

SECURITY

SPIS TREŚCI

Spis treści	1
Cel zajęć.....	1
Rozpoczęcie	1
Uwaga	1
Zabezpieczenie dostępu do akcji i widoków	2
Proste logowanie Basic Auth	6
Użytkownicy bazodanowi.....	9
Formularz logowania.....	11
Wylogowywanie	14
Hierarchia ról.....	16
Commit projektu do GIT.....	18
Podsumowanie.....	18

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

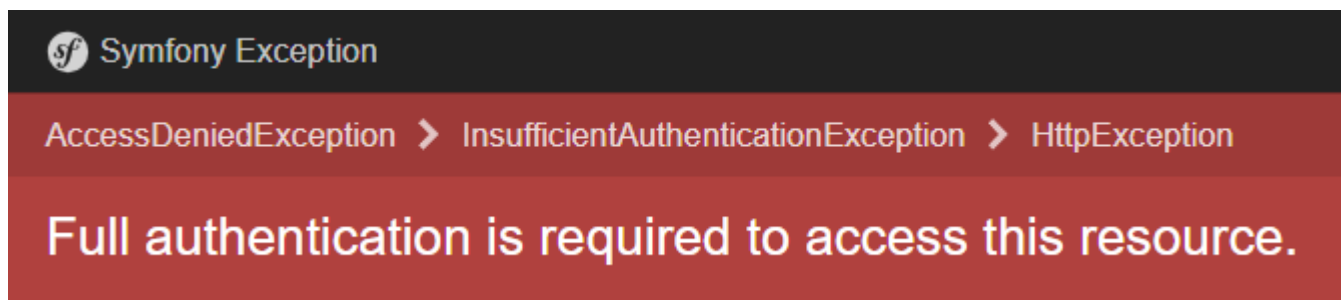
Ten dokument aktywnie wykorzystuje niestandardowe właściwości. Podobnie jak w LAB A wejdź do Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe i zaktualizuj pola. Następnie uruchom ten dokument ponownie lub Ctrl+A -> F9.

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:



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`:

```
--- a/templates/location/index.html.twig
+++ b/templates/location/index.html.twig
@@ -38,5 +38,7 @@
     </tbody>
 </table>

-    <a href="{{ path('app_location_new') }}">Create new</a>
+    {% if is_granted('ROLE_LOCATION_NEW') %}
+    <a href="{{ path('app_location_new') }}">Create new</a>
+    {% endif %}
+{% endblock %}
```

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():

Artur Karczmarczyk

```
#[Route('/new', name: 'app_location_new', methods: ['GET', 'POST'])]
#[IsGranted('ROLE_LOCATION_NEW')]
public function new(Request $request, EntityManagerInterface $entityManager): Response
{
    new *
25     #[Route('/new', name: 'app_location_new', methods: ['GET', 'POST'])]
26     #[IsGranted('ROLE_LOCATION_NEW')]
27     public function new(Request $request, EntityManagerInterface $entityManager): Response
28 }
```

LocationController::index():

```
17     #[Route('/', name: 'app_location_index', methods: ['GET'])]
18     #[IsGranted('ROLE_LOCATION_INDEX')]
19     public function index(LocationRepository $locationRepository): Response
```

LocationController::show():

```
48     #[Route('/{id}', name: 'app_location_show', methods: ['GET'])]
49     #[IsGranted('ROLE_LOCATION_SHOW')]
50     public function show(Location $location): Response
```

LocationController::edit():

```
57     #[Route('/{id}/edit', name: 'app_location_edit', methods: ['GET', 'POST'])]
58     #[IsGranted('ROLE_LOCATION_EDIT')]
59     public function edit(Request $request, Location $location, EntityManagerInterface $entityManager): Response
60 }
```

LocationController::delete():

```
76     #[Route('/{id}', name: 'app_location_delete', methods: ['POST'])]
77     #[IsGranted('ROLE_LOCATION_DELETE')]
78     public function delete(Request $request, Location $location, EntityManagerInterface $entityManager): Response
79 }
```

MeasurementController::new():

```
30     #[Route('/weather/new', name: 'app_weather_new', methods: ['GET', 'POST'])]
31     #[IsGranted('ROLE_WEATHER_NEW')]
32     public function new(Request $request, EntityManagerInterface $entityManager): Response
```

MeasurementController::index():

```

19     #[Route('/weather', name: 'app_weather', methods: ['GET'])]
20     #[IsGranted('ROLE_WEATHER_INDEX')]
21     public function index(WeatherDataRepository $repository): Response
22     {

```

MeasurementController::show():

```

46     #[Route('/weather/{id}', name: 'app_weather_show', methods: ['GET'])]
47     #[IsGranted('ROLE_WEATHER_SHOW')]
48     public function show(WeatherData $weatherData): Response

```

MeasurementController::edit():

```

55     #[Route('/weather/{id}/edit', name: 'app_weather_edit', methods: ['GET', 'POST'])]
56     #[IsGranted('ROLE_WEATHER_EDIT')]
57     public function edit(Request $request, WeatherData $weatherData, EntityManagerInterface $entityManager): Response

```

MeasurementController::delete():

```

71     #[Route('/weather/{id}', name: 'app_weather_delete', methods: ['POST'])]
72     #[IsGranted('ROLE_WEATHER_DELETE')]
73     public function delete(Request $request, WeatherData $weatherData, EntityManagerInterface $entityManager): Response

```

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`:

```

LocationController.php WeatherController.php index.html.twig x city.html.twig
1 {% extends 'base.html.twig' %}
2 {% block title %}Location index{% endblock %}
3 {% block body %}
4     <h1>Location index</h1>
5     <table class="table">
6         <thead>
7             <tr>
8                 <th>Id</th>
9                 <th>City</th>
10                <th>Country</th>
11                <th>Latitude</th>
12                <th>Longitude</th>
13                <th>actions</th>
14            </tr>
15        </thead>
16        <tbody>
17            {% for location in locations %}
18                <tr>
19                    <td>{{ location.id }}</td>
20                    <td>{{ location.city }}</td>
21                    <td>{{ location.country }}</td>
22                    <td>{{ location.latitude }}</td>
23                    <td>{{ location.longitude }}</td>
24                    <td>
25                        <a href="{{ path('app_location_show', {'id': location.id}) }}">show</a>
26                        <a href="{{ path('app_location_edit', {'id': location.id}) }}">edit</a>
27                    </td>
28                </tr>
29            {% else %}
30                <tr>
31                    <td colspan="6">no records found</td>
32                </tr>
33            {% endfor %}
34        </tbody>
35    </table>
36    {% if is_granted('ROLE_LOCATION_NEW') %}
37        <a href="{{ path('app_location_new') }}">Create new</a>
38    {% endif %}
39 {% endblock %}

```

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

```

1  {% extends 'base.html.twig' %}
2
3  {% @var location \App\Entity\Location %}
4  {% @var weatherData \App\Entity\WeatherData[] %}
5
6  {% block title %}Weather in {{ location.city }}, {{ location.country }}{% endblock %}
7
8  {% block body %}
9      <main>
10         <h1>Weather in {{ location.city }}, {{ location.country }}</h1>
11
12         <ul>
13             {% for data in weatherData %}
14                 <li>{{ data.date|date('d.m.Y') }}: {{ data.getTemperatureCelsius() }}&deg;C</li>
15             {% endfor %}
16         </ul>
17
18         {% if is_granted('ROLE_WEATHER_NEW') %}
19             <a href="{{ path('app_weather_new', {'city': location.city, 'countryCode': location.country}) }}">Add new measurement</a>
20         {% endif %}
21     </main>
22 {% endblock %}

```

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`:

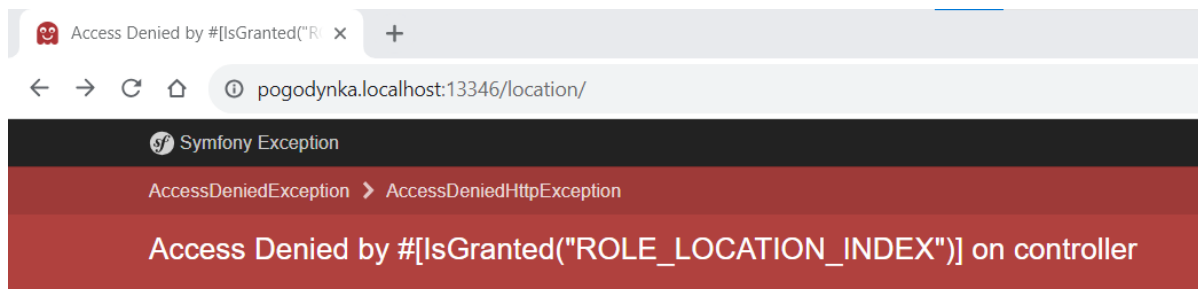
AI2 LAB E – Bancewicz Aleksandra – Wersja 1

```
--- a/config/packages/security.yaml
+++ b/config/packages/security.yaml
@@ -1,10 +1,17 @@
 security:
     # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
     password_hashers:
-        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'auto'
+        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: 'plaintext'
     # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
     providers:
-        users_in_memory: { memory: null }
+        users_in_memory:
+            memory:
+                users:
+                    admin:
+                        password: '12345678'
+                        roles:
+                            - 'ROLE_ADMIN'
+                            - 'ROLE_USER'
     firewalls:
         dev:
             pattern: ^/(_(profiler|wdt)|css|images|js)/
@@ -12,6 +19,8 @@ security:
     main:
         lazy: true
         provider: users_in_memory
+        http_basic:
+            realm: 'My Secured Area'

     # activate different ways to authenticate
     # https://symfony.com/doc/current/security.html#the-firewall
@@ -22,8 +31,7 @@ security:
     # Easy way to control access for large sections of your site
     # Note: Only the *first* access control that matches will be used
     access_control:
-        # - { path: ^/admin, roles: ROLE_ADMIN }
-        # - { path: ^/profile, roles: ROLE_USER }
+        - { path: ^/, roles: PUBLIC_ACCESS }
```

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:



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 działania:

Location index

Id	City	Country	Latitude	Longitude	actions
1	Szczecin	PL	53.4289	14.553	weather
2	Police	PL	53.5521	14.5718	weather

200 @app_location_i 3830 ms 50.0 MiB 1 2 admin 20 ms 3 6.3.4

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:

http://pogodynka.localhost:34753

Nazwa użytkownika

Hasło

Zaloguj się Anuluj

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 $2y$13$027%
-----

[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 hashu:

```
Type in your password to be hashed:
```

```
>
```

```

-----
Key          Value
-----
Hasher used  Symfony\Component\PasswordHasher\Hasher\MigratingPasswordHasher
Password hash $2y$13$qqc.KRHRGNnkaSN5pRIji0GR0VjHnsYPTp6cZrEZ5wq9EjwMDQuS6
-----

```

```
! [NOTE] Self-salting hasher used: the hasher generated its own built-in salt.
```

```

-----
Key          Value
-----
Hasher used  Symfony\Component\PasswordHasher\Hasher\MigratingPasswordHasher
Password hash $2y$13$HYQ8NfxDbrG0tD8DHn5e1.7Ic3mAXd0C2IT8yD2MrPuueYRCCRh4S
-----

```

```
! [NOTE] Self-salting hasher used: the hasher generated its own built-in salt.
```

```
$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:

	id	username	roles	password
1	1	admin	["ROLE_ADMIN"]	\$2y\$13\$fhE1St0yxs0GnGc.ygWmt.c1A7...

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:

	id	username	roles	password
1	1	admin	["ROLE_ADMIN"]	\$2y\$13\$HYQ8NfxDbrG0tD8DHn5e1.7Ic3...

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`:

<pre>main: lazy: true provider: app_user_provider http_basic: realm: 'My Secured Area'</pre>	<pre>16 16 17 17 18 18 >> 19 19 20 20 21 21</pre>	<pre>main: lazy: true provider: app_user_provider form_login: login_path: app_login check_path: app_login</pre>
--	---	---

Teraz zmodyfikuj `LoginController`:

```
class LoginController extends AbstractController
{
    #[Route('/login', name: 'app_login')]
    public function index(AuthenticationUtils $authenticationUtils): Response
    {
        // get the login error if there is one
        $error = $authenticationUtils->getLastAuthenticationError();

        // last username entered by the user
        $lastUsername = $authenticationUtils->getLastUsername();

        return $this->render('login/index.html.twig', [
            'controller_name' => 'LoginController',
            'last_username' => $lastUsername,
            'error' => $error,
        ]);
    }
}
```

Zmodyfikuj również szablon akcji logowania:

```

1  {% extends 'base.html.twig' %}
2
3  {% block title %}Login{% endblock %}
4
5  {% block body %}
6      {% if error %}
7          <div>{{ error.messageKey|trans(error.messageData, 'security') }}</div>
8      {% endif %}
9
10     <form action="{{ path('app_login') }}" method="post">
11         <label for="username">Username:</label>
12         <input type="text" id="username" name="_username" value="{{ last_username }}" />
13
14         <label for="password">Password:</label>
15         <input type="password" id="password" name="_password" />
16
17         {# If you want to control the URL the user is redirected to on success
18         <input type="hidden" name="_target_path" value="/admin" /> #}
19
20         <button type="submit">login</button>
21     </form>
22 {% endblock %}

```

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

W App Cities Measurements

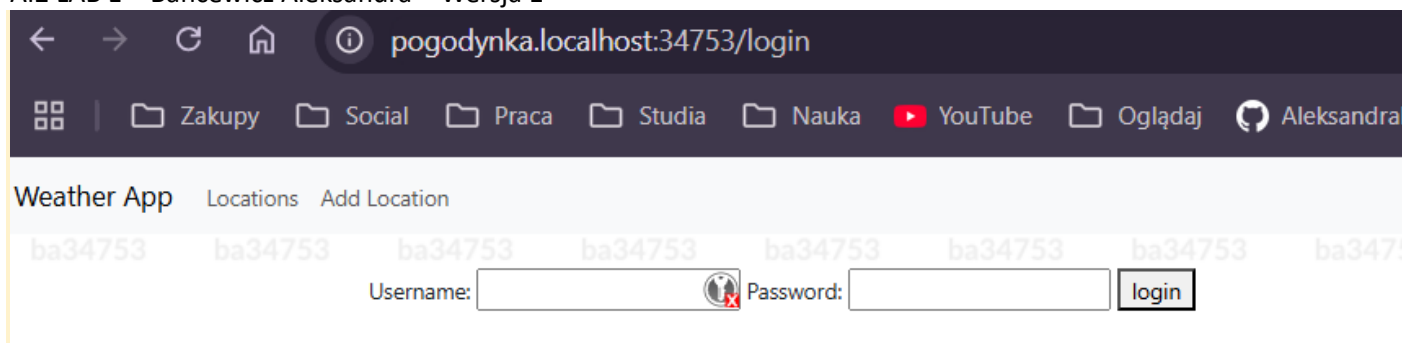
Username: Password:

00000 00000 00000 00000 00000 00000 00000 00000 00000 00000

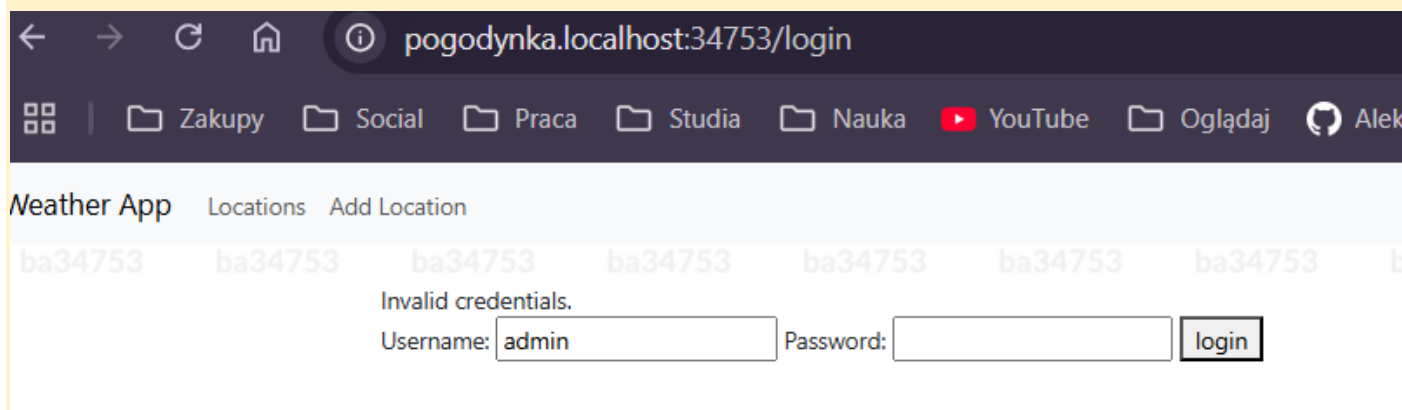
00000 00000 00000 00000 00000 00000 00000 00000 00000 00000

200 @app_login 284 ms 26.0 MiB admin 8 ms 1 6.3.4

Wstaw zrzut ekranu strony z formularzem logowania:



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



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:

hka.localhost:34753/login

referrer URLIP: ::1Profiled on: November 3, 2024 at 9:53:23 PMToken: ad7469

Security

Token

Firewall

Listeners

Authenticators

Access Decision

Username

admin

Authenticated

✓

Property	Value
Roles	[<div>ROLE_ADMINROLE_USER</div>]
Inherited Roles	none
Token	Symfony\Component\Security\Core\Authentication\Token\UsernamePasswordToken {#757 <div></div> <div>-user: App\Entity\User {#892 ...}</div> <div>-roleNames: [▶]</div> <div>-attributes: []</div> <div>-firewallName: "main"</div> }

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
```

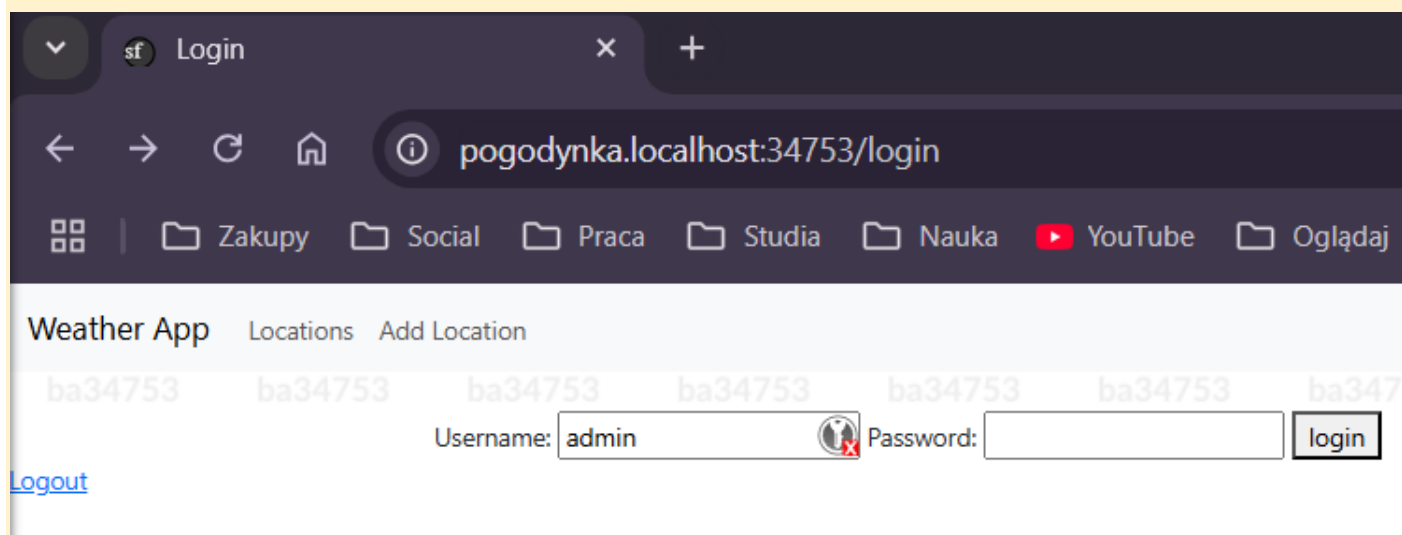
oraz zmodyfikować security.yaml:

main:	16	16	main:
lazy: true	17	17	lazy: true
provider: app_user_provider	18	18	provider: app_user_provider
http_basic:	>> 19	19	form_login:
realm: 'My Secured Area'	20	20	login_path: app_login
	21	21	check_path: app_login
# activate different ways to authenticat	22	22	logout:
# https://symfony.com/doc/current/securi	23	23	path: app_logout

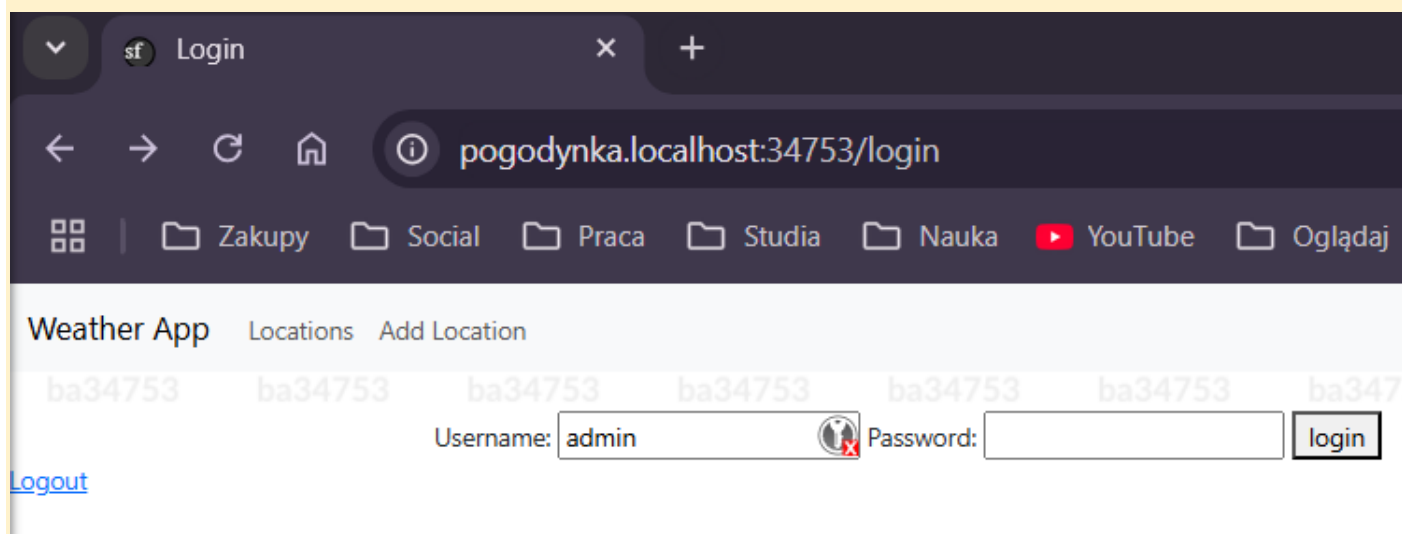
Na koniec należy dodać link do logowania / wylogowywania do pliku szablonu base.html.twig:

```
{% 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:



Wstaw zrzut ekranu strony z widocznym linkiem logowania:



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

```
{% if is_granted('ROLE_USER') %}
    <a href="{{ path('app_logout') }}">Logout</a>
{% else %}
    <a href="{{ path('app_login') }}">Login</a>
{% endif %}
```

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

```
firewalls:
  dev:
    pattern: ^/(_(profiler|wdt)|css|images|js)/
    security:
  main:
    lazy: true
    provider: app_user_provider
    form_login:
      login_path: app_login
      check_path: app_login
    logout:
      path: app_logout
      target: /
```

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):

<pre>access_control: - { path: ^/, roles: PUBLIC_ACCESS }</pre>	<pre>30 31 >> 32 33 34 35 36 37 38</pre>	<pre>access_control: - { path: ^/, roles: PUBLIC_ACCESS }</pre>
<pre>@test: security: password_hashers: # By default, password hashers are resour # important to generate secure password l # are not important, waste resources and</pre>	<pre>32 33 34 35 36 37 38 39 40</pre>	<pre>role_hierarchy: ROLE_ADMIN: - ROLE_USER - ROLE_LOCATION_EDIT ROLE_USER: - ROLE_LOCATION_INDEX</pre>

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`:

W App

Cities

Measurements

Logout

Location index

Id	City	Country	Latitude	Langitude	actions
1	Szczecin	PL	53.4289	14.553	edit weather
2	Police	PL	53.5521	14.5718	edit weather

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`:

```

32     access_control:
33         - { path: ^/logout, roles: IS_AUTHENTICATED_ANONYMOUSLY }
34         - { path: ^/location/new, roles: ROLE_ADMIN }           # Tworzenie nowej lokalizacji
35         - { path: ^/location/edit, roles: ROLE_LOCATION_EDIT } # Edytowanie lokalizacji
36         - { path: ^/location, roles: ROLE_LOCATION_INDEX }    # Wyświetlanie lokalizacji
37         - { path: ^/measurement, roles: ROLE_MEASUREMENT_VIEW } # Widok pomiarów
38         - { path: ^/admin, roles: ROLE_ADMIN }
39         - { path: ^/, roles: PUBLIC_ACCESS }
40
41     role_hierarchy:
42         ROLE_ADMIN:
43             - ROLE_USER
44             - ROLE_LOCATION_EDIT      # Uprawnienia do edytowania lokalizacji
45             - ROLE_MEASUREMENT_VIEW  # Uprawnienia do przeglądania pomiarów
46             - ROLE_LOCATION_NEW      # Uprawnienia do tworzenia nowej lokalizacji
47         ROLE_USER:
48             - ROLE_LOCATION_INDEX    # Uprawnienia do przeglądania listy lokalizacji

```

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:

The screenshot shows a web browser at `pogodynka.localhost:34753/location/`. The page has a navigation bar with links like 'Zakupy', 'Social', 'Praca', 'Studia', 'Nauka', 'YouTube', 'Oglądaj', 'AleksandraBancewicz', 'Location index', and 'Dom szkieletowy M...'. Below the navigation bar, there's a 'Weather App' section with 'Locations' and 'Add Location' links. The main content area is titled 'Location index' and contains a table with columns: Id, City, Country, Latitude, Longitude, and actions. The table has two rows: one for 'Szczecin' (ID 1) and one for 'Police' (ID 2). Below the table is a 'Create new' link. At the bottom, there's a 'Logout' link. Below the main content area, there's a dark-themed database query interface showing a table with columns: id, username, roles, and password. The table has two rows: one for 'admin' (ID 1) and one for 'user' (ID 2). The 'password' column for the 'user' row is highlighted in blue.

Id	City	Country	Latitude	Longitude	actions
1	Szczecin	PL	53.4289	14.553	show edit
2	Police	PL	53.5521	14.5718	show edit

[Create new](#)

[Logout](#)

id	username	roles	password
1	admin	["ROLE_ADMIN"]	\$2y\$13\$HYQ8NfxDbrG0tD8DHn5e1.7Ic3...
2	user	["ROLE_USER"]	1234

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.