

FORMULARZE I WALIDACJA

SPIS TREŚCI

Spis treści	1
Cel zajęć.....	1
Rozpoczęcie	1
Uwaga	1
Encja Location	Błąd! Nie zdefiniowano zakładki.
Pozostałe encje.....	Błąd! Nie zdefiniowano zakładki.
Kontroler	Błąd! Nie zdefiniowano zakładki.
Repozytorium	Błąd! Nie zdefiniowano zakładki.
Wyszukiwanie lokacji po nazwie miasta.....	Błąd! Nie zdefiniowano zakładki.
Commit projektu do GIT.....	12
Podsumowanie.....	13

CEL ZAJĘĆ

Celem głównym zajęć jest zdobycie następujących umiejętności:

- Tworzenie typów formularzy.
- Wizualizacja formularzy w Twig.
- Obsługa formularzy w kontrolerze.
- Walidacja danych.

ROZPOCZĘCIE

Rozpoczęcie zajęć. Powtórzenie tworzenia formularzy w Symfony. 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.

PREZENTACJA TWORZENIA CRUD

Obserwuj wspólnie z resztą grupy. Prowadzący przedstawi komendę `make:crud` służącą do budowania kompletu operacji tworzenia, odczytu, aktualizacji i kasowania encji. Komenda przedstawiona zostanie na przykładzie kontrolera `LocationController` dla encji `Location`.

```
php .\bin\console make:crud

The class name of the entity to create CRUD (e.g. GentleChef):
> Location

Choose a name for your controller class (e.g. LocationController)
[LocationController]:
>

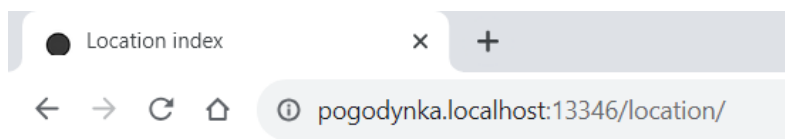
Do you want to generate tests for the controller?. [Experimental] (yes/no) [no]:
>

created: src/Controller/LocationController.php
created: src/Form/LocationType.php
created: templates/location/_delete_form.html.twig
created: templates/location/_form.html.twig
created: templates/location/edit.html.twig
created: templates/location/index.html.twig
created: templates/location/new.html.twig
created: templates/location/show.html.twig

Success!

Next: Check your new CRUD by going to /location/
```

Zgodnie z informacją wyświetloną na końcu komendy, można od razu zobaczyć wyniki działania komendy w przeglądarce pod adresem `http://pogodynka.localhost:00000/location`:



Location index

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					

Wspólnie zbadać zawartość:

- kontrolera `src/Controller/LocationController.php`;
- typu formularza `src/Form/LocationType.php`;

- widoków `templates/location/`.

Wspólnie zmodyfikujcie `LocationType`, żeby określić typy kontrolek i ich opcje (np. wybór państwa z listy zamiast pola tekstowego). Dodajcie walidacje w dwóch grupach walidacyjnych – dla `edit` i dla `new`.

Przykłady:

```
class LocationType extends AbstractType
{
    no usages
    public function buildForm(FormBuilderInterface $builder, array $options): void
    {
        $builder
            ->add( child: 'city', type: null, [
                'attr' => [
                    'placeholder' => 'Enter city name',
                ],
            ])
            ->add( child: 'country', type: ChoiceType::class, [
                'choices' => [
                    'Poland' => 'PL',
                    'Germany' => 'DE',
                    'France' => 'FR',
                    'Spain' => 'ES',
                    'Italy' => 'IT',
                    'United Kingdom' => 'GB',
                    'United States' => 'US',
                ],
            ])
    }
}
```

```
#[Route('/new', name: 'app_location_new', methods: ['GET', 'POST'])]
public function new(Request $request, EntityManagerInterface $entityManager): Response
{
    $location = new Location();
    $form = $this->createForm( type: LocationType::class, $location, [
        'validation_groups' => 'create',
    ]);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $entityManager->persist($location);
        $entityManager->flush();

        return $this->redirectToRoute( route: 'app_location_index', [], status: Response::HTTP_SEE_OTHER);
    }

    return $this->render( view: 'location/new.html.twig', [
        'location' => $location,
        'form' => $form,
    ]);
}
```

Na koniec edytujcie twig.yaml oraz base.htm.twig w celu zmiany renderowania strony na Bootstrap 5.

Oczekiwany efekt:

W App

Weather

Measurements

☰

Edit Location

City

Szczecin

Country

Poland

Latitude

53,4289000

Longitude

14,5530000

Update

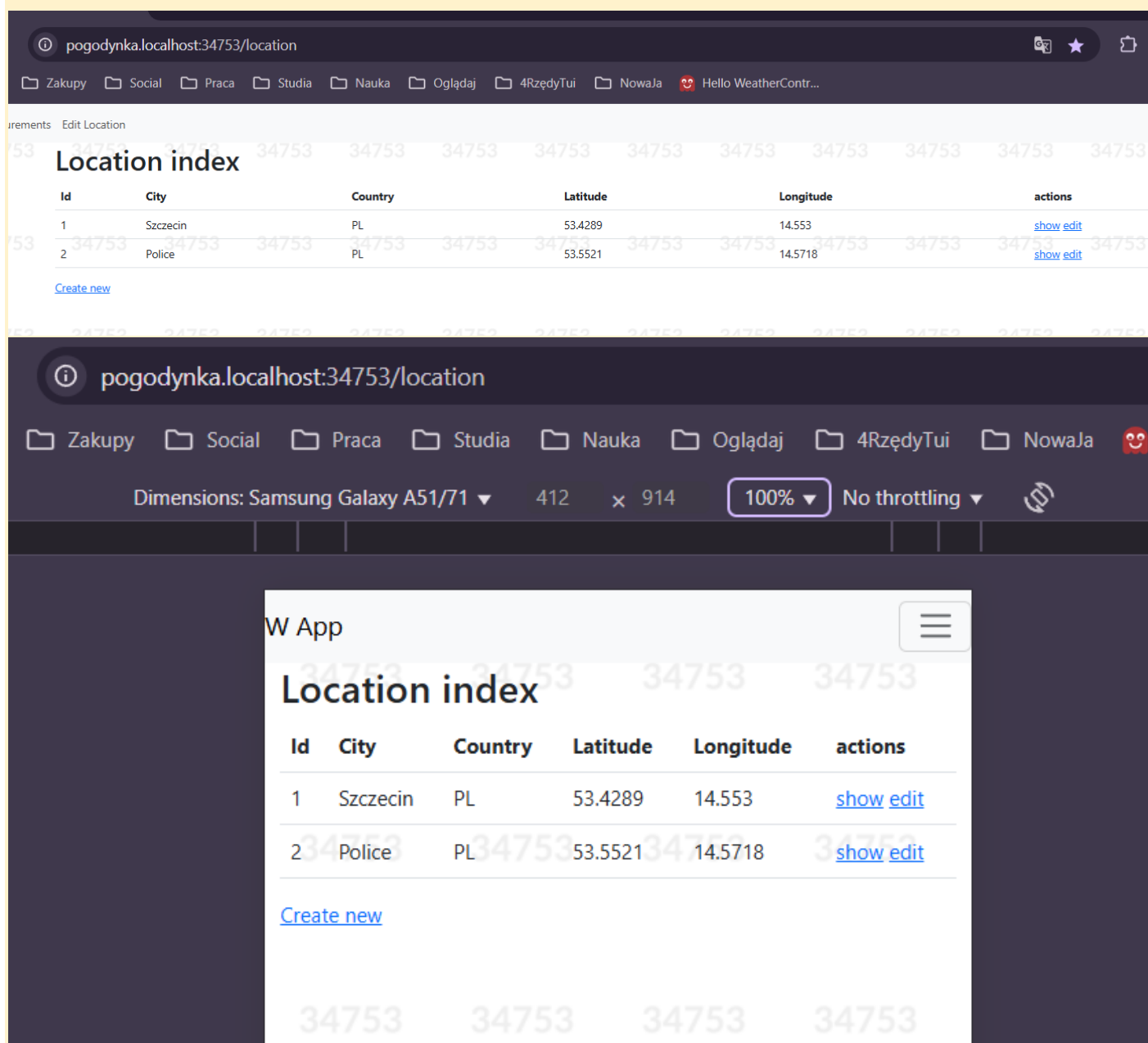
[back to list](#)

Delete

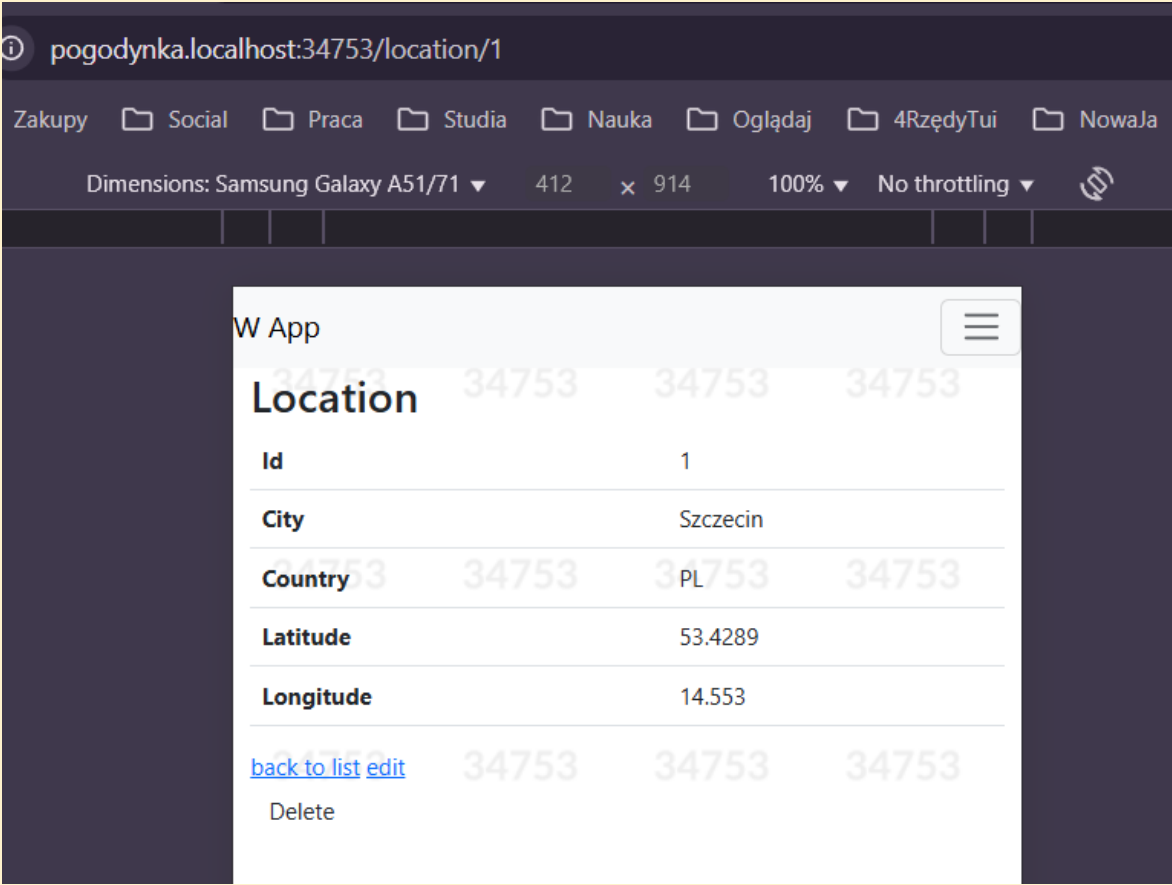
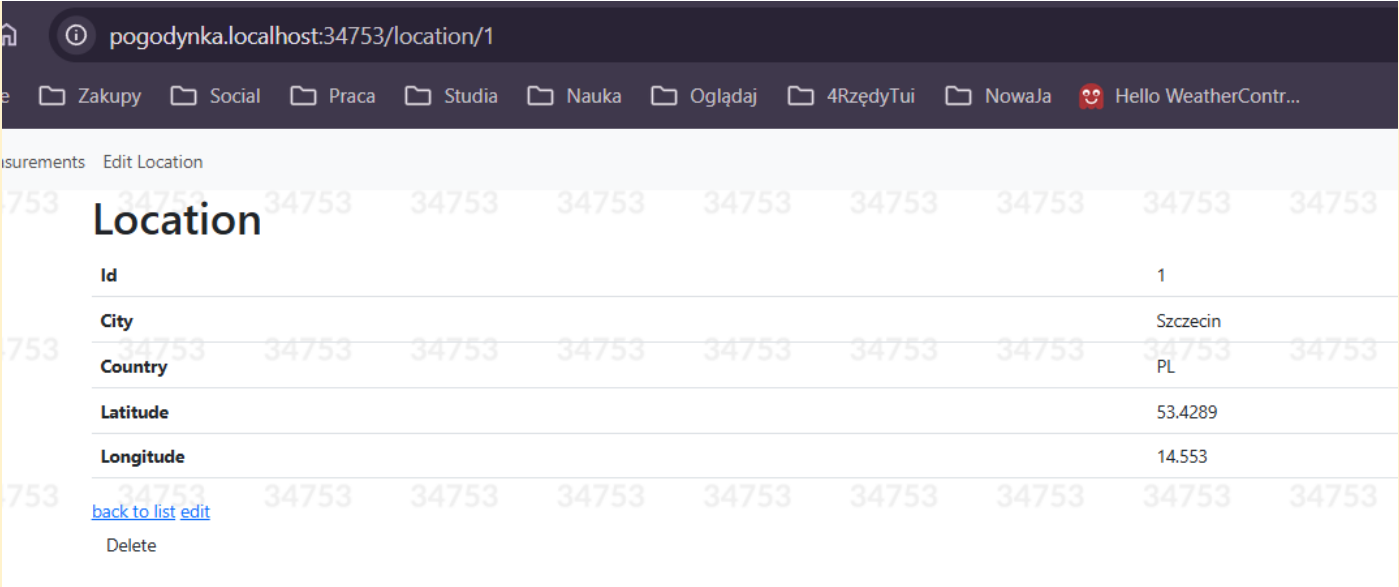
BUDOWA CRUD

Zbuduj CRUD dla encji Location jak podczas prezentacji oraz analogicznie formularz MeasurementType do tworzenia i edycji prognoz pogody, a następnie kontroler Measurement z akcjami listy, tworzenia, edycji i usuwania. Zaprezentuj osobne walidacje do tworzenia i edycji obu encji. Wykorzystaj konfigurację walidacji w YAML. Na koniec dodaj do szablonu strony (base.html.twig) linki do CRUDA lokalizacji i prognoz.

Wstaw zrzut ekranu listy lokalizacji. Upewnij się, że zastosowano layout Bootstrap 5. Wstaw zrzut wersji desktop i wersji mobilnej:



Wstaw analogiczne zrzuty dla listy pomiarów:

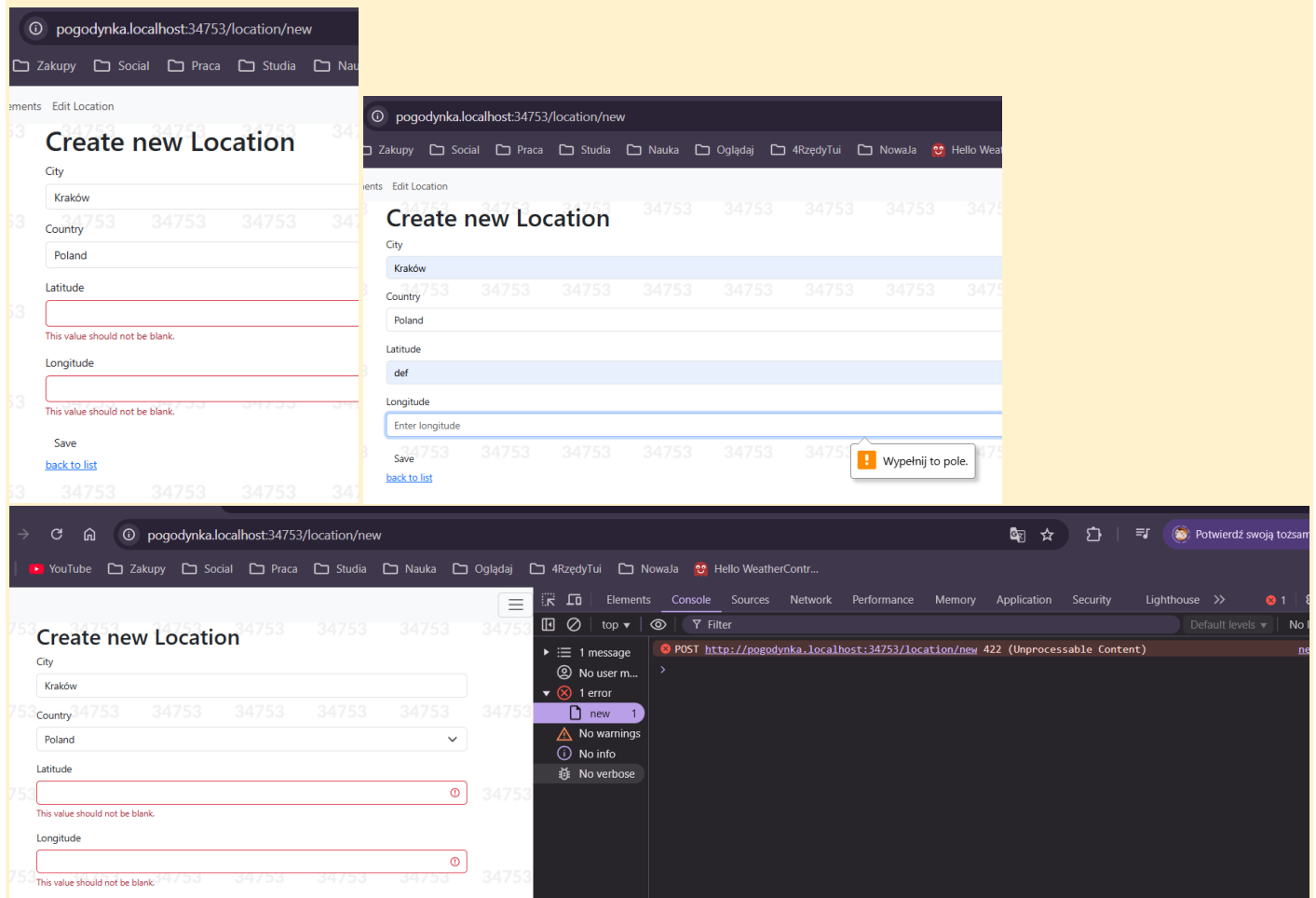


Upewnij się, że na wszystkich powyższych 4 zrzutach ekranu znajdują się linki umożliwiające przejście z lokalizacji do pomiarów i z powrotem.

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

LOKALIZACJA

Wstaw zrzut ekranu tworzenia lokalizacji (tylko desktop). Upewnij się, że na zrzucie ekranu widoczna jest działająca walidacja (wyświetlony komunikat o błędzie w którymś polu na formularzu). Uwaga! Tu ma się znaleźć walidacja po stronie serwera, a nie frontendowa walidacja HTML5 (statystycznie 90% osób prezentuje tutaj złą walidację 😊).



Wstaw zrzut ekranu kodu odpowiedzialnego za walidację (w formacie YAML).

```

2 usages
#[ORM\Column(type: Types::DECIMAL, precision: 10, scale: 7)]
#[Assert\NotBlank(groups: ['create'])]
#[Assert\Range(min: -90, max: 90, groups: ['create', 'edit'])]
private ?string $latitude = null;

2 usages
#[ORM\Column(type: Types::DECIMAL, precision: 10, scale: 7)]
#[Assert\NotBlank(groups: ['create'])]
#[Assert\Range(min: -180, max: 180, groups: ['create', 'edit'])]
private ?string $longitude = null;

```

```

Location.php  validator.yaml  Measurement.php  MeasurementType.php  base.html.twig
1 framework:
2   validation:
3     email_validation_mode: html5
4
5     # Enables validator auto-mapping support.
6     # For instance, basic validation constraints will be inferred from Doctrine's metadata.
7     #auto_mapping:
8     #   App\Entity\: []
9
10 when@test:
11   framework:
12     validation:
13       not_compromised_password: false

```

```

Location.yaml  validator.yaml
1 # config/validator/Location.yaml
2 App\Entity\Location:
3   properties:
4     city:
5       - NotBlank: ~
6       - Length:
7         max: 255
8     country:
9       - NotBlank: ~
10      - Length:
11        max: 2
12     latitude:
13       - NotBlank: ~
14       - Type:
15         type: numeric
16       - Range:
17         min: -90
18         max: 90
19     longitude:
20       - NotBlank: ~
21       - Type:
22         type: numeric
23       - Range:
24         min: -180
25         max: 180
26

```

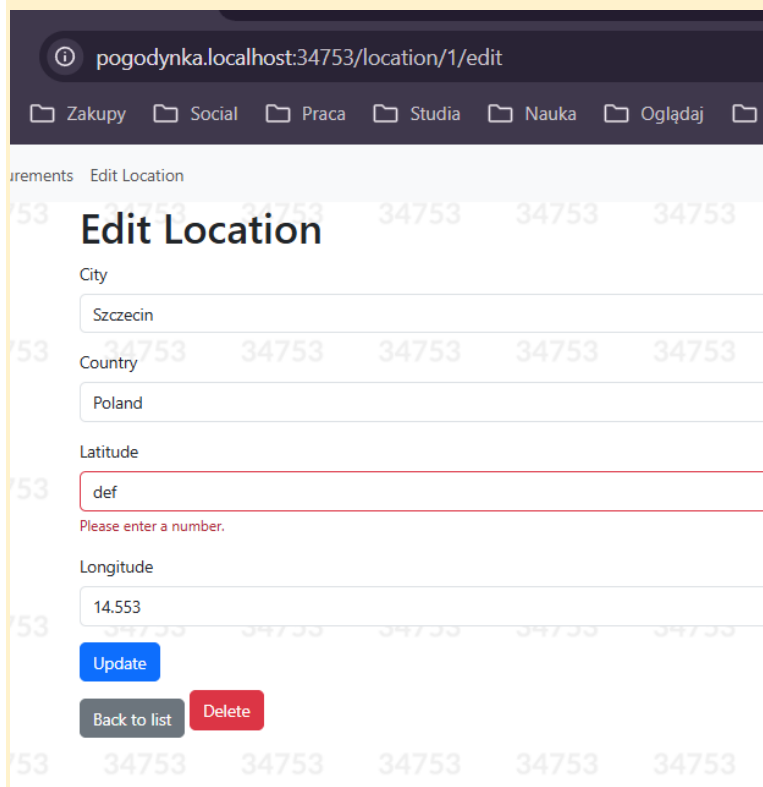
Plik w yaml zazwyczaj wygląda tak, ale moja struktura plików wygląda nieco inaczej niż to czego dowiedziałam się z internetów, tak więc używam Assertów do walidacji. Zakładam, że chodzi o wersję Symfony.

Punkty:

0

1

Wstaw zrzut ekranu edycji lokalizacji (tylko desktop). Upewnij się, że na zrzucie ekranu widoczna jest działająca walidacja (wyświetlony komunikat o błędzie w którymś polu na formularzu).



The screenshot shows a web browser window with the address bar displaying 'pogodynka.localhost:34753/location/1/edit'. The browser's tab bar shows several tabs: 'Zakupy', 'Social', 'Praca', 'Studia', 'Nauka', 'Oglądaj', and an empty tab. The page title is 'Edit Location'. The form contains the following fields and values:

- City: Szczecin
- Country: Poland
- Latitude: def (with a red border and a red error message 'Please enter a number.' below it)
- Longitude: 14.553

At the bottom of the form, there are three buttons: 'Update' (blue), 'Back to list' (grey), and 'Delete' (red).

Wstaw zrzut ekranu kodu odpowiedzialnego za walidację (w formacie YAML).

```
2 usages
#[ORM\Column(type: Types::DECIMAL, precision: 10, scale: 7)]
#[Assert\NotBlank(groups: ['create'])]
#[Assert\Range(min: -90, max: 90, groups: ['create', 'edit'])]
private ?string $latitude = null;

2 usages
#[ORM\Column(type: Types::DECIMAL, precision: 10, scale: 7)]
#[Assert\NotBlank(groups: ['create'])]
#[Assert\Range(min: -180, max: 180, groups: ['create', 'edit'])]
private ?string $longitude = null;
```

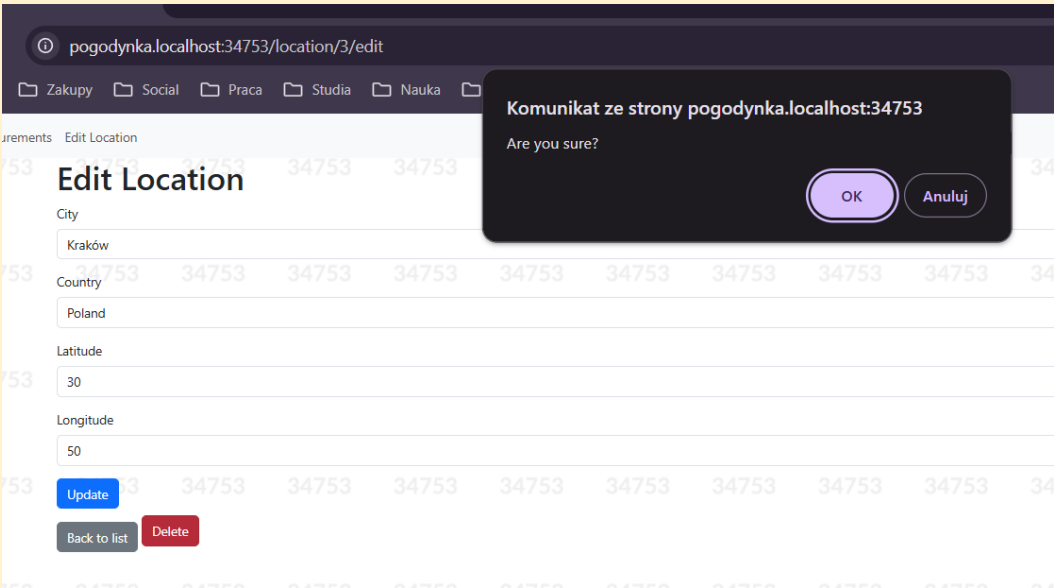
Punkty:

0

1

AI2 LAB D – BancewiczBancewicz Aleksandra – Wersja 1

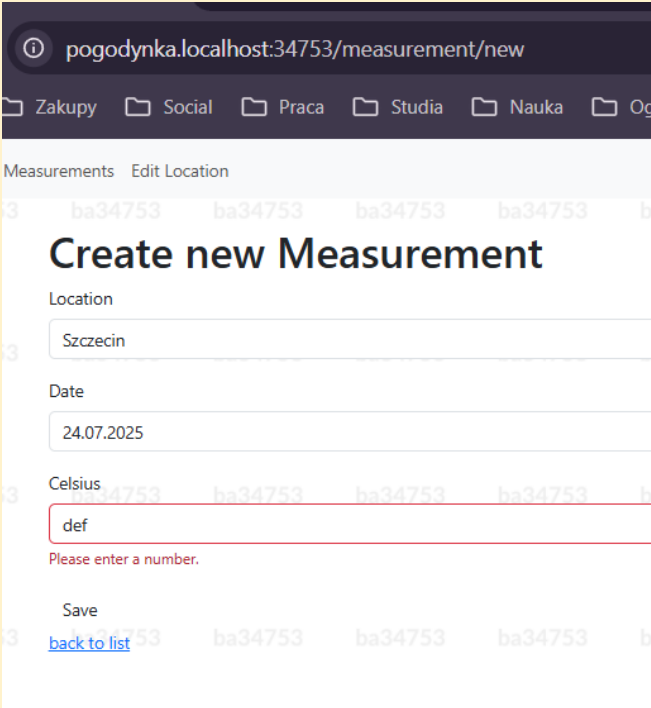
Wstaw zrzuty ekranu kasowania lokalizacji (tylko desktop).



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

POMIARY

Wstaw zrzut ekranu tworzenia pomiarów (tylko desktop). Upewnij się, że na zrzucie ekranu widoczna jest działająca walidacja (wyświetlony komunikat o błędzie w którymś polu na formularzu).



Wstaw zrzut ekranu kodu odpowiedzialnego za walidację (w formacie YAML).

```

2 usages
#[ORM\Column(type: Types::DECIMAL, precision: 3, scale: 0)]
#[Assert\NotBlank(groups: ['create', 'edit'])]
#[Assert\Range(min: -50, max: 50, groups: ['create', 'edit'])]
private ?string $celsius = null;

```

1 usage

Punkty:

0

1

Wstaw zrzut ekranu edycji pomiarów (tylko desktop). Upewnij się, że na zrzucie ekranu widoczna jest działająca walidacja (wyświetlony komunikat o błędzie w którymś polu na formularzu).

The screenshot shows a web browser at the URL `pogodynka.localhost:34753/measurement/1/edit`. The page title is "Edit Measurement". The form contains three input fields: "Location" with the value "Szczecin", "Date" with the value "23.01.2025", and "Celsius" with the value "def". A red border highlights the "Celsius" field, and a red error message "Please enter a number." is displayed below it. At the bottom of the form, there are buttons for "Update", "back to list", and "Delete".

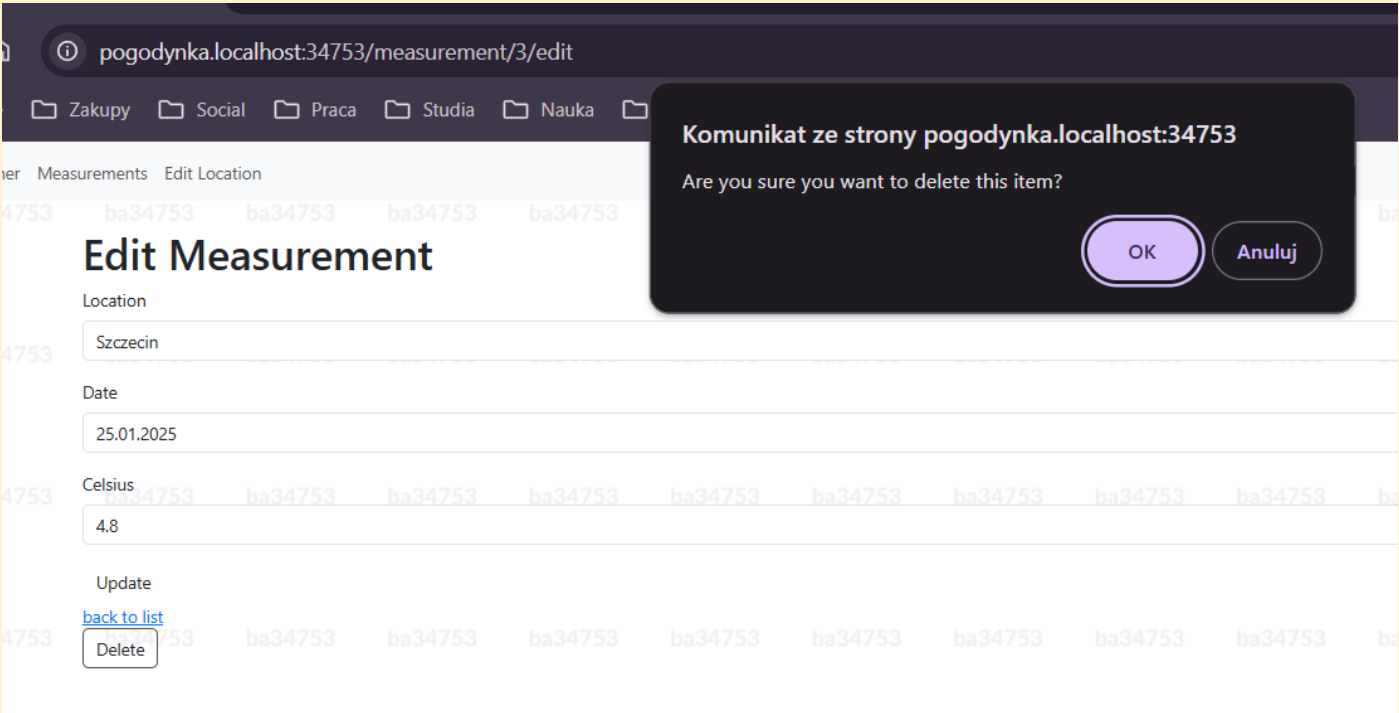
Wstaw zrzut ekranu kodu odpowiedzialnego za walidację (w formacie YAML).

```
2 usages
#ORM\Column(type: Types::DECIMAL, precision: 3, scale: 0)]
#[Assert\NotBlank(groups: ['create', 'edit'])]
#[Assert\Range(min: -50, max: 50, groups: ['create', 'edit'])]
private ?string $celsius = null;

1 usage
```

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

Wstaw zrzuty ekranu kasowania pomiarów (tylko desktop).



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-d na podstawie głównej gałęzi kodu.

Podaj link do brancha lab-d w swoim repozytorium:
https://github.com/AleksandraBancewicz/AI2/tree/main/LD

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Podczas laboratorium nauczyłam się, jak tworzyć pełne CRUD-y w Symfony, w tym generować kontrolery, formularze oraz implementować walidację danych. Zrozumiałam, jak ważne jest przeprowadzanie walidacji po stronie serwera, unikając walidacji HTML5, a także jak korzystać z adnotacji Assert do definiowania reguł walidacyjnych. Praktykowałam używanie formularzy w Symfony i dostosowywanie ich do specyficznych potrzeb, np. zmieniając typy pól, jak wybór państwa. Poznałam również konfigurację walidacji w YAML, co pozwala na centralne zarządzanie regułami w projekcie. Dodatkowo nauczyłam się korzystać z Twig do tworzenia estetycznych formularzy z użyciem Bootstrap 5. Całość pozwoliła mi rozwinąć umiejętności w zakresie tworzenia aplikacji webowych w Symfony.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.