Laporan Modul 10: Authentication JWT

Mata Kuliah: Workshop Web Lanjut
Nama: Maila Aziza
NIM: 2024573010024 **Kelas:** TI-2C

Abstrak

Modul ini mengeksplorasi implementasi mekanisme keamanan API menggunakan teknologi JWT (JSON Web Token) dalam platform Laravel. Token-based authentication dipilih karena menawarkan pendekatan stateless yang mengeliminasi kebutuhan penyimpanan data sesi di sisi server, sehingga lebih scalable dan efisien.

Cakupan materi meliputi: proses instalasi dan konfigurasi library JWT terpopuler (tymon/jwt-auth), setup authentication guard untuk API routing, pengembangan middleware custom guna validasi integritas token, dan implementasi lengkap fitur-fitur autentikasi termasuk user registration, authentication credentials verification, session termination, token renewal mechanism, dan profile information retrieval. Praktikum juga mendemonstrasikan verifikasi fungsionalitas melalui testing tools industri (Postman) untuk memastikan setiap endpoint beroperasi sesuai spesifikasi.

1. Dasar Teori

Pengenalan JSON Web Token (JWT)

JWT merupakan standard terbuka (RFC 7519) yang mengatur mekanisme transmisi data secara aman antara dua entitas dalam bentuk JSON Object. Format token berupa rangkaian karakter panjang yang terbagi menjadi tiga komponen utama, masing-masing dipisahkan oleh karakter pemisah titik (.):

Header – Mengandung metadata tentang token, khususnya algoritma hashing yang digunakan dan tipe token itu sendiri.

Payload (Claims) – Bagian yang menyimpan data aktual atau claims yang hendak dikomunikasikan (contoh: identifier user, role/permission, waktu issuance, atau custom data lainnya).

Signature – Hasil komputasi kriptografi dari kombinasi header dan payload menggunakan secret key yang hanya diketahui server. Komponen ini memastikan integritas token dan mencegah modifikasi tanpa otorisasi.

Sebagai contoh, sebuah JWT pada dunia nyata akan terlihat seperti struktur berikut:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJ1c2VyX2lkIjoxLCJyb2xlIjoiYWRtaW4ifQ.
TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

2. Langkah-Langkah Praktikum

Berikut adalah tahapan praktik yang telah dilaksanakan, disertai dengan potongan kode program dan hasil tangkapan layar.

2.1 Praktikum 1 – Implementasi Authentication dengan JWT

Instalasi Package JWT

Pada proyek Laravel yang sudah ada (`laravel-api`), tambahkan package JWT authentication yang paling populer:

```
composer require tymon/jwt-auth
```

Publikasi File Konfigurasi

Setelah instalasi berhasil, publikasikan file konfigurasi package agar dapat disesuaikan dengan kebutuhan proyek:

```
php artisan vendor:publish --
provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
```

Generate Secret Key untuk JWT

Jalankan command untuk membuat secret key yang akan digunakan dalam signing process:

```
php artisan jwt:secret
```

Konfigurasi Authentication Guard

Buka file `config/auth.php` dan ubah konfigurasi guards section:

```
'guards' => [
    'api' => [
        'driver' => 'jwt',
        'provider' => 'users',
    ],
],
```

Modifikasi User Model untuk JWT Integration

Buka file `app/Models/User.php` dan implementasikan JWT Subject interface:

```
use Tymon\JWTAuth\Contracts\JWTSubject;

class User extends Authenticatable implements JWTSubject
{
    /**
     * Retrieve the unique identifier yang akan disimpan dalam JWT payload.
     */
    public function getJWTIdentifier()
    {
        return $this->getKey(); // biasanya user ID
    }

    /**
     * Tambahkan custom claims tambahan ke dalam JWT jika diperlukan.
     */
    public function getJWTCustomClaims()
    {
        return [];
    }
}
```

Pembuatan JWT Middleware Khusus

Generate middleware baru untuk menangani validasi token JWT:

```
php artisan make:middleware JwtMiddleware
```

Implementasi Logika Middleware

Setelah file middleware dibuat, buka `app/Http/Middleware/JwtMiddleware.php` dan isikan logika untuk memvalidasi token:

```
<?php

namespace App\Http\Middleware;

use Closure;
use Tymon\JWTAuth\Facades\JWTAuth;
use Exception;
use Illuminate\Http\Request;

class JwtMiddleware
{
    public function handle(Request $request, Closure $next)
    {
        try {
            JWTAuth::parseToken()->authenticate();
        } catch (Exception $e) {
```

```
            return response()->json(['error' => 'Unauthorized'], 401);
        }

        return $next($request);
    }
}
```

Registrasi Middleware pada Bootstrap Configuration

Buka file `bootstrap/app.php` dan daftarkan middleware custom ke dalam aliasing:

```
<?php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;
use App\Http\Middleware\JwtMiddleware;

return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__ . '/../routes/web.php',
        api: __DIR__ . '/../routes/api.php',
        commands: __DIR__ . '/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function (Middleware $middleware) {
        // Middleware untuk API
        $middleware->api(prepend: [

\Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
        ]);

        // Alias middleware custom
        $middleware->alias([
            'verified' => \App\Http\Middleware\EnsureEmailIsVerified::class,
            'jwt' => JwtMiddleware::class,
        ]);
    })
    ->withExceptions(function (Exceptions $exceptions) {
        //
    })->create();
```

Pengembangan Controller untuk Autentikasi

Generate controller baru untuk menangani semua operasi authentication:

```
php artisan make:controller API\AuthController
```

Implementasikan method-method autentikasi dalam controller:

```php
<?php

namespace App\Http\Controllers\API;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Tymon\JWTAuth\Facades\JWTAuth;
use Tymon\JWTAuth\Exceptions\JWTException;

class AuthController extends Controller
{
    /**
     * Register akun user baru ke dalam sistem
     */
    public function register(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'name'     => 'required|string|max:255',
            'email'    => 'required|email|unique:users',
            'password' => 'required|string|min:6|confirmed',
        ]);

        if ($validator->fails()) {
            return response()->json([
                'status'  => 'error',
                'message' => 'Validation failed',
                'errors'  => $validator->errors()
            ], 422);
        }

        $user = User::create([
            'name'     => $request->name,
            'email'    => $request->email,
            'password' => Hash::make($request->password),
        ]);

        $token = JWTAuth::fromUser($user);

        return response()->json([
            'status'  => 'success',
            'message' => 'User successfully registered',
            'data'    => [
                'user'  => $user,
                'token' => $this->respondWithToken($token),
            ]
        ], 201);
    }
```

```php
    /**
     * Proses login user dengan credential verification
     */
    public function login(Request $request)
    {
        $credentials = $request->only('email', 'password');

        try {
            if (!$token = JWTAuth::attempt($credentials)) {
                return response()->json([
                    'status'  => 'error',
                    'message' => 'Invalid credentials'
                ], 401);
            }
        } catch (JWTException $e) {
            return response()->json([
                'status'  => 'error',
                'message' => 'Could not create token',
                'error'   => $e->getMessage()
            ], 500);
        }

        return response()->json([
            'status'  => 'success',
            'message' => 'Login successful',
            'data'    => $this->respondWithToken($token),
        ]);
    }

    /**
     * Invalidate token dan terminate sesi user
     */
    public function logout()
    {
        try {
            JWTAuth::invalidate(JWTAuth::getToken());

            return response()->json([
                'status'  => 'success',
                'message' => 'User logged out successfully'
            ]);
        } catch (JWTException $e) {
            return response()->json([
                'status'  => 'error',
                'message' => 'Failed to logout, token invalid',
                'error'   => $e->getMessage()
            ], 500);
        }
    }

    /**
     * Refresh token yang sudah ada dengan token baru
     */
    public function refresh()
```

```
    {
        try {
            $newToken = JWTAuth::refresh(JWTAuth::getToken());

            return response()->json([
                'status'  => 'success',
                'message' => 'Token refreshed',
                'data'    => $this->respondWithToken($newToken),
            ]);
        } catch (JWTException $e) {
            return response()->json([
                'status'  => 'error',
                'message' => 'Failed to refresh token',
                'error'   => $e->getMessage()
            ], 401);
        }
    }

    /**
     * Retrieve informasi profil user yang sedang authenticated
     */
    public function me()
    {
        try {
            $user = JWTAuth::parseToken()->authenticate();

            return response()->json([
                'status'  => 'success',
                'message' => 'User profile fetched',
                'data'    => $user
            ]);
        } catch (JWTException $e) {
            return response()->json([
                'status'  => 'error',
                'message' => 'Token is invalid or expired',
                'error'   => $e->getMessage()
            ], 401);
        }
    }

    /**
     * Helper function untuk format respons token
     */
    protected function respondWithToken($token)
    {
        return [
            'access_token' => $token,
            'token_type'   => 'bearer',
            'expires_in'   => JWTAuth::factory()->getTTL() * 60, // dalam detik
        ];
    }
}
```

Konfigurasi Route API

Buka file `routes/api.php` dan definisikan route untuk endpoints autentikasi dan resource yang dilindungi:

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\ProductController;

Route::prefix('auth')->name('auth.')->group(function () {
    Route::post('register', [AuthController::class, 'register'])-
>name('register');
    Route::post('login', [AuthController::class, 'login'])->name('login');

    Route::middleware('jwt')->group(function () {
        Route::post('logout', [AuthController::class, 'logout'])->name('logout');
        Route::get('me', [AuthController::class, 'me'])->name('me');
        Route::post('refresh', [AuthController::class, 'refresh'])-
>name('refresh');
    });
});

Route::middleware('jwt')->group(function () {
    Route::apiResource('products', ProductController::class);
});
```

Pengujian Endpoint Registration

Testing registrasi user baru dengan Postman:

 register

Skenario: Registration Error

 error

Pengujian Endpoint Login

Melakukan request login dengan credentials yang valid:

 login

Pengujian Endpoint Profile dengan Token

Setelah berhasil melakukan login dan mendapatkan token JWT, gunakan token tersebut untuk mengakses endpoint profile. Header request harus mencakup:

```
Authorization: Bearer <token_dari_login>
```

Respons endpoint akan menampilkan detail user profile:

Pengujian Endpoint Logout

Testing proses session termination dengan menginvalidate token JWT. Request menggunakan token yang sama dari login, endpoint akan merespons dengan konfirmasi logout berhasil:

---

3. Kesimpulan

Melalui pengerjaan modul ini, pemahaman komprehensif tentang mekanisme authentication berbasis JWT dalam konteks API development telah diperoleh. JWT terbukti efektif sebagai metode autentikasi karena karakteristik stateless-nya yang memungkinkan skalabilitas tinggi, ringan dalam transmisi data, dan mudah diimplementasikan di berbagai platform klien.

Alur implementasi di Laravel mencakup rangkaian proses: konfigurasi library JWT (tymon/jwt-auth), setup authentication guard untuk routing API, development middleware untuk validasi token integrity, dan pembuatan lengkap endpoint autentikasi meliputi registration, credential verification, session invalidation, token renewal, dan profile retrieval.

Hasil pengujian melalui tools testing industri standar (Postman) mendemonstrasikan bahwa semua endpoint berfungsi dengan baik tanpa adanya anomali. Dari tahap registration hingga logout, sistem token-based authentication beroperasi sesuai spesifikasi desain. Secara keseluruhan, JWT menyediakan solusi authentication yang robust, secure, dan flexible untuk mendukung pengembangan API application modern yang scalable.

---

4. Referensi

Cantumkan sumber yang Anda baca (buku, artikel, dokumentasi) — minimal 2 sumber. Gunakan format sederhana (judul — URL).

Tutorial Laravel 12 RESTful API — https://lagikoding.com/episode/tutorial-laravel-12-restful-api-1-install-laravel-12 Tutorial Laravel Rest API Untuk Pemula — https://www.rumahweb.com/journal/tutorial-laravel-rest-api/
