

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	10
Formularz logowania.....	11
Wylogowywanie.....	15
Hierarchia ról	17
Commit projektu do GIT	19
Podsumowanie	19

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

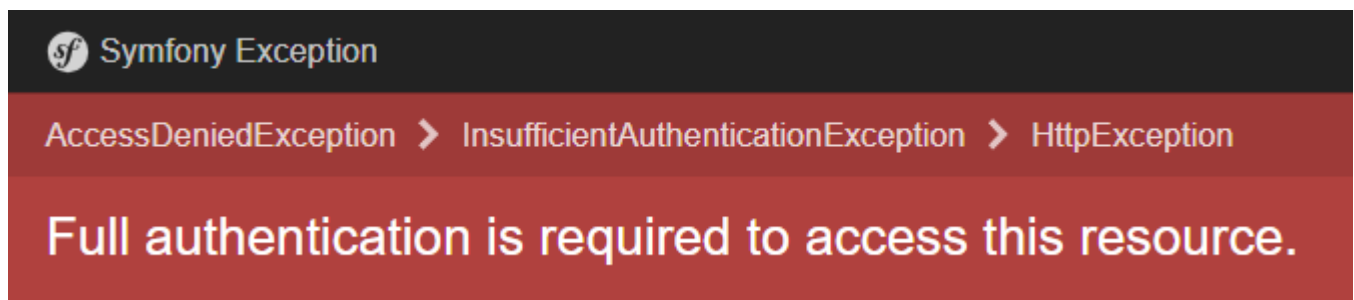
Ten dokument aktywnie wykorzystuje niestandardowe właściwości. Podobnie jak w LAB A wejdź do Plik -> Informacje -> Właściwości -> Właściwości zaawansowane -> Niestandardowe i zaktualizuj pola. Następnie uruchom ten dokument ponownie lub Ctrl+A -> F9.

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:51114/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 działa, ponieważ akcja `index` w kontrolerze `LocationController`

```
#[Route(name: 'app_location_index', methods: ['GET'])]
public function index(LocationRepository $locationRepository): Response
```

nie jest chroniona. Jedynie akcja `new` wymaga odpowiedniej roli użytkownika, dzięki atrybutowi `IsGranted`. Brak tej roli powoduje wyrzucenie wyjątku.

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():

```
#[Route('/new', name: 'app_location_new', methods: ['GET', 'POST'])]
#[IsGranted('ROLE_LOCATION_NEW')]
public function new(Request $request, EntityManagerInterface $entityManager): Response
```

LocationController::index():

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

LocationController::show():

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

LocationController::edit():

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

LocationController::delete():

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

MeasurementController::new():

```
#[Route('/new', name: 'app_measurement_entry_new', methods: ['GET', 'POST'])]
#[IsGranted('ROLE_MEASUREMENT_NEW')]
public function new(Request $request, EntityManagerInterface $entityManager): Response
```

MeasurementController::index():

```
#[Route(name: 'app_measurement_entry_index', methods: ['GET'])]
#[IsGranted('ROLE_MEASUREMENT_INDEX')]
public function index(MeasurementEntryRepository $measurementEntryRepository): Response
```

MeasurementController::show():

```
#[Route('/{id}', name: 'app_measurement_entry_show', methods: ['GET'])]
#[IsGranted('ROLE_MEASUREMENT_SHOW')]
public function show(MeasurementEntry $measurementEntry): Response
```

MeasurementController::edit():

```
#[Route('/{id}/edit', name: 'app_measurement_entry_edit', methods: ['GET', 'POST'])]
#[IsGranted('ROLE_MEASUREMENT_EDIT')]
public function edit(Request $request, MeasurementEntry $measurementEntry, EntityManagerInterface $entityManager): Response
```

MeasurementController::delete():

```
#[Route('/{id}', name: 'app_measurement_entry_delete', methods: ['POST'])]
#[IsGranted('ROLE_MEASUREMENT_DELETE')]
public function delete(Request $request, MeasurementEntry $measurementEntry, 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`:

```

{% extends 'base.html.twig' %}

{% block title %}Location index
{% endblock %}

{% block body %}
    <h1>Location index</h1>

    <div class="table-responsive-sm">
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>Id</th>
                    <th>City</th>
                    <th>Country</th>
                    <th>Latitude</th>
                    <th>Longitude</th>
                    <th>actions</th>
                </tr>
            </thead>
            <tbody>
                {% for location in locations %}
                    <tr>
                        <td>{{ location.id }}</td>
                        <td>{{ location.city }}</td>
                        <td>{{ location.country }}</td>
                        <td>{{ location.latitude }}</td>
                        <td>{{ location.longitude }}</td>
                        <td>
                            {% if is_granted('ROLE_LOCATION_SHOW') %}
                                <a href="{{ path('app_location_show', {'id': location.id}) }}" class="btn btn-info p-0.5">show</a>
                            {% endif %}

                            {% if is_granted('ROLE_LOCATION_EDIT') %}
                                <a href="{{ path('app_location_edit', {'id': location.id}) }}" class="btn btn-secondary">edit</a>
                            {% endif %}
                        </td>
                    </tr>
                {% else %}
                    <tr>
                        <td colspan="6">no records found</td>
                    </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>

    {% if is_granted('ROLE_LOCATION_NEW') %}
        <a href="{{ path('app_location_new') }}" class="btn btn-primary">Create new</a>
    {% endif %}

{% endblock %}

```

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

```
{% extends 'base.html.twig' %}

{% block title %}MeasurementEntry index{% endblock %}

{% block body %}
<h1>MeasurementEntry index</h1>

<div class="table-responsive-sm">
<table class="table table-striped">
<thead>
<tr>
<th>Id</th>
<th>DateTime</th>
<th>Temperature_celcius</th>
<th>Humidity</th>
<th>Pressure</th>
<th>Wind_speed</th>
<th>Wind_direction</th>
<th>Feels_like</th>
<th>actions</th>
</tr>
</thead>
<tbody>
{% for measurement_entry in measurement_entries %}
<tr>
<td>{{ measurement_entry.id }}</td>
<td>{{ measurement_entry.dateTime ? measurement_entry.dateTime|date('Y-m-d H:i:s') : '' }}</td>
<td>{{ measurement_entry.temperatureCelcius }}</td>
<td>{{ measurement_entry.humidity }}</td>
<td>{{ measurement_entry.pressure }}</td>
<td>{{ measurement_entry.windSpeed }}</td>
<td>{{ measurement_entry.windDirection }}</td>
<td>{{ measurement_entry.feelsLike }}</td>
<td>
{% if is_granted('ROLE_MEASUREMENT_SHOW') %}
<a href="{{ path('app_measurement_entry_show', {'id': measurement_entry.id}) }}" class="btn btn-info p-0.5">show</a>
{% endif %}

{% if is_granted('ROLE_MEASUREMENT_EDIT') %}
<a href="{{ path('app_measurement_entry_edit', {'id': measurement_entry.id}) }}" class="btn btn-secondary">edit</a>
{% endif %}
</td>
</tr>
{% else %}
<tr>
<td colspan="9">no records found</td>
</tr>
{% endfor %}
</tbody>
</table>
</div>

{% if is_granted('ROLE_MEASUREMENT_NEW') %}
<a href="{{ path('app_measurement_entry_new') }}" class="btn btn-primary">Create new</a>
{% endif %}

{% 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:51114` – lub w wyświetlonym przez przeglądarkę okienku).

Modyfikuj plik `config/packages/security.yaml`:

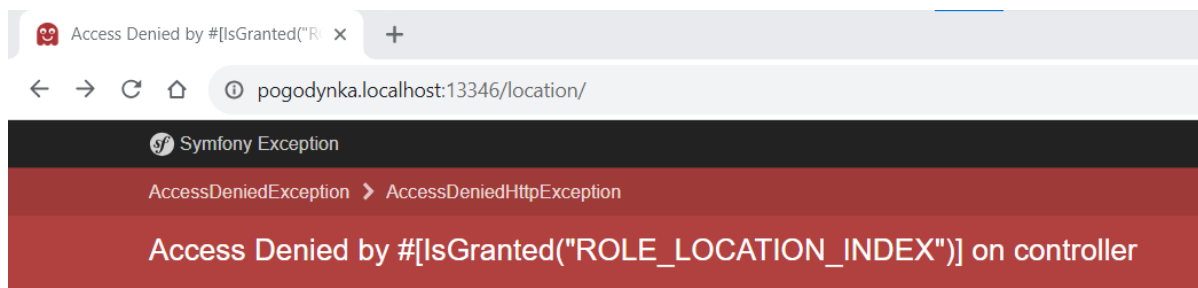
AI2 LAB E – Szabat Krystian – Wersja 1

```
--- 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 poprawności działania:

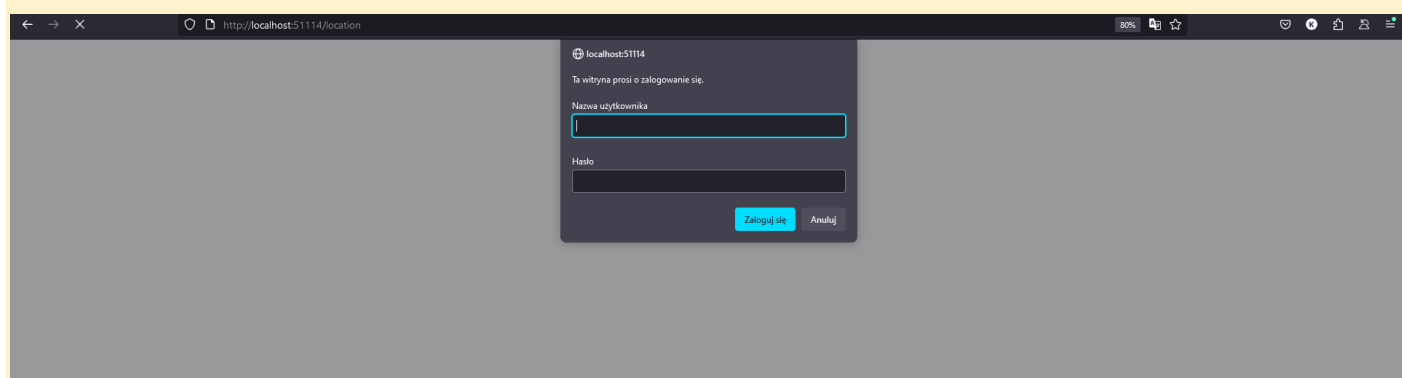
Location index

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:



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    1330467
-----

[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:

```
security:
# https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
password_hashers:
    Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface: "auto"
# https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
providers:
    users_in_memory:
        memory:
            users:
                admin:
                    password: "$2y$13$GgKpAXhDrAT7tAd2Q.h6W.chUrBfcy4GomkaL7ynGbe6XYkmMDm8a"
                    roles:
                        - "ROLE_ADMIN"
                        - "ROLE_USER"
firewalls:
    dev:
        pattern: ^/(_(profiler|wdt)|css|images|js)/
        security: false
    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

# https://symfony.com/doc/current/security/impersonating_user.html
# switch_user: true

# 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: ^/, roles: PUBLIC_ACCESS }
```

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:

Security

Token Firewall Listeners Authenticators Access Decision

Username: admin Authenticated:

Property	Value
Roles	["ROLE_ADMIN" "ROLE_USER"]
Inherited Roles	none
Token	Symfony\Component\Security\Core\Authentication\Token\UsernamePasswordToken {#711 ▾ -user: App\Entity\User {#707 ...} -roleNames: [-attributes: [] -firewallName: "main" }]

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

var > data.db

Filter 7 tables... Rows: 1

TABLES

- location
- measurement_entry
- measurement_entry_weather_...
- messenger_messages
- sqlite_sequence
- user
- weather_condition

id	username	roles	password
1	2	admin	["ROLE_ADMIN"]

↑ ↓ ↻ user / 2

id: 2 (INTEGER, READONLY)

username: admin (VARCHAR(180), READONLY)

roles: ["ROLE_ADMIN"] (CLOB, READONLY)

password: \$2y\$13\$GgKpAXhDrAT7tAd2Q.h6W.chUrBfcy4GomkaL7ynGbe6XYkmMDm8a (VARCHAR(255), READONLY)

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:

The screenshot shows a web browser window with the address bar displaying `pogodynka.localhost:13346/login`. The page title is "Login". The form contains two input fields: "Username:" and "Password:", followed by a "login" button. Below the form, there are two rows of ten "00000" characters each. The browser's developer tools are open at the bottom, showing the "200 @app_login" status, a response time of 284 ms, a size of 26.0 MiB, and the user "admin".

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:

Security

Token Firewall Listeners Authenticators Access Decision

Username: **admin** Authenticated:

Property	Value
Roles	["ROLE_ADMIN" "ROLE_USER"]
Inherited Roles	none
Token	Symfony\Component\Security\Core\Authentication\Token\UsernamePasswordToken {#770 ▾ -user: App\Entity\User {#875 ...} -roleNames: [-attributes: [] -firewallName: "main" }

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:

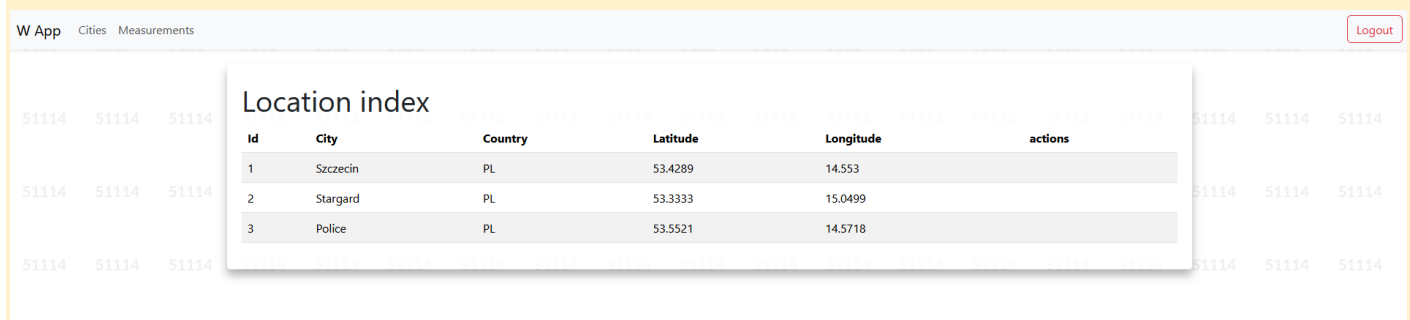
main:	16	16	main:
lazy: true	17	17	lazy: true
provider: app_user_provider	18	18	provider: app_user_provider
http_basic:	>> 19	19 <input type="checkbox"/>	form_login:
realm: 'My Secured Area'	20	20	login_path: app_login
	21	21	check_path: app_login
# activate different ways to authenticate	22	22	logout:
# https://symfony.com/doc/current/security.html	23	23	path: app_logout

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

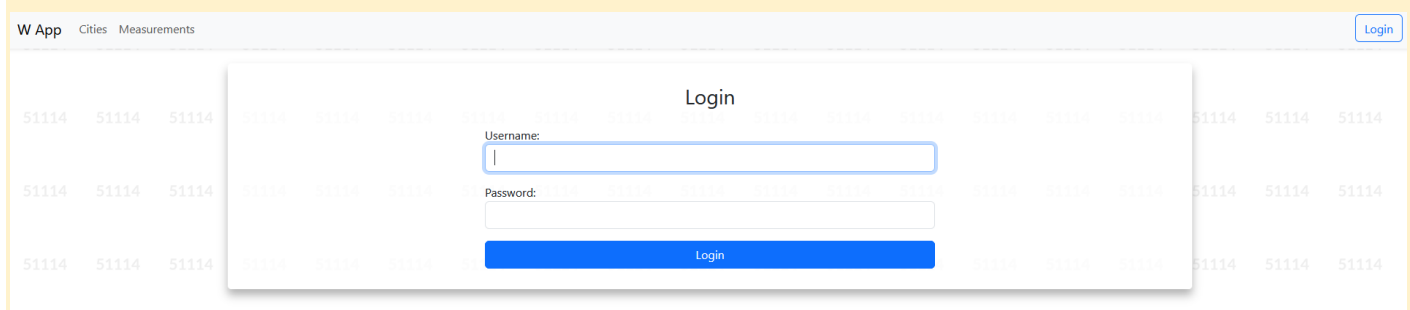
```
{% if is_granted('ROLE_USER') %}
    <a href="{{ path('app_logout') }}">Logout</a>
{% else %}
```

```
<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:

```
</div>
{% if is_granted("ROLE_USER") %}
    <a href="{{ path('app_logout') }}" class="btn btn-outline-danger">Logout</a>
{% else %}
    You, 2 seconds ago • Uncommitted changes
    <a href="{{ path('app_login') }}" class="btn btn-outline-primary">Login</a>
{% endif %}
</div>
</nav>
```

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

```
main:
  lazy: true
  provider: app_user_provider
  form_login:
    login_path: app_login
    check_path: app_login
  logout:
    path: app_logout
```


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`):

<code>access_control:</code>	30	32	<code>access_control:</code>
<code>- { path: ^/, roles: PUBLIC_ACCESS }</code>	31	33	<code>- { path: ^/, roles: PUBLIC_ACCESS }</code>
	>> 32	34 <input type="checkbox"/>	
<code>!@test:</code>	33	35	<code>role_hierarchy:</code>
<code>security:</code>	34	36	<code>ROLE_ADMIN:</code>
<code>password_hashers:</code>	35	37	<code>- ROLE_USER</code>
<code># By default, password hashers are resou</code>	36	38	<code>- ROLE_LOCATION_EDIT</code>
<code># important to generate secure password I</code>	37	39	<code>ROLE_USER:</code>
<code># are not important, waste resources and</code>	38	40	<code>- ROLE_LOCATION_INDEX</code>

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`:

role_hierarchy:**ROLE_ADMIN:**

- ROLE_USER
- ROLE_LOCATION_EDIT
- ROLE_LOCATION_DELETE
- ROLE_LOCATION_NEW
- ROLE_MEASUREMENT_EDIT
- ROLE_MEASUREMENT_DELETE
- ROLE_MEASUREMENT_NEW

ROLE_USER:

- ROLE_LOCATION_INDEX
- ROLE_LOCATION_SHOW
- ROLE_MEASUREMENT_INDEX
- ROLE_MEASUREMENT_SHOW

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:

Location index

Id	City	Country	Latitude	Longitude	actions
1	Szczecin	PL	53.4289	14.553	show edit
2	Stargard	PL	53.3333	15.0499	show edit
3	Police	PL	53.5521	14.5718	show edit

[Create new](#)

W App Cities Measurements [Logout](#)

© app_location_index 3099 ms 60.0 MB 2 2 in 1.86 ms admin 79 ms 2 in 1.55 ms 6.4.12

W App Cities Measurements Logout

Location index

Id	City	Country	Latitude	Longitude	actions
1	Szczecin	PL	53.4289	14.553	show
2	Stargard	PL	53.3333	15.0499	show
3	Police	PL	53.5521	14.5718	show

@app_location_index 330 ms 24.0 MB 2 user1 41 ms 2 in 1.03 ms 6.4.12

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/szvbvtk/ai2-pogodynka/tree/lab-e>

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Podczas laboratorium nauczyłem się jak zabezpieczać dostęp do wybranych akcji poprzez przypisywanie odpowiednich ról do użytkowników. Dowiedziałem się również jak zaimplementować logowanie z wykorzystaniem Basic Auth.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.