

# Laravel – uwierzytelnianie i autoryzacja

### Początek laboratorium:

- w XAMPP uruchomić Apache oraz MySQL, następnie przejść do phpMyAdmin,
- pobrać na pulpit archiwum Lab008\_AI1\_start.zip, w którym umieszczony jest projekt startowy do wykonania zadań oraz rozpakować to archiwum,
- przejść do rozpakowanego folderu oraz w przypadku posiadania innych ustawień niż domyślne (np. połączenia z bazą), wykonać ich zmianę w .env.example oraz start....,
- uruchomić skrypt start.bat (Windows, 2x kliknięciem) lub start.sh (inne systemy, przez polecenie `bash start.sh`).

### Zadania (Laravel):

#### Zadanie 8.1: \*

Wyjaśnić następujące zagadnienia:

- *identyfikacja* (*identification*),
- *uwierzytelnienie* (*authentication*),
- *autoryzacja* (*authorization*),
- jakie są niepoprawne tłumaczenia słowa „*authentication*” na język polski?,
- czy HTTP jest *protokołem bezstanowym* (*stateless protocol*)?,
- aplikacja *stanowa* (*stateful application*),
- sesje (*sessions*), identyfikator sesji (*session id*),
- ciasteczka (*cookies*).

–

#### Zadanie 8.2:

Otworzyć terminal *cmd* (*Command Prompt*) w VSCode.

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

```
php artisan serve
```

W przeglądarce internetowej przejść pod adres: <http://localhost:8000/trips>

#### Zadanie 7.3:

Zapoznać się z następującymi (już przygotowanymi) elementami aplikacji:

- *migracją* dodającą do tabeli *users* nową kolumnę *country\_id* (będącą kluczem obcym z nałożonymi więzami) w celu określenia pochodzenia danej osoby,
- *migracją* dodającą nową tabelę *roles* do przechowywania *ról użytkowników* (kolumny: *id*, *name*) oraz dodającą do tabeli *users* nową kolumnę *role\_id* (będącą kluczem obcym z nałożonymi więzami) w celu określenia roli danej osoby (użytkownika aplikacji),
- *database\seeders\RoleSeeder.php* dotyczący roli użytkowników (dwie role: *admin*, *user*),

- `database\seeders\UserSeeder.php` (tylko pierwszy użytkownik *Jan* jest *administratorem*, reszta zwykłymi użytkownikami),
- `database\seeders\DatabaseSeeder.php` (wywołanie trzech pozostałych *seederów*),
- `routes\web.php` (*routing* dla funkcji *AuthController'a*),
- `app\Http\Controllers\AuthController.php`,
- `resources\views\shared\navbar.blade.php` (linki do podstrony *logowania się*/ operacji *wylogowania się*, widocznymi w zależności od tego czy użytkownik jest obecnie *zalogowany*),
- `resources\views\auth\login.blade.php`,
- `app\Providers\AppServiceProvider.php` (wyłączenie szyfrowania ciasteczek – na potrzeby laboratorium).

<https://laravel.com/docs/11.x/seeding#calling-additional-seeders>  
<https://laravel.com/docs/11.x/authentication#authenticating-users>  
<https://laravel.com/docs/11.x/authentication#logging-out>  
<https://laravel.com/docs/11.x/authentication#determining-if-the-current-user-is-authenticated>  
<https://laravel.com/docs/11.x/authentication#retrieving-the-authenticated-user>  
<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/password>  
[http://localhost/phpmyadmin/index.php?route=/database/structure&db=ai1\\_lab8](http://localhost/phpmyadmin/index.php?route=/database/structure&db=ai1_lab8)

#### Zadanie 8.4:

Sprawdzić funkcjonowanie *logowania się* użytkowników poprzez:

- próbę zalogowania się wpisując jakiekolwiek dane,
- zalogowanie się na dowolnego użytkownika wpisując poprawne dane,
- wylogowanie się klikając w link w *navbarze*,
- przejście na podstronę logowania, gdy użytkownik jest aktualnie zalogowany.

W przeglądarce internetowej przejść pod adres: <http://localhost:8000/auth/login>

#### Zadanie 8.5:

Wylogować się (jeśli jakiś użytkownik jest obecnie zalogowany). Następnie:

- przejść do *phpMyAdmin* w celu sprawdzenia zawartości tabeli *sessions*,
- w innej karcie przeglądarki przejść na podstronę logowania,
- otworzyć *Panel dla programistów (F12)* i przejść do zakładki *Dane/Aplikacja*,
- przejść do *ciasteczek*, rozwinąć szczegóły ciasteczka *laravel\_session*.

Każdorazowo sprawdzając zawartość tabeli i ciasteczka:

- zalogować się na dowolnego użytkownika,
- wylogować się.

[http://localhost/phpmyadmin/index.php?route=/sql&pos=0&db=ai1\\_lab8&table=sessions](http://localhost/phpmyadmin/index.php?route=/sql&pos=0&db=ai1_lab8&table=sessions)

id	user_id	ip_address	user_agent	payload	last_activity
k91iZ8IT5ZxmN1c	NULL	127.0.0.1	Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA...	YTozOntzOjY6Il90b2tI	1713444353

Name	Value	D...	P...	E...	St...	H...	S...	P...	P...
XSRF-TOKEN	mP4X0TG0UUXjh3...	I...	/	2...	50			Lax	M...
laravel_session	k91iZ8IT5ZxmN1c...	I...	/	2...	55	✓		Lax	M...

Cookie Value: ☒ Show URL-decoded  
k91iZ8IT5ZxmN1cyaPeUwWbntWfVdk4KU0fXtLAZ

#### Zadanie 8.6:

Uzupełnić *trasowanie* dla kontrolera *CountryController* o możliwość dostępu do jego funkcjonalności tylko dla *zalogowanych* użytkowników.

Sprawdzić działanie poprzez:

- przejście na podstronę *krajów*, gdy użytkownik jest zalogowany,
- przejście na podstronę *krajów*, gdy użytkownik jest niezalogowany.

Wskazać fragment kodu w projekcie wykonujący zaobserwowane automatyczne przekierowania.

Po sprawdzeniu działania, przywrócić dostęp dla niezalogowanych użytkowników.

<https://laravel.com/docs/11.x/authentication#protecting-routes>

```
->middleware('auth');
```

W przeglądarce internetowej przejść pod adres: <http://localhost:8000/countries>

#### Zadanie 8.7:

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

<https://laravel.com/docs/11.x/authorization#gates>

```
use Illuminate\Support\Facades\Gate;
use App\Models\User;
```

```
Gate::define('is-admin', function (User $user) {
    return $user->role_id == 1;
});
```

#### Zadanie 8.8:

Zastosować powyższą *bramkę* w *resources\views\trips\index.blade.php* tak, aby linki do edycji wycieczek (z ostatniej kolumny tabelki) były widoczne tylko dla *zalogowanego administratora*. Sprawdzić działanie poprzez:

- przejście na podstronę *krajów*, gdy użytkownik jest niezalogowany,
- przejście na podstronę *krajów*, gdy administrator jest zalogowany,
- przejście na podstronę *krajów*, gdy użytkownik jest zalogowany.

Wyjaśnić czy powyższe zastosowanie jest skuteczne w celu uniemożliwienia operacji edycji innym użytkownikom niż *administratorzy*. Jeśli nie jest, to zaproponować skuteczne rozwiązanie.

<https://laravel.com/docs/11.x/authorization#via-blade-templates>

```
@can('is-admin')
    <!-- ... -->
@endcan
```

W przeglądarce internetowej przejść pod adres: <http://localhost:8000/trips>

<https://laravel.com/docs/11.x/authorization#authorizing-actions-via-gates>

```
if (! Gate::allows('is-admin')) {
    abort(403);
}
```

#### Zadanie 8.9:

Otworzyć drugą kartę terminala *cmd* (*Command Prompt*) w *VSCode*.

Wygenerować plik z *polityką uprawnień* dla modelu *Country*. Manualnie zarejestrować *politykę* w *app\Http\Providers\AppServiceProvider.php*.

<https://laravel.com/docs/11.x/authorization#generating-policies>

```
php artisan make:policy CountryPolicy --model=Country
```

<https://laravel.com/docs/11.x/authorization#manually-registering-policies>

```
Gate::policy(Country::class, CountryPolicy::class);
```

#### Zadanie 8.10:

Przejsć do *app\Policies\CountryPolicy.php*.

We wszystkich funkcjach ustawić zwracaną wartość na `true`.

Ustawić działanie funkcji *update* tak, aby zezwolenie na aktualizację kraju było tylko dla tego użytkownika, który pochodzi z tego kraju.

<https://laravel.com/docs/11.x/authorization#policy-methods>

```
public function update(User $user, Country $country): bool
{
    return $user->country_id === $country->id;
}
```

#### Zadanie 8.11:

Zastosować *politykę* z poprzedniego zadania w funkcji *update* w *CountryController* na dwa sposoby:

- z *if-em* z *cannot*,
- z wykorzystaniem *fasady: Gate*.

Sprawdzić edycję różnych krajów z poziomu różnych użytkowników.

<https://laravel.com/docs/11.x/authorization#via-the-user-model>

```
if ($request->user()->cannot('update', $country)) {
    abort(403);
}
```

<https://laravel.com/docs/11.x/authorization#via-the-gate-facade>

```
Gate::authorize('update', $country);
```

#### Zadanie 8.12:

W *resources\views\countries\index.blade.php* ukryć linki do edycji *krajów* dla tych użytkowników, którzy nie mają możliwości ich edycji.

<https://laravel.com/docs/11.x/authorization#via-blade-templates>

```
@can('update', $country)
    {{-- ... --}}
@endcan
```

Wycieczki górskie Wycieczki Kraje						Jan, wyloguj się...
Kraje						
<a href="#">Dodaj nowy kraj</a>						
#	Nazwa	Kod	Waluta	Powierzchnia	Język	
1	USA	US	dolar amerykański	9833520 km <sup>2</sup>	angielski	<a href="#">Usuń</a>
2	Chiny	CN	yuan	9596960 km <sup>2</sup>	mandaryński	<a href="#">Usuń</a>
3	Austria	AT	euro	83879 km <sup>2</sup>	niemiecki	<a href="#">Usuń</a>
4	Tanzania	TZ	szyling tanzański	947300 km <sup>2</sup>	suahili	<a href="#">Usuń</a>
5	Polska	PL	złoty	38179800 km <sup>2</sup>	polski <a href="#">Edycja</a>	<a href="#">Usuń</a>
6	Australia	AU	dolar australijski	7686850 km <sup>2</sup>	angielski	<a href="#">Usuń</a>
© Wycieczki górskie – 2024						

#### Zadanie 8.13:

Sprawdzić czy mechanizm „*autoodkrywania*” polityk w odpowiednio nazwanych plikach działa automatycznie, poprzez zakomentowanie rejestracji polityki *CountryPolicy* w *app\Http\Providers\AppServiceProvider*.

<https://laravel.com/docs/11.x/authorization#policy-discovery>

#### Zadanie 8.14:

Dodać funkcję *isAdmin()* w modelu *User*, tak aby można było sprawdzić, czy użytkownik jest *adminem*.

Użyć ją w bramce *is-admin*.

```
public function isAdmin() : bool {
    return $this->role_id == DB::table('roles')->where('name', 'admin')->value('id');
}
```

#### Zadanie 8.15:

Dodać nową *bramkę* typu *before* pozwalającą administratorowi na wykonanie dowolnej akcji chronionej innymi *bramkami*, *politykami*.

Później zakomentować tą *bramkę*.

<https://laravel.com/docs/11.x/authorization#intercepting-gate-checks>

```
Gate::before(function (User $user, string $ability) {
    return $user->isAdmin();
});
```

## Zadania (Laravel c.d.):

#### Zadanie 8.16: \*

Utworzyć formularz *rejestracji* użytkowników oraz zaprogramować możliwość *rejestracji* nowego użytkownika. Zrealizować potwierdzanie hasła w drugim polu formularza.

Wykorzystać *AuthController* i dodać w nim nowe funkcje.

Dodać link do podstrony z rejestracją do *navbara*. Ma być widoczny, gdy użytkownik nie jest *zalogowany*.

```
php -r "touch('resources\views\auth\register.blade.php');"
```

<https://laravel.com/docs/11.x/validation#rule-confirmed>

**Zadanie 8.17: \***


Temat „Bezpieczne przechowywanie haseł w bazie danych”. Rozważyć kwestie:

- z czego wynika ta potrzeba?,
- przechowywanie haseł *plain-textem*,
- operacja *hashowania*, *hashowanie* a *szyfrowanie*,
- formy ataków: *łamanie haseł*, *metoda siłowa*, *metoda słownikowa*, *tablice tęczowe*,
- dodanie *sol*/*pieprzu* do hasła,
- algorytmy *hashowania* i ich współczesne bezpieczeństwo: *MD5*, *SHA...*, *bcrypt*,
- *hashowanie* haseł w *PHP* i frameworku *Laravel*,
- tworzenie hasła w *database\seeders\UserSeeder.php*.

–

**Zadanie 8.18: \***

Temat „Bezpieczeństwo danych i haseł z perspektywy użytkownika”. Rozważyć kwestie:

- *HTTPS*, obecność  w pasku adresu URL,
- „silne” hasła,
- przechowywanie haseł w przeglądarce internetowej,
- menedżery haseł: lokalne (np. *KeePassXC*), chmurowe,
- uwierzytelnianie wieloskładnikowe (*multi-factor authentication*, *MFA*),
- uwierzytelnianie dwuskładnikowe (*two-factor authentication*, *2FA*),
- ograniczone czasowe hasło jednorazowe (*TOTP*),
- klucz zabezpieczający *U2F* (np. *YubiKey*),
- ataki na konta użytkowników (*keylogger’y*, przechwytywanie sesji, kradzież ciasteczek, *malware*, *phishing*).

–

**Zadanie 8.19: \***

Utworzyć nowy projekt wykorzystując pakiet startowy *Laravel Breeze*. Zapoznać się z dostarczonymi składnikami m.in.: *resetowanie hasła*, *potwierdzaniem adresu email*, *weryfikacją hasła*.

<https://laravel.com/docs/11.x/starter-kits#laravel-breeze>

\* – zadania/podpunkty do samodzielnego dokończenia/wykonania,

\* – zadania/podpunkty dla zainteresowanych.

Do spakowania projektu należy wykorzystać skrypt *archiwizacja.bat*. Po rozpakowaniu projektu należy użyć ponownie *start.bat*.

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