Aplikacje internetowe 1 (AI1) – laboratorium nr 13

Laravel, JWT – uwierzytelnianie i autoryzacja

Początek laboratorium:

- w XAMPP uruchomić Apache oraz MySQL, następnie przejść do phpMyAdmin,
- pobrać na pulpit archiwum Lab013_AI1_start.zip, w którym umieszczony jest projekt startowy do wykonania zadań oraz rozpakować to archiwum,
- przejść do rozpakowanego folderu oraz w przypadku posiadania <u>innych ustawień</u> niż domyślne (np. połączenia z bazą), wykonać ich zmianę w .env.example oraz start...,
- uruchomić skrypt <u>start.bat</u> (Windows, 2x kliknięciem) lub <u>start.sh</u> (inne systemy, przez polecenie bash start.sh),
- zainstalować program Postman oraz pominąć zakładanie konta: lightweight API client.

Zadania (Laravel, JWT):

Zadanie 13.1:

Zapoznać się z następującymi zagadnieniami:

- JSON Web Tokens,
- jak wymawiać JWT? (ang. /dzpt/ = jak wyraz "jot"),
- budowa tokena, trzy sekcje, zawartość sekcji,
- · dekodowanie tokena, dostępne narzędzia do dekodowania,
- · weryfikowanie tokena, secret key,
- · uwierzytelnianie żądań z użyciem tokena,
- pakiet PHP-Open-Source-Saver/jwt-auth,
- · instalacja i konfiguracja pakietu,
- inne zagadnienia: unieważnienie tokena, odświeżenie tokena, ustalenie czasu wygaśniecia tokena.

https://jwt.io/introduction

https://datatracker.ietf.org/doc/html/rfc7519#section-1

https://www.digitalocean.com/community/tutorials/the-anatomy-of-a-json-web-token

https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true)

https://iwt.io

https://dinochiesa.github.io/jwt

https://github.com/PHP-Open-Source-Saver/jwt-auth

https://laravel-jwt-auth.readthedocs.io/en/latest

https://laravel-jwt-auth.readthedocs.io/en/latest/laravel-installation

https://laravel-jwt-auth.readthedocs.io/en/latest/quick-start

https://laravel-jwt-auth.readthedocs.io/en/latest/auth-guard

Zadanie 13.2:

Otworzyć terminal cmd (Command Prompt) w VSCode.

Uruchomić serwer deweloperski php dla przy użyciu komendy serve artisan'a.

W przeglądarce przejść do Telescope oraz pozostawić go otwartym.

php artisan serve

http://localhost:8000/telescope/requests

Zadanie 13.3:

Na potrzeby tego laboratorium został już zainstalowany i skonfigurowany pakiet https://github.com/PHP-Open-Source-Saver/jwt-auth (fork tymondesigns/jwt-auth). Otworzyć drugą kartę terminala cmd w VSCode i wykonać poniższą komendę:

https://laravel-jwt-auth.readthedocs.io/en/latest/laravel-installation https://laravel-jwt-auth.readthedocs.io/en/latest/quick-start/ https://laracoding.com/exclude-middleware-from-route-or-method/

php artisan jwt:secret

Zadanie 13.4:

Wykonać przez *Postman'a* następujące żądanie pod *endpoint "logowania"* użytkowników, z ciałem żądania zawierającym *JSON*, w którym:

- · podać nieprawidłowe dane,
- podać dane Jana (email: jan@email.com, hasło: 1234).

Zwrócić uwagę na odpowiedź (kod odpowiedzi, zawartość ciała odpowiedzi).

Następnie wykorzystać token Jana i wykonać żądanie pod endpoint zwracający dane uwierzytelniającego się użytkownika. (Authorization -> Bearer Token -> wkleić token to pola Token)

POST http://localhost:8000/api/auth/login

```
{
    "email": "user@email.com",
    "password": "pass"
}

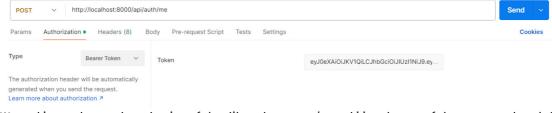
Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON > 

Wrap Line

**access_token*: "eyJBeXA101JKY1Q1LCJhbGcxxxxxxxxxxx139.
    eyJpc3M10130dHRw018vb69jYwxx0b3N60jgwM0xxXXxxxxxxx139.
    eyJpc3M10130dHRw018vb69jYwxx0b3N60jgwM0xxXB1ZF3dGgybG9naM41LCJpYXQ10jE3MTU4NTQxMDgsImV4cC16MTcxNTg1Nzcw0CwibmJmIjoxNzE10DU6MTA4LCJqd
    Gk101JTR0ZtZFV2eVR6MTZxb2NC1iwic3ViIjoiMSIsInByd161jIzYmQ1Yzg5NDlmNjAwYWRiMz1NzAYYzQwMDg3MmRiN2E10Tc2Zjc1fQ.
    Lobzyy80c4AkhkzYDU1VBMSNEh42KxmGIMPR_oJlw8-k",
    **token_type*: "bearer",
    expires_in*: 3680
```

POST http://localhost:8000/api/auth/me



Wszystkie uzyskane tokeny kopiować do pliku tekstowego/notatki i zachowywać do następnych zadań.

Zadanie 13.5:

Zdekodować token i odczytać zawartość dwóch pierwszych sekcji tokena:

- · z wykorzystaniem strony CyberChef,
- z wykorzystaniem strony jwt.io, dinochiesa.github.io.
- Z końca pliku .env odczytać JWT_SECRET, którym następnie wykorzystać do zweryfikowania prawdziwości tokena (czy zawartość dwóch pierwszych sekcji nie została zmodyfikowana).

https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true)
https://jwt.io/

HS256

https://dinochiesa.github.io/jwt/

Zadanie 13.6:

Zastosować użycie middleware'a jwt:auth dla CountryController.

Sprawdzić czy można pobrać wszystkie kraje bez uwierzytelnienia się tokenem.

Następnie wykorzystać token Jana.

Route::apiResource('countries', CountryController::class)->middleware('jwt.auth');

GET http://localhost:8000/api/countries

Zadanie 13.7:

Sprawdzić w Telescope szczegóły poprzednio wykonanych żądań (Authenticated User).

http://localhost:8000/telescope/requests

Zadanie 13.8:

Uzupełnić funkcję JWTCustomClaims modelu User w celu dodawania do payload'u tokena roli użytkownika.

Uzupełnić funkcję *authorize* w *UpdateCountryRequest*, tak aby zezwolenie na edycję wycieczek przysługiwało tylko *administratorom* (sprawdzenie dokonuje odczytywania roli z *tokena*, czyli nie z bazy danych).

```
use Illuminate\Support\Facades\DB;
public function getJWTCustomClaims()
{
    return ['role' => DB::table('roles')->find($this->role_id)->name];
}
use Illuminate\Support\Facades\Auth;
public function authorize()
{
    return Auth::payload()->get('role') == 'admin';
}
```

Zadanie 13.9:

Sprawdzić działanie *autoryzacji* ustalonej w poprzednim zadaniu poprzez chęć aktualizacji danych kraju np. *Polska*:

- bez uwierzytelnienia się tokenem,
- użyciem tokena użytkownika nie będącego administratorem (Marta),
- użyciem tokena administratora Jana, uzyskanym przed zadaniem 13.8,

• użyciem nowego tokena administratora Jana (uzyskać go teraz).

Po utworzeniu nowego tokena sprawdzić zawartość jego payload'u.

Zwrócić uwagę na odpowiedź (kod statusu, zawartość ciała odpowiedzi).

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/403

Zadanie 13.10:

Sprawdzić działanie funkcjonalności "wylogowania się" użytkownika poprzez wykonanie żądania pod odpowiedni endpoint z użyciem ważnego tokena.

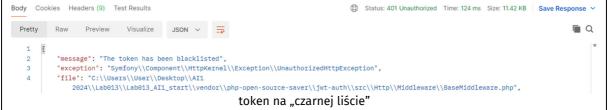
Następnie sprawdzić zawartość tabeli cache (kolumna key zawiera JWT ID).

Następnie pobrać wszystkie kraje z użyciem tego tokena, który jest teraz na "czarnej liście" (podejście twórców pakietu jwt-auth).

POST http://localhost:8000/api/auth/logout



http://localhost/phpmyadmin/index.php?db=ai1_lab13&table=cache



https://koddlo.pl/jwt-a-mialo-byc-bezstanowo/

Zadanie 13.11:

Sprawdzić działanie funkcjonalności *odświeżania tokena* poprzez wykonanie żądania pod odpowiedni *endpoint* z użyciem ważnego *tokena*.

Następnie pobrać wszystkie kraje z:

- · użyciem nowego tokena uzyskanego po odświeżeniu,
- · użyciem starego tokena.

POST http://localhost:8000/api/auth/refresh

Zadania (Laravel, JWT, cd.):

Zadanie 13.12: *

Uzupełnić funkcję boot w app\Providers\AppServiceProvider.php o nową bramkę o nazwie is-admin sprawdzającą czy użytkownik jest administratorem.

Zastosować powyższą *bramkę* tak, aby tylko administratorzy mogli <u>usuwać</u> kraje.

Sprawdzić usuwanie krajów:

- bez uwierzytelnienia się tokenem,
- z użyciem tokena użytkownika nie będącego administratorem (Marta),
- z użyciem tokena administratora Jana.

```
use Illuminate\Support\Facades\Gate;
use Illuminate\Support\Facades\Auth;

Gate::define('is-admin', function () {
    return Auth::payload()->get('role') == 'admin';
});
```

Zadanie 13.13: *

Wygenerować plik z polityką uprawnień dla modelu Trip.

Ustawić działanie wybranych funkcji pliku app\Policies\TripPolicy tak, aby zezwolenie na:

- pobranie wszystkich wycieczek było możliwe tylko dla "zalogowanych użytkowników" (dołączających prawidłowy token do żądania),
- usuwanie wycieczki było możliwe tylko dla administratorów.

Sprawdzić powyższe operacje CRUD z poziomu różnych użytkowników.

Zastosować politykę w odpowiednich funkcjach kontrolera z wykorzystaniem fasady: Gate.

```
php artisan make:policy TripPolicy --model=Trip
use Illuminate\Support\Facades\Gate;
Gate::authorize(..., ...);
```

Zadanie 13.14: *

Zmienić ustalenie czasu ważności tokenów na jedną minutę (w config/jwt.php).

Uzyskać nowy token dla Jana. Sprawdzić zawartość jego payload'u.

Następnie po minucie pobrać wszystkie kraje z:

- użyciem tokena uzyskanego po odświeżeniu w zadaniu 13.11,
- użyciem nowego tokena uzyskanego po zmianie czasu ważności.

```
Tittl' => env('JWT_TTL', 1),

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON > 

"message": "Token has expired",

"exception": "Symfony\Component\HttpKernel\Exception\UnauthorizedHttpException",

"file": "C:\Users\User\User\\Desktop\\AI1

2024\\Lab013\\Lab013_AI1_start\vendor\php-open-source-saver\\jwt-auth\\src\\Http\\Middleware\\BaseMiddleware.php",
```

Zadanie 13.15: *

Przywrócić ustalenie ważności tokenów na domyślne (60 minut).

Wykonać następujące kroki:

- · uzyskać token dla administratora Jana,
- zmienić jego rolę na user (role_id na 2) bezpośrednio w phpMyAdminie,
- spróbować zaktualizować dane dowolnego kraju.

Czy aktualizacja się powiodła, pomimo że Jan nie jest już administratorem?



Zadanie 13.16: *

Mając ważny token, wykonać następujące żądania (z poziomu *Postmana*/innego klienta *HTTP*). Wyjaśnić rezultat dla pięciu pierwszych oraz kod odpowiedzi dla ostatniego.

OPTIONS http://localhost:8000/api/countries/1

HEAD http://localhost:8000/api/countries
HEAD http://localhost:8000/api/countries/1
HEAD http://localhost:8000/api/countries/111

PUT http://localhost:8000/api/countries

Zadanie 13.17: *

W projektach zamiast *tokenów JWT* można wykorzystać alternatywnie *tokeny Sanctum*. Ich sposób funkcjonowanie znacznie różni się od *JWT*, a działanie całkiem zależy od ich obecnego stanu w bazie danych.

https://laravel.com/docs/11.x/sanctum
https://laravel.com/docs/11.x/sanctum#installation

- * zadania/podpunkty do samodzielnego dokończenia/wykonania,
- * zadania/podpunkty dla zainteresowanych.

<u>Po zakończonym laboratorium należy skasować wszystkie pobrane oraz utworzone przez siebie pliki z komputera w sali laboratoryjnej.</u>

Wersja pliku: v1.0