

BAZA DANYCH I OPERACJE ODCZYTU

SPIS TREŚCI

Spis treści.....	1
Cel zajęć.....	1
Rozpoczęcie	1
Uwaga.....	1
Encja Location.....	2
Pozostałe encje	4
Kontroler	7
Repozytorium	10
Wyszukiwanie lokacji po nazwie miasta.....	14
Commit projektu do GIT.....	17
Podsumowanie	17

CEL ZAJĘĆ

Celem głównym zajęć jest zdobycie umiejętności tworzenia encji na podstawie diagramów ERD oraz opanowanie procesu tworzenia akcji w systemie monolitycznym – routing, kontroler, widok.

ROZPOCZĘCIE

Rozpoczęcie zajęć. Powtórzenie zasad routingów w Symfony – atrybuty, adnotacje, yaml. Określanie parametrów. Określenie wymagań parametrów. Powtórzenie przekazywania parametrów do akcji kontrolera (parametry, serwisy, type-hinting i argument resolving). Powtórzenie TWIG – trzy typy wąsów, filtry (np. join, raw), pętle.

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**.

ENCJA LOCATION

Pracuj wspólnie z resztą grupy. Utworzymy wspólnie encję `Location` z wykorzystaniem komendy `make:entity`.

Otwórz projekt I : \AI2-lab\pogodynka w PhpStorm / VS Code. W pliku `.env` zmień bazę danych na SQLITE:

```
# .env
#...
DATABASE_URL="sqlite:///kernel.project_dir%/var/data.db"
```

Ten wpis oznacza, że aplikacja będzie korzystać z bazy danych SQLite umieszczonej w pliku I : \AI2-lab\pogodynka\var\data.db.

Otwórz terminal. Wykonaj polecenia, w celu utworzenia encji `Location`. Prowadzący omówi proces na udostępnionym ekranie:

```
cd I:\AI2-lab\pogodynka
php bin\console make:entity

Class name of the entity to create or update (e.g. GentleKangaroo):
> Location

created: src/Entity/Location.php
created: src/Repository/LocationRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):
> city

Field type (enter ? to see all types) [string]:
>

Field length [255]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Location.php

Add another property? Enter the property name (or press <return> to stop adding fields):
> country

Field type (enter ? to see all types) [string]:
>

Field length [255]:
> 2

Can this field be null in the database (nullable) (yes/no) [no]:
```

```
>

updated: src/Entity/Location.php

Add another property? Enter the property name (or press <return> to stop adding
fields):
> latitude

Field type (enter ? to see all types) [string]:
> decimal

Precision (total number of digits stored: 100.00 would be 5) [10]:
> 10

Scale (number of decimals to store: 100.00 would be 2) [0]:
> 7

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Location.php

Add another property? Enter the property name (or press <return> to stop adding
fields):
> longitude

Field type (enter ? to see all types) [string]:
> decimal

Precision (total number of digits stored: 100.00 would be 5) [10]:
> 10

Scale (number of decimals to store: 100.00 would be 2) [0]:
> 7

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Location.php

Add another property? Enter the property name (or press <return> to stop adding
fields):
>

Success!

Next: When you're ready, create a migration with php bin/console make:migration
```

Razem z grupą omówcie powstałe pliki `Location.php` i `LocationRepository.php`.

Na tym etapie model danych nie został jeszcze naniesiony na bazę danych. Wykonaj komendy:

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

Czym różni się --dump-sql od --force?

--dump-sql: wyświetla kod sql potrzebny do wygenerowania odpowiedniego schematu bazy danych na podstawie encji, nie dokonuje żadnych zmian w bazie danych, kod można wykonać samemu.

--force: wykonuje wygenerowane zapytania, aktualizując tym samym schemat bazy danych, np. dodając nowe tabele lub kolumny

Umieść zrzut ekranu lub skopiuj SQL, który został wygenerowany:

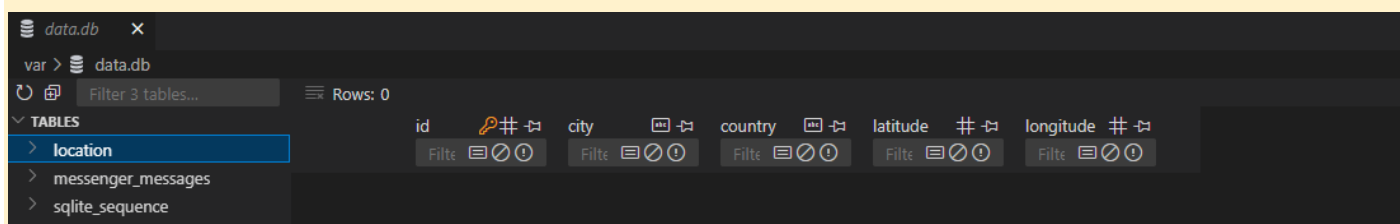
```
CREATE TABLE location (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, city VARCHAR(255) NOT NULL, country VARCHAR(2) NOT NULL, latitude NUMERIC(10, 7) NOT NULL, longitude NUMERIC(10, 7) NOT NULL);
CREATE TABLE messenger_messages (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, body CLOB NOT NULL, headers CLOB NOT NULL, queue_name VARCHAR(190) NOT NULL, created_at DATETIME NOT NULL --(DC2Type:datetime_immutable)
, available_at DATETIME NOT NULL --(DC2Type:datetime_immutable)
, delivered_at DATETIME DEFAULT NULL --(DC2Type:datetime_immutable)
);
CREATE INDEX IDX_75EA56E0F87336F0 ON messenger_messages (queue_name);
CREATE INDEX IDX_75EA56E0E3BD61CE ON messenger_messages (available_at);
CREATE INDEX IDX_75EA56E016BA31DB ON messenger_messages (delivered_at);

Updating database schema...

5 queries were executed

[OK] Database schema updated successfully!
```

Wykorzystaj PhpStorm lub VS Code do połączenia się z bazą danych w pliku I: \AI2-1ab\pogodynka\var\data.db. Umieść poniżej zrzut ekranu drzewa tabel/kolumn:



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

POZOSTAŁE ENCJE

Stwórz pozostałe encje na podstawie swojego diagramu ERD z poprzednich zajęć. Zwrócić uwagę na typ danych **relation** przy tworzeniu relacji pomiędzy encjami.

Wymagane co najmniej encje Location i Measurement (lub odpowiedniki).

```
pogodynka> php .\bin\console make:entity
```

```
Class name of the entity to create or update (e.g. DeliciousPopsicle):
> Measurement
```

```
created: src/Entity/Measurement.php
```

```
created: src/Repository/MeasurementRepository.php
```

```
Entity generated! Now let's add some fields!
```

```
You can always add more fields later manually or by re-running this command.
```

New property name (press <return> to stop adding fields):

> location

Field type (enter ? to see all types) [string]:

> relation

What class should this entity be related to?:

> Location

What type of relationship is this?

Type	Description
ManyToOne	Each Measurement relates to (has) one Location. Each Location can relate to (can have) many Measurement objects.
OneToMany	Each Measurement can relate to (can have) many Location objects. Each Location relates to (has) one Measurement.
ManyToMany	Each Measurement can relate to (can have) many Location objects. Each Location can also relate to (can also have) many Measurement objects.
OneToOne	Each Measurement relates to (has) exactly one Location. Each Location also relates to (has) exactly one Measurement.

Relation type? [ManyToOne, OneToMany, ManyToMany, OneToOne]:

> ManyToOne

Is the Measurement.location property allowed to be null (nullable)? (yes/no) [yes]:

> no

Do you want to add a new property to Location so that you can access/update Measurement objects from it - e.g. \$location->getMeasurements()? (yes/no) [yes]:

> yes

A new property will also be added to the Location class so that you can access the related Measurement objects from it.

New field name inside Location [measurements]:

>

Do you want to activate orphanRemoval on your relationship?

A Measurement is "orphaned" when it is removed from its related Location.

e.g. \$location->removeMeasurement(\$measurement)

NOTE: If a Measurement may *change* from one Location to another, answer "no".

```
Do you want to automatically delete orphaned App\Entity\Measurement objects
(orphanRemoval)? (yes/no) [no]:
>

updated: src/Entity/Measurement.php
updated: src/Entity/Location.php

Add another property? Enter the property name (or press <return> to stop adding
fields):
> date

Field type (enter ? to see all types) [string]:
> date

Can this field be null in the database (nullable) (yes/no) [no]:
> no

updated: src/Entity/Measurement.php

Add another property? Enter the property name (or press <return> to stop adding
fields):
> celsius

Field type (enter ? to see all types) [string]:
> decimal

Precision (total number of digits stored: 100.00 would be 5) [10]:
> 3

Scale (number of decimals to store: 100.00 would be 2) [0]:
> 0

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Measurement.php

Add another property? Enter the property name (or press <return> to stop adding
fields):
>

Success!
```

Zsynchronizuj schemat bazy danych z utworzonymi encjami. Umieść poniżej wygenerowany i wykonany kod SQL:

AI2 LAB C – Szabat Krystian – Wersja 1

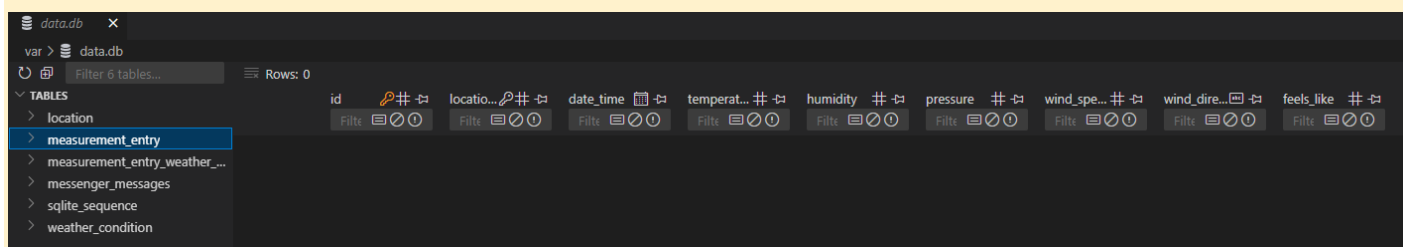
```
PS C:\studia\AI2-Lab\pogodynka> php bin\console doctrine:schema:update --dump-sql --force
CREATE TABLE measurement_entry (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, location_id INTEGER NOT NULL, date_time DATETIME NOT NULL, temperature_celsius NUMERIC(5, 2) NOT NULL, humidity NUMERIC(5, 2) NOT NULL, pressure NUMERIC(5, 1) NOT NULL, wind_speed NUMERIC(3, 0) NOT NULL, wind_direction VARCHAR(3) NOT NULL, feels_like NUMERIC(5, 2) NOT NULL, CONSTRAINT FK_373E6AF664D218E FOREIGN KEY (location_id) REFERENCES location (id) NOT DEFERRABLE INITIALLY IMMEDIATE);
CREATE INDEX IDX_373E6AF664D218E ON measurement_entry (location_id);
CREATE TABLE measurement_entry_weather_condition (measurement_entry_id INTEGER NOT NULL, weather_condition_id INTEGER NOT NULL, PRIMARY KEY(measurement_entry_id, weather_condition_id), CONSTRAINT FK_F577E488B97B54E3 FOREIGN KEY (measurement_entry_id) REFERENCES measurement_entry (id) ON DELETE CASCADE NOT DEFERRABLE INITIALLY IMMEDIATE, CONSTRAINT FK_F577E488B97B54E3 FOREIGN KEY (weather_condition_id) REFERENCES weather_condition (id) ON DELETE CASCADE NOT DEFERRABLE INITIALLY IMMEDIATE);
CREATE INDEX IDX_F577E488B97B54E3 ON measurement_entry_weather_condition (measurement_entry_id);
CREATE INDEX IDX_F577E488B97B54E3 ON measurement_entry_weather_condition (weather_condition_id);
CREATE TABLE weather_condition (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, name VARCHAR(30) NOT NULL, description CLOB DEFAULT NULL);
CREATE INDEX IDX_F577E488B97B54E3 ON weather_condition (name);

Updating database schema...

6 queries were executed

[OK] Database schema updated successfully!
```

Umieść poniżej zrzut ekranu podglądu zaktualizowanej bazy danych SQLite:



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

Finalnie, wypełnij bazę danych przykładowymi wpisami:

- Szczecin, PL, [53.4289, 14.553]
- Police, PL, [53.5521, 14.5718]

KONTROLER

Utwórz pusty kontroler z wykorzystaniem komendy:

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

Choose a name for your controller class (e.g. TinyPopsicleController):
> WeatherController

created: src/Controller/WeatherController.php
created: templates/weather/index.html.twig

Success!

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

Utworzony został plik `src/Controller/WeatherController.php`. Zwróć uwagę na wykorzystanie routingów w postaci atrybutów:

```

1  <?php
2
3  namespace App\Controller;
4
5  > use ...
6
7
8
9  no usages
10 class WeatherController extends AbstractController
11 {
12     #[Route('/weather', name: 'app_weather')]
13     public function index(): Response
14     {
15         return $this->render( view: 'weather/index.html.twig', [
16             'controller_name' => 'WeatherController',
17         ]);
18     }
19 }

```

Utworzone zostały także pliki widoku:

- templates/base.html.twig
- templates/weather/index.html.twig

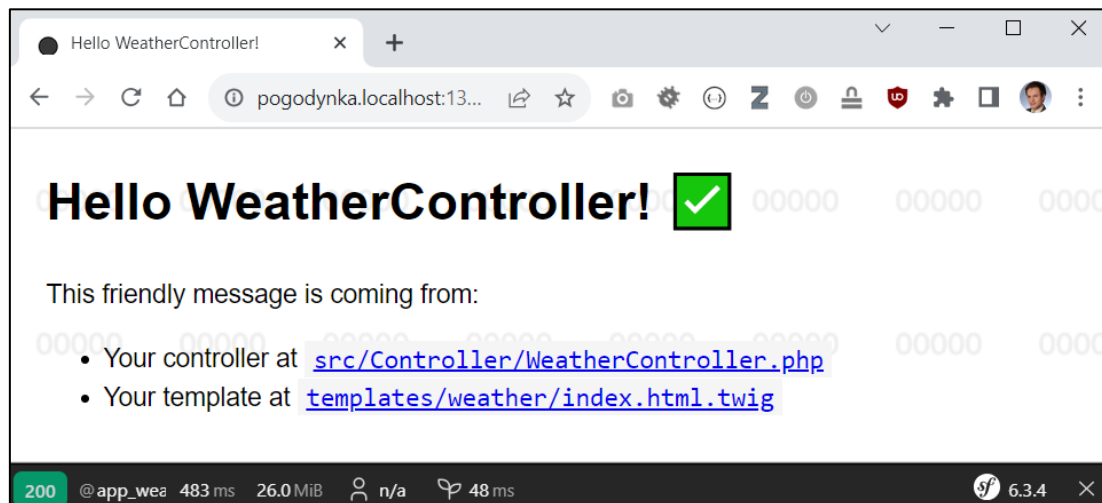
Zmodyfikuj plik templates/base.html.twig poprzez dodanie stylu w <head>, jako text wstawiając swój numer albumu:

```

<style>
    body {
        background: url("https://placeholder.co/100x100/FFFFFF/EFEFEF/png?text=123");
    }
</style>

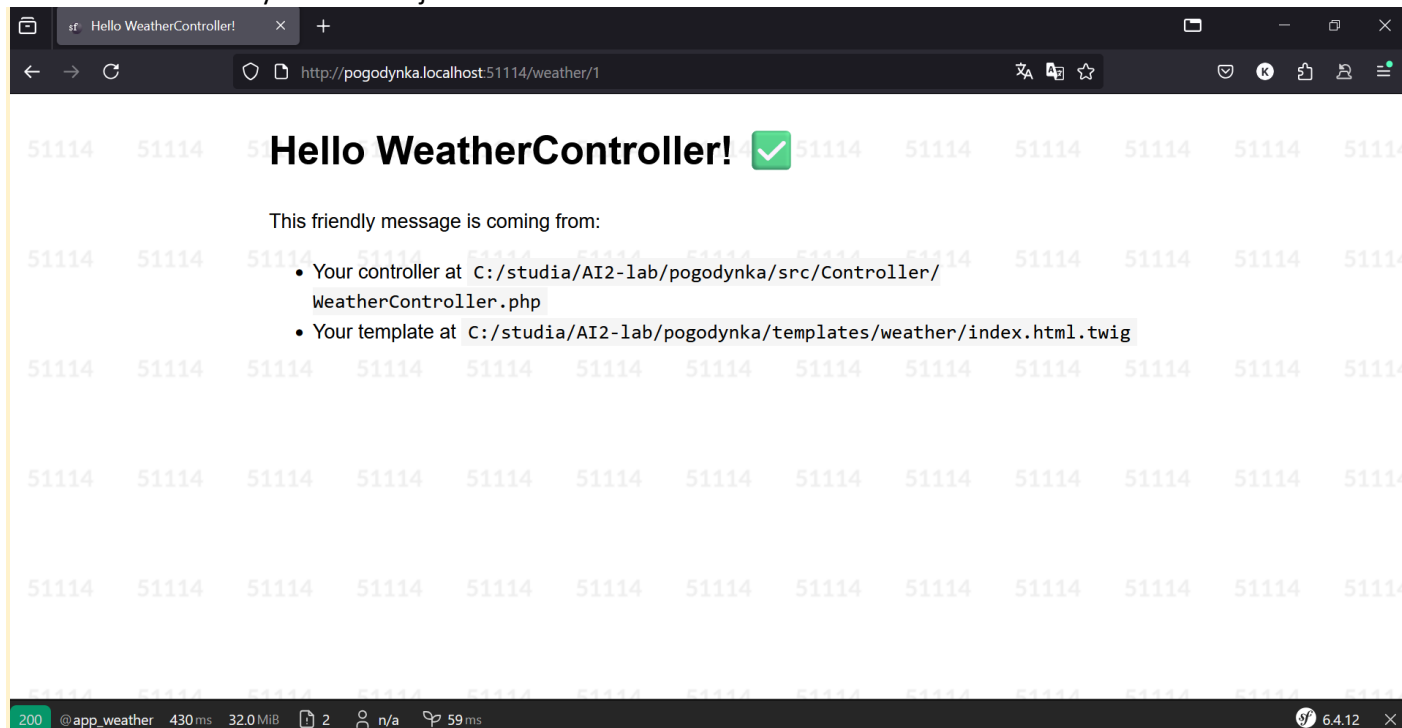
```

Akcję kontrolera można podejrzeć teraz w przeglądarce pod adresem <http://pogodynka.localhost:51114/weather>:



Na koniec zmień nazwę akcji kontrolera z index na city, a ścieżkę z /weather na /weather/{id}. Na ten moment wymuś, aby parametr id mógł być wyłącznie \d+.

Wstaw poniżej zrzut ekranu strony /weather/1, uwzględniający pasek adresu oraz tło z numerem indeksu:



Wstaw poniżej zrzut ekranu kodu kontrolera:

```
1 <?php
2
3 namespace App\Controller;
4
5 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
6 use Symfony\Component\HttpFoundation\Response;
7 use Symfony\Component\Routing\Attribute\Route;
8
9 class WeatherController extends AbstractController
10 {
11     #[Route('/weather/{id}', name: 'app_weather', requirements: ['id' => '\d+'])]
12     public function city(): Response
13     {
14         return $this->render('weather/index.html.twig', [
15             'controller_name' => 'WeatherController',
16         ]);
17     }
18 }
```

Punkty:

0

1

REPOZYTORIUM

Zmodyfikujemy teraz naszą akcję w taki sposób, żeby pobierała dane. Otwórz w IDE plik `src/Repository/MeasurementRepository.php` i dodaj do niego metodę `findByLocation`:

```
public function findByLocation(Location $location)
{
    $qb = $this->createQueryBuilder('m');
    $qb->where('m.location = :location')
        ->setParameter('location', $location)
        ->andWhere('m.date > :now')
        ->setParameter('now', date('Y-m-d'));

    $query = $qb->getQuery();
    $result = $query->getResult();
    return $result;
}
```

Zmodyfikuj także kontroler, aby:

- automatycznie pobierał obiekt klasy `Location` na podstawie identyfikatora ze ścieżki URL;
- wykorzystywał metodę `findByLocation` do pobrania prognozy pogody dla zadanej lokacji;
- przekazywał informacje o lokacji i pobrane prognozy pogody na widok.

Przykładowo:

```
#[Route('/weather/{id}', name: 'app_weather', requirements: ['id' => '\d+'])]
public function city(Location $location, MeasurementRepository $repository): Response
{
    $measurements = $repository->findByLocation($location);

    return $this->render('weather/city.html.twig', [
        'location' => $location,
        'measurements' => $measurements,
    ]);
}
```

Na koniec edytuj widok (zmień `weather/index.html.twig` na `weather/city.html.twig`), aby wyświetlić informacje o lokacji i prognozę pogody:

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

{% @var location \App\Entity\Location %}
{% @var weather \App\Entity\Weather %}

{% block title %}Weather in {{ location.city }}, {{ location.country }}{% endblock %}

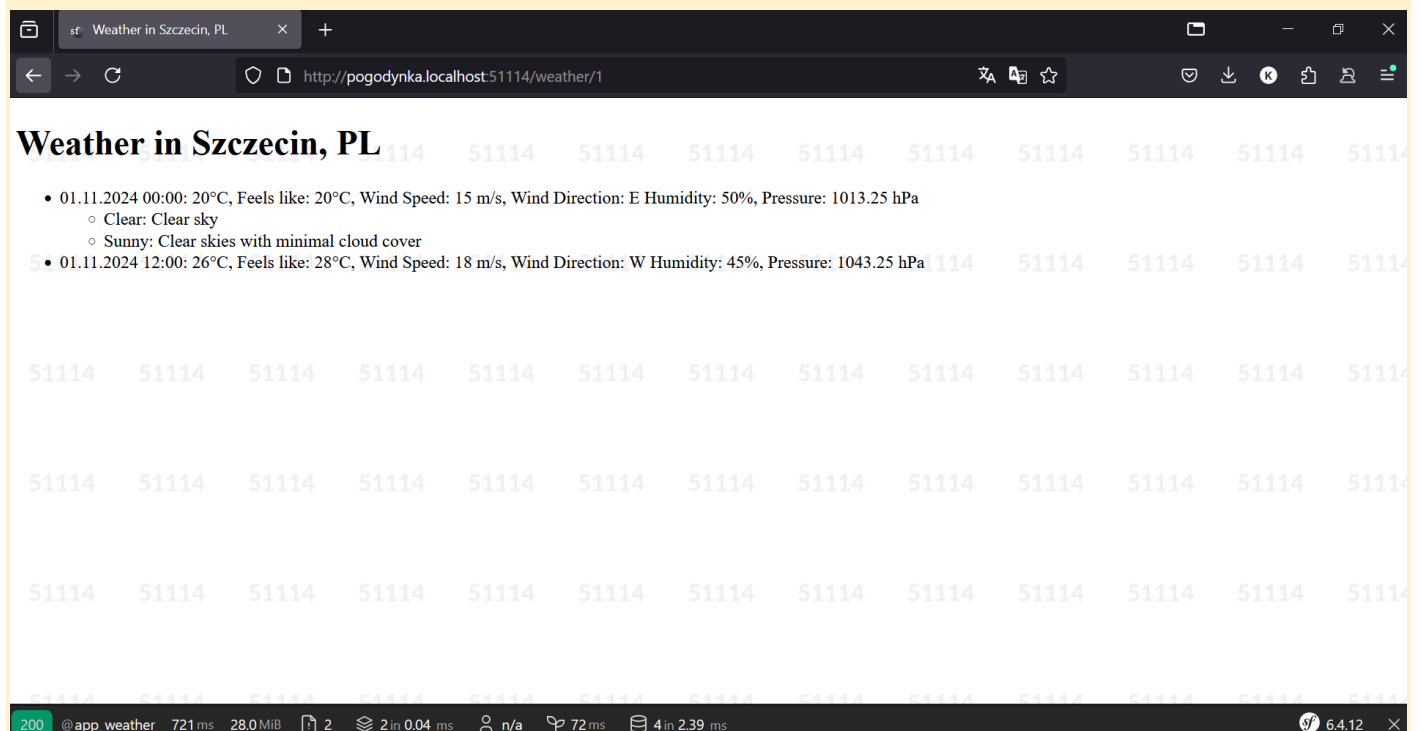
{% block body %}
<main>
    <h1>Weather in {{ location.city }}, {{ location.country }}</h1>

    <ul>
        {% for measurement in measurements %}
            <li>{{ measurement.date|date('d.m.Y') }}: {{ measurement.celsius }}&deg;C</li>
        {% endfor %}
    </ul>
</main>
{% endblock %}
```

Oczekiwany efekt:



Wstaw zrzut ekranu wyglądu strony `/weather/...` z prognozą pogody dla pojedynczej lokacji:



Wstaw zrzut ekranu kodu widoku `weather/city.html.twig`:

```

WeatherController.php M  city.html.twig U X
templates > weather > city.html.twig
1  {% extends 'base.html.twig' %}
2
3  {% @var location \App\Entity\Location %}
4  {% @var weather \App\Entity\Weather %}
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     {% if measurement_entries is not empty %}
13         <ul>
14             {% for measurement in measurement_entries %}
15                 <li>
16                     {{ measurement.dateTime|date('d.m.Y H:i') }}:
17                     {{ measurement.temperatureCelcius }}&deg;C,
18                     Feels like: {{ measurement.feelsLike }}&deg;C,
19                     Wind Speed: {{ measurement.windSpeed }} m/s,
20                     Wind Direction: {{ measurement.windDirection }}
21                     Humidity: {{ measurement.humidity }}%,
22                     Pressure: {{ measurement.pressure }} hPa
23
24                     {% if measurement.weatherConditions is not empty %}
25                         <ul>
26                             {% for condition in measurement.weatherConditions %}
27                                 <li>{{ condition.name }}: {{ condition.description }}</li>
28                             {% endfor %}
29                         </ul>
30                     {% endif %}
31                 </li>
32             {% endfor %}
33         </ul>
34     {% else %}
35         <p>No measurement entries available for this location.</p>
36     {% endif %}
37 </main>
38 {% endblock %}
39

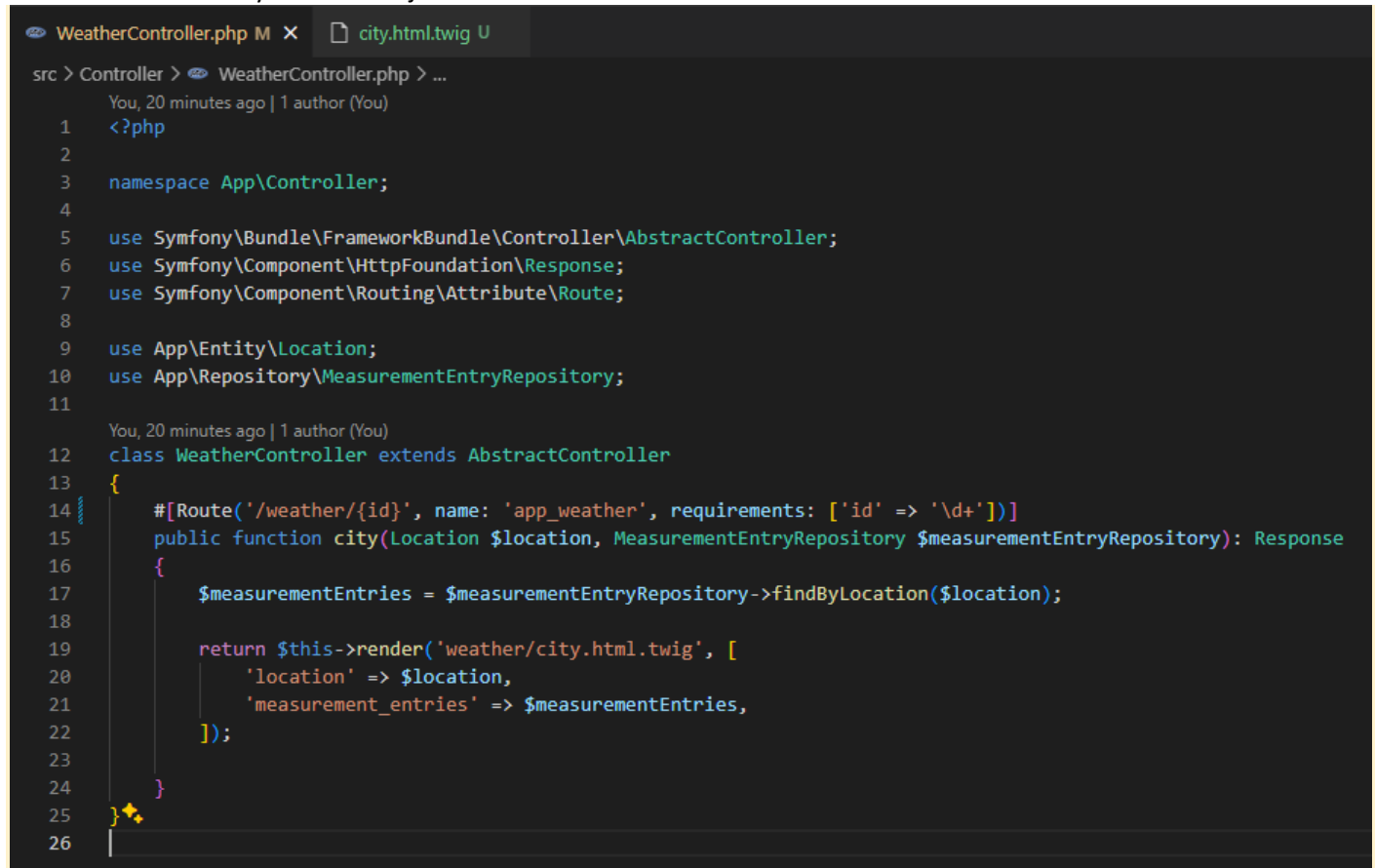
```

Punkty:

0

1

Wstaw zrzut ekranu kodu kontrolera WeatherController:



```
src > Controller > WeatherController.php > ...  
You, 20 minutes ago | 1 author (You)  
1 <?php  
2  
3 namespace App\Controller;  
4  
5 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
6 use Symfony\Component\HttpFoundation\Response;  
7 use Symfony\Component\Routing\Attribute\Route;  
8  
9 use App\Entity\Location;  
10 use App\Repository\MeasurementEntryRepository;  
11  
12 You, 20 minutes ago | 1 author (You)  
13 class WeatherController extends AbstractController  
14 {  
15     #[Route('/weather/{id}', name: 'app_weather', requirements: ['id' => '\d+'])]  
16     public function city(Location $location, MeasurementEntryRepository $measurementEntryRepository): Response  
17     {  
18         $measurementEntries = $measurementEntryRepository->findByLocation($location);  
19  
20         return $this->render('weather/city.html.twig', [  
21             'location' => $location,  
22             'measurement_entries' => $measurementEntries,  
23         ]);  
24     }  
25 }  
26
```

Wstaw zrzut ekranu kodu repozytorium `MeasurementRepository`:

```

src > Repository > MeasurementEntryRepository.php > ...
You, 24 seconds ago | 1 author (You)
1  <?php
2
3  namespace App\Repository;
4
5  use App\Entity\Location;
6  use App\Entity\MeasurementEntry;
7  use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
8  use Doctrine\Persistence\ManagerRegistry;
9
You, 24 seconds ago | 1 author (You)
10 /**
11  * @extends ServiceEntityRepository<MeasurementEntry>
12  */
You, 24 seconds ago | 1 author (You)
13 class MeasurementEntryRepository extends ServiceEntityRepository
14 {
15     public function __construct(ManagerRegistry $registry)
16     {
17         parent::__construct($registry, MeasurementEntry::class);
18     }
19
20     public function findByLocation(Location $location)
21     {
22         $gb = $this->createQueryBuilder('m');
23         $gb->where('m.location = :location')
24             ->setParameter('location', $location)
25             ->andWhere('m.dateTime > :now')
26             ->setParameter('now', new \DateTime('now'));
27
28         $query = $gb->getQuery();
29         $result = $query->getResult();
30
31         return $result;
32     }
33
34
35 }
36

```

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

WYSZUKIWANIE LOKACJI PO NAZWIE MIASTA

Zmodyfikuj kod akcji `WeatherController::city()` w taki sposób, żeby przyjmowała w ścieżce parametr z nazwą miejscowości (i opcjonalnie kodem państwa) zamiast parametru ID.

Wstaw zrzut ekranu kodu zmodyfikowanego kontrolera:

```

8
9 use App\Repository\LocationRepository;
10 use App\Repository\MeasurementEntryRepository;
11
12 use Symfony\Bridge\Doctrine\Attribute\MapEntity;
13
14 You, 14 seconds ago | 1 author (You)
15 class WeatherController extends AbstractController
16 {
17     #[Route('/weather/{city}/{country?}', name: 'app_weather', requirements: ['city' => '[a-zA-Z-]+', 'country' => '[a-zA-Z]{2}'])]
18     public function city(
19         // nie wiem czy da się zrobić tak żeby country było opcjonalne w tym miejscu
20         // #[MapEntity(mapping: ['city' => 'city', 'country' => 'country'])]
21         // Location $location,
22         string $city,
23         ?string $country,
24         MeasurementEntryRepository $measurementEntryRepository,
25         LocationRepository $locationRepository,
26     ): Response
27     {
28         $city = str_replace('-', ' ', $city);
29         $city = ucfirst($city);
30
31         $location = null;
32         if ($country) {
33             $country = strtoupper($country);
34
35             $location = $locationRepository->findOneByCityAndCountry($city, $country);
36         } else {
37             $location = $locationRepository->findOneByCity($city);
38         }
39
40
41
42
43         $measurementEntries = $measurementEntryRepository->findByLocation($location);
44
45         return $this->render('weather/city.html.twig', [
46             'location' => $location,
47             'measurement_entries' => $measurementEntries,
48         ]);
49     }
50 }
51
52

```

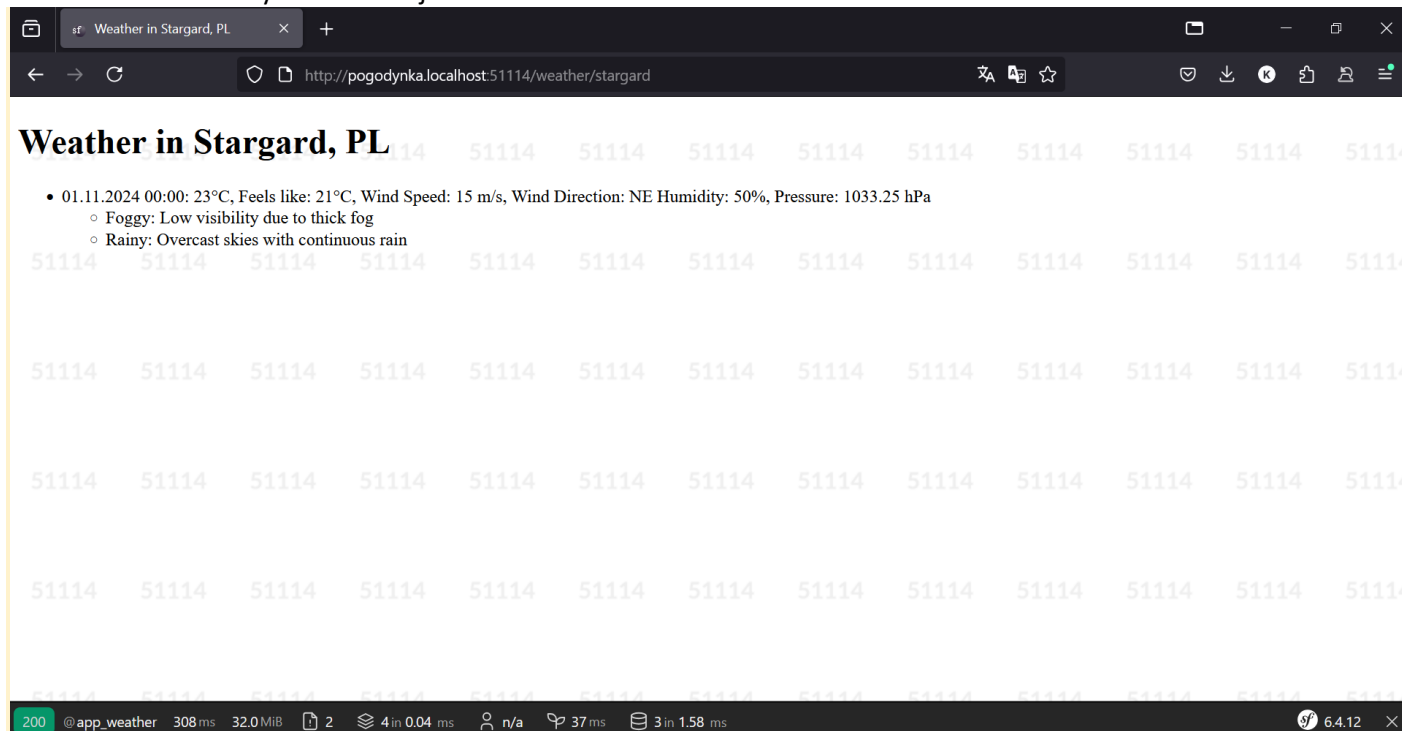
Wstaw zrzuty ekranu kodu zmodyfikowanych repozytoriów:

```

WeatherController.php M  LocationRepository.php M X
src > Repository > LocationRepository.php > PHP Intelephense > LocationRepository > findOneByCity
You, 1 second ago | 1 author (You)
1  <?php
2
3  namespace App\Repository;
4
5  use App\Entity\Location;
6  use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
7  use Doctrine\Persistence\ManagerRegistry;
8
9  /**
10   * @extends ServiceEntityRepository<Location>
11   */
12  class LocationRepository extends ServiceEntityRepository
13  {
14      public function __construct(ManagerRegistry $registry)
15      {
16          parent::__construct($registry, Location::class);
17      }
18
19      public function findOneByCityAndCountry(string $city, string $country): ?Location
20      {
21          return $this->createQueryBuilder('l')
22              ->andWhere('l.city = :city')
23              ->andWhere('l.country = :country')
24              ->setParameter('city', $city)
25              ->setParameter('country', $country)
26              ->getQuery()
27              ->getOneOrNullResult()
28      };
29  }
30
31  public function findOneByCity(string $city): ?Location
32  {
33      return $this->createQueryBuilder('l')
34          ->andWhere('l.city = :city')
35          ->setParameter('city', $city)
36          ->getQuery()
37          ->getOneOrNullResult()
38      ;
39  }
40  }
41
You, 1 second ago • Uncommitted changes

```

Wstaw zrzut ekranu wynikowej strony pod adresem uwzględniającym nazwę miasta:



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-c` na podstawie głównej gałęzi kodu.

Podaj link do brancha `lab-c` w swoim repozytorium:

<https://github.com/szvbvtk/ai2-pogodynka/tree/lab-c>

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

Podczas tego laboratorium zdobyłem umiejętności związane generowaniem encji oraz definiowaniem kontrolerów wraz z routingiem.

Dowiedziałem się również jak aktualizować schemat bazy danych na podstawie encji za pomocą Doctrine.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.