOKEN }}
Request :

```
{
	“judul”: “xxx”,
	“isi”: “xxx”
}
```

Response :

```
{
	“code”: 201,
	“post”: {
		“id”: xxx,
		“judul”: “xxx”,
		“isi”: “xxx”
	}
}
```

Untuk mengambil semua post
GET http://{{ BASE_URI }}/api/post.php
Response :

```
{
      “code”: 200,
      “posts”: [
            {
                  “id”: xxx,
                  “judul”: “xxx”,
                  “isi”: “xxx”
            },
            {
                  “id”: xxx,
                  “judul”: “xxx”,
                  “isi”: “xxx”
            }
            ...
      ]
}
```

Untuk susunan file
1. /folder-di-htdocs/api/db.php
2. /folder-di-htdocs/api/login.php
3. /folder-di-htdocs/api/register.php
4. /folder-di-htdocs/api/posts.php
5. /folder-di-htdocs/api/response.php

Untuk response.php bisa menggunakan yang praktek-nya RESTFUL API yang dulu, dan db.php juga sama seperti sebelum-nya

# Pembuatan REST Server

## Register.php

Untuk register.php seperti biasa kita melakukan require terhadap db.php dan response.php, lalu melakukan pengambilan data dari request seperti biasa juga, lalu kita juga melakukan pengecekan data yang di dalam request itu tadi

```php
// Ambil data dari request dan melakukan deserialization.
$dataFromRequest = json_decode(file_get_contents('php://input'), true);

// Check apakah request menggunakan http method POST.
if ($_SERVER['REQUEST_METHOD'] ≠ 'POST')
    return response(404, [
        'code' ⇒ 404,
        'message' ⇒ 'Halaman tidak ada'
    ]);

// Check apakah request mencantumkan username dan password.
if (!isset($dataFromRequest['username']) || !isset($dataFromRequest['password']))
    return response(400, [
        'code' ⇒ 400,
        'message' ⇒ 'Data kurang lengkap!'
    ]);
```

Lalu seperti biasa melakukan hash di password lalu kita gunakan data dari request dan hasil hash itu tadi untuk penambahan user di database, setelah itu kita ambil id yang terakhir di tambahkan dan kita simpan di $userId karena kita nanti butuh, lalu kita buat $token dengan hash juga tetapi kali ini menggunakan data username, lalu melakukan penambahan token ke database lalu kita kirim response.

```php
// Melakukan hash password untuk keamanan
$password = password_hash($dataFromRequest['password'], PASSWORD_DEFAULT);

// Menambah user ke database
$stmt = $pdo->prepare("INSERT INTO users (username, password) VALUES (:username, :password)");
$stmt->bindParam('username', $dataFromRequest['username']);
$stmt->bindParam('password', $password);
$stmt->execute();

// ambil id user yang baru ditambah
$userId = $pdo->lastInsertId();

// melakukan enkripsi username untuk membuat token
$token = password_hash($dataFromRequest['username'], PASSWORD_DEFAULT);

// Menambah token ke database
$stmt = $pdo->prepare("INSERT INTO tokens (token, user_id) VALUES (:token, :user_id)");
$stmt->bindParam('token', $token);
$stmt->bindParam('user_id', $userId);
$stmt->execute();

// Lalu mengirim response dengan data user dan token-nya
response(201, [
    'code' => 201,
    'user' => [
        'id' => $userId,
        'username' => $dataFromRequest['username'],
    ],
    'token' => $token
]);
```

## Login.php

Login.php ini sama seperti di register.php tetapi beda-nya kita melakukan verifikasi data user yang di berikan bukan melakukan penambahan user ke database, setelah user di verifikasi lalu kita membuat token baru yang disimpan ke database.

```php
// Ambil data user dari database dengan nama yang sama dengan request tadi
$stmt = $pdo->prepare("SELECT * FROM users WHERE username=:username");
$stmt->setFetchMode(PDO::FETCH_ASSOC);
$stmt->bindParam('username', $dataFromRequest['username']);
$stmt->execute();
$user = $stmt->fetch();

// Check password-nya
if (password_verify($dataFromRequest['password'], $user['password'])) {
    // melakukan enkripsi username untuk membuat token
    $token = password_hash($dataFromRequest['username'], PASSWORD_DEFAULT);

    // Menambah token ke database
    $stmt = $pdo->prepare("INSERT INTO tokens (token, user_id) VALUES (:token, :user_id)");
    $stmt->bindParam('token', $token);
    $stmt->bindParam('user_id', $user['id']);
    $stmt->execute();

    response(201, [
        'code' => 201,
        'token' => $token
    ]);
}
// Kalau password tidak cocok kirim http code 406 Not Acceptable
response(406, [
    'code' => 406,
    'message' => 'Password Salah!'
]);
```

## Logout.php

Untuk logout agak ribet dari praktek yang kita lakukan sebelum-nya, tetapi terdapat contoh untuk melakukan parsing Authorization Header untuk mengambil token-nya, lalu kita melakukan penghapusan token di database menggunakan token dari request

```php
// Check apakah request menggunakan http method POST.
if ($_SERVER['REQUEST_METHOD'] ≠ 'POST')
    return response(404, [
        'code' ⇒ 404,
        'message' ⇒ 'Halaman tidak ada'
    ]);

// Check apakah request membawa HTTP Authorization
if (!isset($_SERVER['HTTP_AUTHORIZATION']))
    return response(400, [
        'code' ⇒ 400,
        'message' ⇒ 'Data kurang lengkap!'
    ]);

// Karena Token-nya "Bearer {{Token}}"
// Kita menggunakan function explode untuk memecah string menjadi array
$token = explode(' ', $_SERVER['HTTP_AUTHORIZATION']);

$stmt = $pdo→prepare("DELETE FROM tokens WHERE token=:token");
$stmt→bindParam('token', $token[1]);
$stmt→execute();

response(200, [
    'code' ⇒ 200,
    'message' ⇒ 'Logout berhasil!'
]);
```

function explode() ini dapat memecah string menjadi array dengan pemisah, di contoh ini yang memisahkan token tersebut adalah “ ” jadi di argument pertama spasi lalu yang kedua adalah string-nya. Jadi kita dapat mengambil token menggunakan array index

## Post.php

Menurut schema yang kita buat post.php ini menerima GET dan POST jadi kita akan melakukan verifikasi kedua method tersebut agar kita mendapat-kan behavior yang pasti.

Untuk GET ini sangat mudah sekali karena di schema tidak perlu login untuk menggunakan-nya jadi kita tinggal kirim saja hasil dari database langsung

```php
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    // Jika request http method-nya GET berikan semua data dari database posts
    $stmt = $pdo->prepare("SELECT * FROM posts");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->execute();
    $posts = $stmt->fetchAll();

    response(200, [
        'code' => 200,
        'posts' => $posts
    ]);
}
```

Untuk POST ini agak sulit, kita harus melakukan pengecekan token-nya tadi, lalu kita verifikasi token-nya apakah sesuai dengan token di database, setelah itu kalau verifikasi gagal kita beri kode 203

```php
// Cek apakah http request method-nya post
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    // Cek apakah memiliki HTTP Authorization
    if (!isset($_SERVER['HTTP_AUTHORIZATION']))
        response(203, [
            'code' => 203,
            'message' => 'Token salah!'
        ]);

    // Olah HTTP Authorization dan ambil token-nya
    $token = explode(' ', $_SERVER['HTTP_AUTHORIZATION']);

    // Ambil token di database sesuai dengan data yang diberikan
    $stmt = $pdo->prepare("SELECT * FROM tokens WHERE token=:token");
    $stmt->setFetchMode(PDO::FETCH_ASSOC);
    $stmt->bindParam('token', $token[1]); // Karena token-nya di index 1
    $stmt->execute();
    $token = $stmt->fetch();

    // Kalau token di database tidak ada langsung response token salah
    if (!$token)
        response(203, [
            'code' => 203,
            'message' => 'Token salah!'
        ]);
}
```

Setelah kita verifikasi kita bisa melakukan parsing data dari request lalu kita melakukan pengecekan apakah data-nya lengkap atau tidak, lalu kita langsung tambahkan ke database dengan id user yang kita dapatkan ketika kita melakukan pengecekan token tadi, lalu kita memberikan response 201 ke client.

```php
// Melakukan deserialize data request
$dataFromRequest = json_decode(file_get_contents('php://input'), true);

// Cek apakah data request telah memenuhi yang nanti kita butuhkan untuk
// Membuat data
if (!isset($dataFromRequest['judul']) || !isset($dataFromRequest['isi']))
    response(400, [
        'code' => 400,
        'message' => 'Data kurang lengkap'
    ]);

// Melakukan penambahan data posts ke database
$stmt = $pdo->prepare("INSERT INTO posts (user_id, judul, isi) VALUES (:user_id, :judul, :isi)");
$stmt->bindParam('user_id', $token['user_id']);
$stmt->bindParam('judul', $dataFromRequest['judul']);
$stmt->bindParam('isi', $dataFromRequest['isi']);
$stmt->execute();

// Memberikan response balik ke request
response(201, [
    'code' => 201,
    'post' => [
        'id' => $pdo->lastInsertId(),
        'user_id' => $token['user_id'],
        'judul' => $dataFromRequest['judul'],
        'isi' => $dataFromRequest['isi'],
    ]
]);
```

Untuk penyobaan API tadi bisa menggunakan postman atau extensi di vscode, tapi kalau misal pengen versi html-nya bisa menjadi pr sendiri buat temen-temen.