|  |  |  |
| --- | --- | --- |
| **AI2** | Bancewicz Aleksandra  Album ba34753  Grupa 1 | **Wersja 1** |
|  |
| **LAB E** |

Security

# Spis treści

[Spis treści 1](#_Toc181341707)

[Cel zajęć 1](#_Toc181341708)

[Rozpoczęcie 1](#_Toc181341709)

[Uwaga 1](#_Toc181341710)

[Zabezpieczenie dostępu do akcji i widoków 2](#_Toc181341711)

[Proste logowanie Basic Auth 6](#_Toc181341712)

[Użytkownicy bazodanowi 9](#_Toc181341713)

[Formularz logowania 11](#_Toc181341714)

[Wylogowywanie 14](#_Toc181341715)

[Hierarchia ról 16](#_Toc181341716)

[Commit projektu do GIT 18](#_Toc181341717)

[Podsumowanie 18](#_Toc181341718)

# Cel zajęć

Celem głównym zajęć jest zdobycie umiejętności implementacji i konfiguracji logowania i wylogowywania użytkowników oraz ograniczania dostępu do akcji.

# Rozpoczęcie

Rozpoczęcie zajęć. Powtórzenie security is\_granted w Twig i atrybutach kontrolerów.

Wejściówka?

# Uwaga

Ten dokument aktywnie wykorzystuje niestandardowe właściwości. Podobnie jak w LAB A wejdź do Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe i zaktualizuj pola. Następnie uruchom ten dokument ponownie lub Ctrl+A -> F9.

# Zabezpieczenie dostępu do akcji i widoków

Obecnie dostęp do wszystkich akcji w systemie jest publicznie otwarty. W tej sekcji zabezpieczymy dostęp. Otwórz plik src/Controller/LocationController.php i edytuj akcję new:

#[Route('/new', name: 'app\_location\_new', methods: ['GET', 'POST'])]

#[IsGranted('ROLE\_LOCATION\_NEW')]

public function new(Request $request, EntityManagerInterface $entityManager): Response

{

$location = new Location();

$form = $this->createForm(LocationType::class, $location, [

'validation\_groups' => 'create',

]);

$form->handleRequest($request);

Odwiedź teraz w przeglądarce stronę http://pogodynka.localhost:ba34753/location. Powinna wyświetlić się lista lokalizacji. Następnie kliknij w przycisk tworzenia nowej lokalizacji. Powinien wyświetlić się błąd:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Dlaczego lista lokalizacji wciąż działa, a akcja tworzenia lokalizacji wyrzuca wyjątek?

Lista lokalizacji jest dostępna, ponieważ nie ma na niej żadnego ograniczenia dostępu – akcja wyświetlania listy jest otwarta dla wszystkich użytkowników. Natomiast akcja tworzenia lokalizacji jest zabezpieczona adnotacją #[IsGranted('ROLE\_LOCATION\_NEW')], która wymaga przypisanej roli ROLE\_LOCATION\_NEW u użytkownika. Bez tej roli użytkownik nie ma uprawnień do wykonania tej akcji, co skutkuje wyjątkiem dostępu

Teraz zmodyfikuj widok listy lokalizacji w pliku templates/location/index.html.twig:

Obraz zawierający tekst, Czcionka, zrzut ekranu

Opis wygenerowany automatycznie

Ponownie wejdź w przeglądarce na listę lokalizacji. Przycisk tworzenia nowej lokalizacji powinien teraz zniknąć.

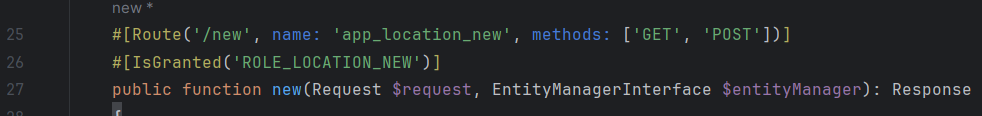
Dodaj wymuszanie posiadania przez użytkownika roli dla wszystkich akcji kontrolerów lokalizacji i pomiarów. Zabronione jest korzystanie z roli typu ROLE\_ADMIN. Zamiast tego utwórz dla każdej akcji w każdym kontrolerze osobną rolę odnoszącą się do czynności, w postaci ROLE\_<kontroler>\_<akcja>, np. ROLE\_LOCATION\_INDEX.

Poniżej wstaw zrzuty ekranu kodu atrybutów i sygnatury każdej akcji każdego kontrolera:

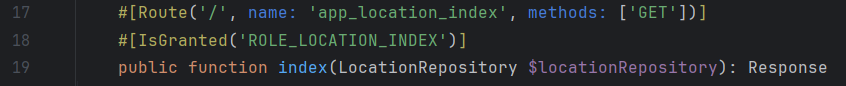
LocationController::new():

Obraz zawierający tekst, Czcionka, zrzut ekranu, linia

Opis wygenerowany automatycznie



LocationController::index():

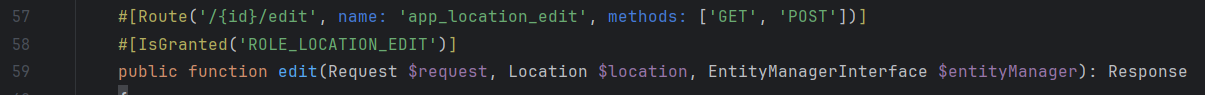


LocationController::show():

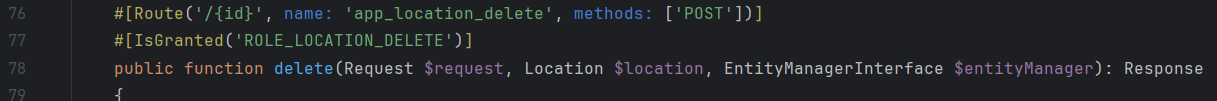
Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

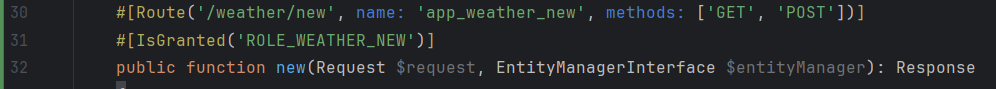
LocationController::edit():



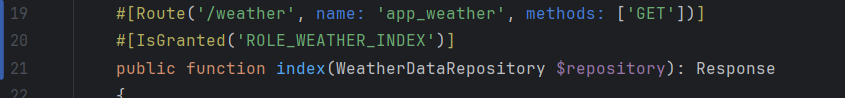
LocationController::delete():



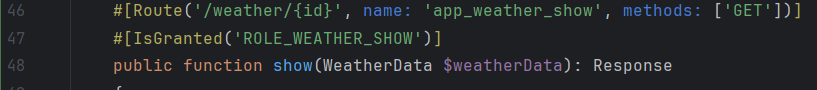
MeasurementController::new():



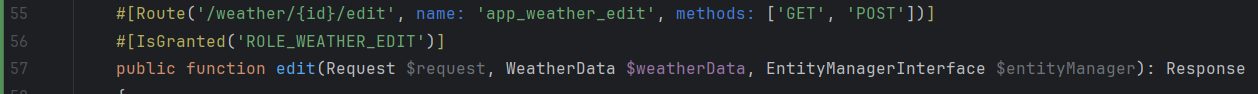
MeasurementController::index():



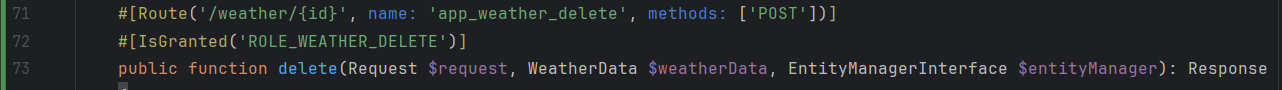
MeasurementController::show():



MeasurementController::edit():



MeasurementController::delete():



|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

Dodaj weryfikację ról w szablonach TWIG na liście lokalizacji i liście pomiarów.

Wstaw zrzut ekranu kodu pliku templates/location/index.html.twig:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Wstaw zrzut ekranu kodu pliku templates/measurement/index.html.twig:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Proste logowanie Basic Auth

W tej sekcji dodamy możliwość logowania użytkownika poprzez Basic Auth (login i hasło podawane bezpośrednio w adresie URL – http://login:pass@pogodynka.localhost:ba34753 – lub w wyświetlonym przez przeglądarkę okienku).

Modyfikuj plik config/packages/security.yaml:

Obraz zawierający tekst, zrzut ekranu, Czcionka, dokument

Opis wygenerowany automatycznie

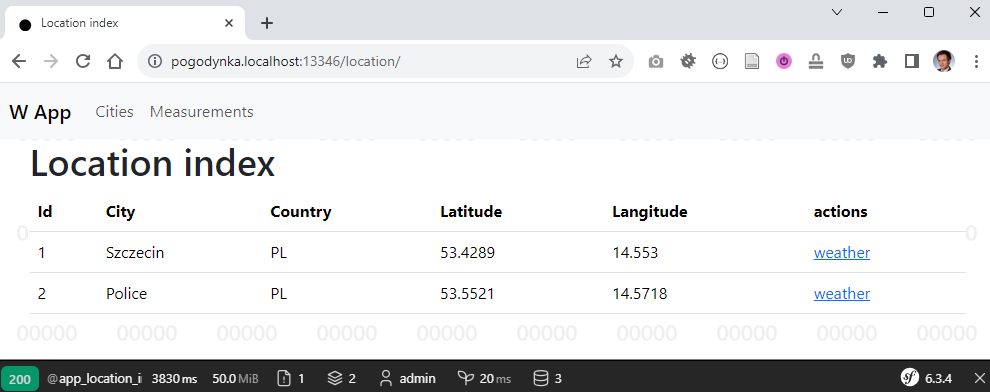
W efekcie po wejściu na stronę powinno wyświetlić się okienko logowania Basic Auth. **Zrób zrzut ekranu! Po wpisaniu poprawnego loginu i hasła to okienko się nie pojawi ponownie!**

Po poprawnym logowaniu wyświetlony zostanie jednak błąd braku dostępu:

Obraz zawierający tekst, zrzut ekranu, Czcionka, Strona internetowa

Opis wygenerowany automatycznie

Wynika to z faktu, że obecnie użytkownik admin ma tylko role ROLE\_ADMIN i ROLE\_USER, a wymagana jest rola ROLE\_LOCATION\_INDEX. W dalszej części tego laboratorium rozwiążemy ten problem. Na ten moment można tymczasowo zmienić wymaganą rolę w LocationController::index() na ROLE\_ADMIN, w celu weryfikacji poprawności działania:



Zwróć uwagę, że chociaż lista się pojawiła, wszystkie linki zabezpieczone rolami ROLE\_<kontroler>\_<akcja> są ukryte.

Wstaw zrzut ekranu wyskakującego okienka Basic Auth na tle Twojej strony:

**Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie**

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

W dalszej części wykorzystamy komendę security:hash-password do zaszyfrowania hasła przed umieszczeniem go w pliku konfiguracyjnym:

php .\bin\console security:hash-password

Symfony Password Hash Utility

=============================

Type in your password to be hashed:

>

! [NOTE] The command will take care of generating a salt for you. Be aware that some hashers advise to let them

generate their own salt. If you're using one of those hashers, please answer 'no' to the question below.

! Provide the 'empty-salt' option in order to let the hasher handle the generation itself.

Confirm salt generation ? (yes/no) [yes]:

> no

--------------- -----------------------------------------------------------------

Key Value

--------------- -----------------------------------------------------------------

Hasher used Symfony\Component\PasswordHasher\Hasher\PlaintextPasswordHasher

Password hash 12345678

--------------- -----------------------------------------------------------------

[OK] Password hashing succeeded

Zwróć uwagę, że zaszyfrowane hasło to wciąż plaintext. Wynika to z konfiguracji w pliku security.yaml. Zmień w nim opcję plaintext z powrotem na auto i ponów hashowanie. Wygenerowany hash podstaw jako hasło użytkownika w pliku security.yaml.

Wstaw zrzut ekranu konfiguracji security.yaml z hasłem w postaci uzyskanego hasha:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie  
  
$2y$13$HYQ8NfxDbrG0tD8DHn5e1.7Ic3mAXd0C2IT8yD2MrPuueYRCCRh4S

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Użytkownicy bazodanowi

W tej sekcji usuniemy użytkownika z security.yaml na rzecz użytkowników przechowywanych w bazie danych.

Wykonaj komendę make:user:

php .\bin\console make:user

The name of the security user class (e.g. User) [User]:

>

Do you want to store user data in the database (via Doctrine)? (yes/no) [yes]:

>

Enter a property name that will be the unique "display" name for the user (e.g. email, username, uuid) [email]:

> username

Will this app need to hash/check user passwords? Choose No if passwords are not needed or will be checked/hashed by some other system (e.g. a single sign-on server).

Does this app need to hash/check user passwords? (yes/no) [yes]:

>

created: src/Entity/User.php

created: src/Repository/UserRepository.php

updated: src/Entity/User.php

updated: config/packages/security.yaml

Success!

Next Steps:

- Review your new App\Entity\User class.

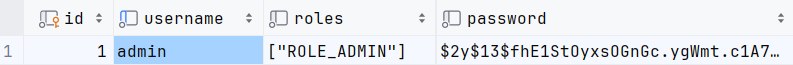
- Use make:entity to add more fields to your User entity and then run make:migration.

- Create a way to authenticate! See https://symfony.com/doc/current/security.html

Zaktualizuj schemat bazy danych:

php bin\console doctrine:schema:update --dump-sql --force

Dodaj co najmniej jednego użytkownika bezpośrednio w bazie danych:



Zaloguj się poprzez Basic Auth na użytkownika, którego dane zapisane są w bazie danych. W pasku profilera kliknij na nazwę użytkownika. Otworzy się panel Security.

Wstaw zrzut ekranu panelu Security z danymi zalogowanego użytkownika:

Wstaw zrzut ekranu podglądu tabeli bazy danych z danymi użytkownika zalogowanego na powyższym zrzucie ekranu:

Obraz zawierający tekst, oprogramowanie, Oprogramowanie multimedialne, zrzut ekranu

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Formularz logowania

W tej sekcji wykorzystamy wbudowany w Symfony mechanizm form\_login do obsługi formularza logowania. Wykorzystaj komendę make:controller do utworzenia kontrolera logowania:

php .\bin\console make:controller Login

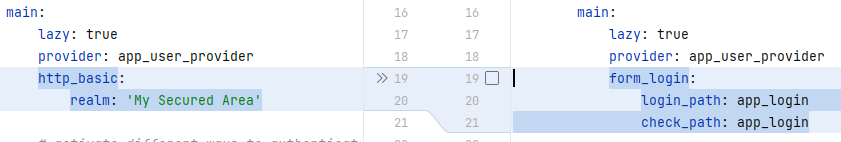
created: src/Controller/LoginController.php

created: templates/login/index.html.twig

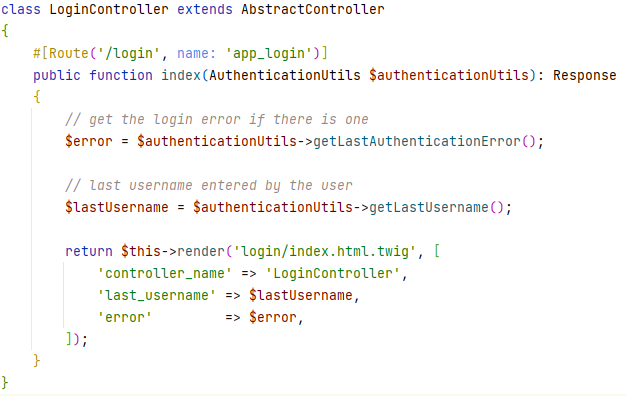
Success!

Next: Open your new controller class and add some pages!

Zaktualizuj ustawienia firewalla main w security.yaml. Zastąp logowanie http\_basic przez form\_login:



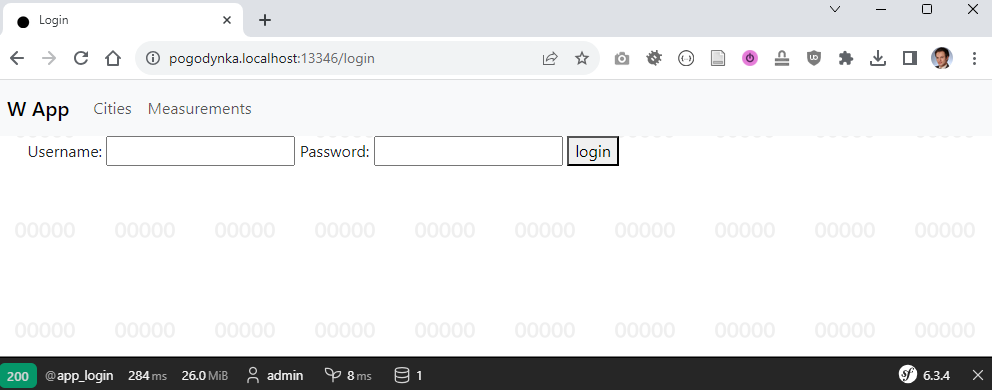
Teraz zmodyfikuj LoginController:



Zmodyfikuj również szablon akcji logowania:



Po wejściu na stronę z ograniczeniem dostępu wyświetli się teraz formularz logowania:



Wstaw zrzut ekranu strony z formularzem logowania:

Obraz zawierający tekst, zrzut ekranu, Czcionka, linia

Opis wygenerowany automatycznie

Podaj błędne dane logowania. Wstaw zrzut ekranu strony logowania z wyświetlonymi informacjami o błędach:

Obraz zawierający tekst, Czcionka, linia, zrzut ekranu

Opis wygenerowany automatycznie

Zaloguj się na użytkownika, którego dane zapisane są w bazie danych. W pasku profilera kliknij na nazwę użytkownika. Otworzy się panel Security.

Wstaw zrzut ekranu panelu Security z danymi zalogowanego użytkownika:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Wylogowywanie

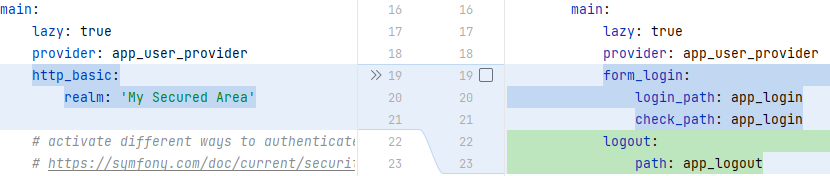
Obecnie po zalogowaniu jedyną opcją wylogowania użytkownika jest skasowanie ciasteczka PHPSESSID. Dodanie logowania w Symfony jest proste. Wystarczy dodać nową ścieżkę do routes.yaml:

app\_logout:

path: /logout

methods: GET

oraz zmodyfikować security.yaml:



Na koniec należy dodać link do logowania / wylogowywania do pliku szablonu base.html.twig:

{% if is\_granted('ROLE\_USER') %}

<a href="{{ path('app\_logout') }}">Logout</a>

{% else %}

<a href="{{ path('app\_login') }}">Login</a>

{% endif %}

Wstaw zrzut ekranu strony z widocznym linkiem wylogowywania:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Wstaw zrzut ekranu strony z widocznym linkiem logowania:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Opis wygenerowany automatycznie

Wstaw zrzut fragmentu kodu TWIG odpowiedzialnego za wyświetlanie linku logowania / wylogowywania:

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie

Wstaw zrzut ekranu konfiguracji security.yaml – fragment dotyczący wylogowywania:

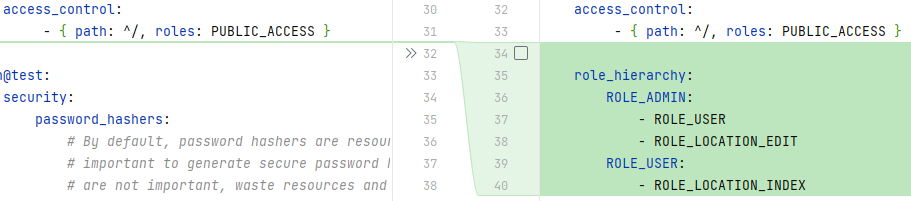
Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

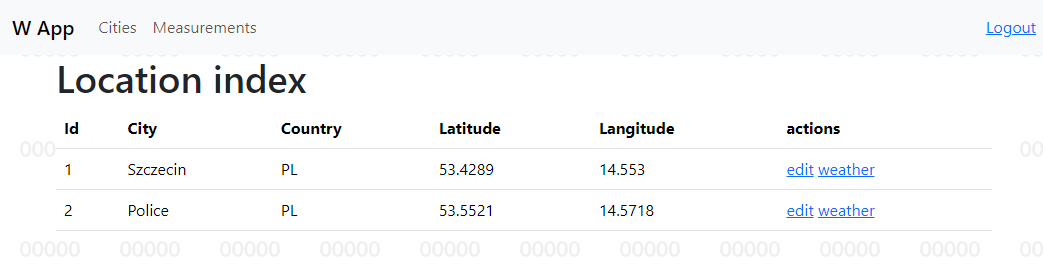
# Hierarchia ról

W tej sekcji zmodyfikujemy ustawienia security.yaml w taki sposób, aby do poszczególnych ról użytkowników (np. ROLE\_ADMIN) przypisać role zachowań (np. ROLE\_LOCATION\_INDEX):



W powyższym przykładzie użytkownik o roli ROLE\_ADMIN może wszystko to co użytkownik o roli ROLE\_USER, a ponadto może edytować lokalizacje. Użytkownik o roli ROLE\_USER może jedynie wyświetlać lokalizacje.

Wprowadź zmiany i wejdź na stronę /location:



Widoczna jest lista lokalizacji, a przy każdej z nich link do edycji. Nie ma natomiast linku do tworzenia nowej lokalizacji. Wejście na strony pomiarów zakończy się wyświetleniem błędu braku dostępu.

Uzupełnij hierarchię ról dla roli ROLE\_ADMIN i dla ROLE\_USER, tak aby obsłużyć wszystkie role obu kontrolerów Location i Measurement. Wstaw zrzut ekranu odpowiedniego fragmentu security.yaml:

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, wyświetlacz

Opis wygenerowany automatycznie

Utwórz nowego użytkownika, z innym zestawem uprawnień (ROLE\_USER? Inna, nowa rola użytkownika?). Wstaw zrzut ekranu listy lokalizacji dla pierwszego użytkownika i dla drugiego użytkownika. Upewnij się, że poziom uprawnień jest różny, przez co różnią się dostępne na stronie akcje:

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Opis wygenerowany automatycznie

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

|  |  |  |
| --- | --- | --- |
| Punkty: | 0 | 1 |

# Commit projektu do GIT

Zacommituj zmiany. Wyślij zmiany do repozytorium (push). Upewnij się, czy wszystko dobrze się wysłało. Jeśli tak, to z poziomu przeglądarki utwórz branch o nazwie lab-e na podstawie głównej gałęzi kodu.

Podaj link do brancha lab-e w swoim repozytorium:

https://github.com/AleksandraBancewicz/AI2/tree/main/LE

# Podsumowanie

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Podczas tego laboratorium zdobyłam umiejętności związane z konfigurowaniem systemu autoryzacji w aplikacji Symfony, w tym tworzeniem i zarządzaniem użytkownikami w bazie danych oraz definiowaniem ról i uprawnień. Nauczyłam się, jak zrealizować logowanie i wylogowywanie użytkowników, a także jak obsługiwać różne poziomy dostępu do zasobów w aplikacji. Dodatkowo zrozumiałam, jak implementować hierarchię ról oraz jak dynamicznie zmieniać interfejs w zależności od uprawnień użytkownika. Używanie narzędzi takich jak php bin/console do generowania i zarządzania konfiguracją oraz debugowania aplikacji również wzbogaciło moje doświadczenie programistyczne.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.