

BAZA DANYCH I OPERACJE ODCZYTU

SPIS TREŚCI

Spis treści	1
Cel zajęć.....	1
Rozpoczęcie.....	1
Uwaga	1
Encja Location	2
Pozostałe encje.....	5
Akcja kontrolera	9
Pobieranie danych przez repozytorium.....	11
Wyszukiwanie lokacji po nazwie miasta.....	15
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. W szczególności zaktualizuj ścieżkę do swojego projektu pogodynka. 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 `C:\Users\akarczmarczyk\Desktop\ai2-labs\pogodynka` w PhpStorm / VS Code. W pliku `.env` zmień bazę danych na SQLite:

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

Ten wpis oznacza, że aplikacja będzie korzystać z bazy danych SQLite umieszczonej w pliku `C:\Users\akarczmarczyk\Desktop\ai2-labs\pogodynka\var\data.db`.

Uwaga! Plik `.env` jest analizowany przez Symfony od góry do dołu. W poprzednich edycjach kursu studenci często mimo odkomentowania linijki z SQLite, zapominali zakomentować linijkę z domyślnym PostgreSQL. To skutkowało nadpisaniem zmiennej środowiskowej `DATABASE_URL`, a w efekcie ignorowaniem bazy danych SQLite. Ponieważ w `php.ini` nie włączliśmy rozszerzenia PDO dla PostgreSQL, w takim przypadku wizualnie w przeglądarce wyskakuje błąd o braku sterownika.

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

```
cd C:\Users\akarczmarczyk\Desktop\ai2-labs\pogodynka
php .\bin\console make:entity

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

Add the ability to broadcast entity updates using Symfony UX Turbo? (yes/no) [no]:
>

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 napisany 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`? **Nie używaj LLM.**

`--dump-sql` tylko wyświetla zapytania SQL, które Doctrine mogłoby wykonać, ale ich nie uruchamia.

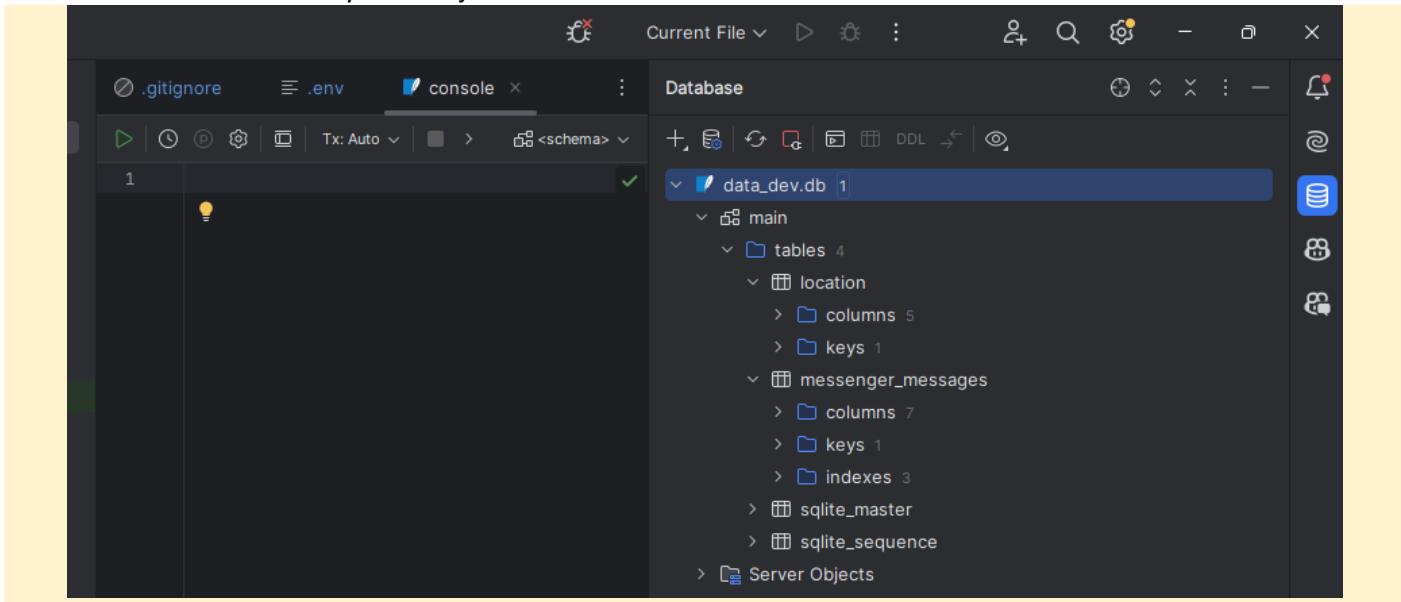
`--force` natomiast faktycznie wykonuje te zapytania i aktualizuje schemat bazy danych.

Umieść zrzut ekranu lub skopiuj SQL, który został wygenerowany:

```
Windows PowerShell  
PS D:\Semestr 7\ai2-labs\pogodynka> php bin\console doctrine:schema:update --dump-sql  
CREATE TABLE location (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, city VARCHAR(255) NOT NULL, country VARCHAR(2) NOT N  
ULL, 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_75EA56E0FB7336F0 ON messenger_messages (queue_name);  
CREATE INDEX IDX_75EA56E0E3BD61CE ON messenger_messages (available_at);  
CREATE INDEX IDX_75EA56E016BA31DB ON messenger_messages (delivered_at);  
PS D:\Semestr 7\ai2-labs\pogodynka> php bin\console doctrine:schema:update --dump-sql --force  
CREATE TABLE location (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, city VARCHAR(255) NOT NULL, country VARCHAR(2) NOT N  
ULL, 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_75EA56E0FB7336F0 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

`C:\Users\akarczmarczyk\Desktop\ai2-labs\pogodynka\var\data_dev.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. GrumpyPopsicle):
> Measurement

Add the ability to broadcast entity updates using Symfony UX Turbo? (yes/no) [no]:
>

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]:  
>
```

```
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!

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

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

```
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

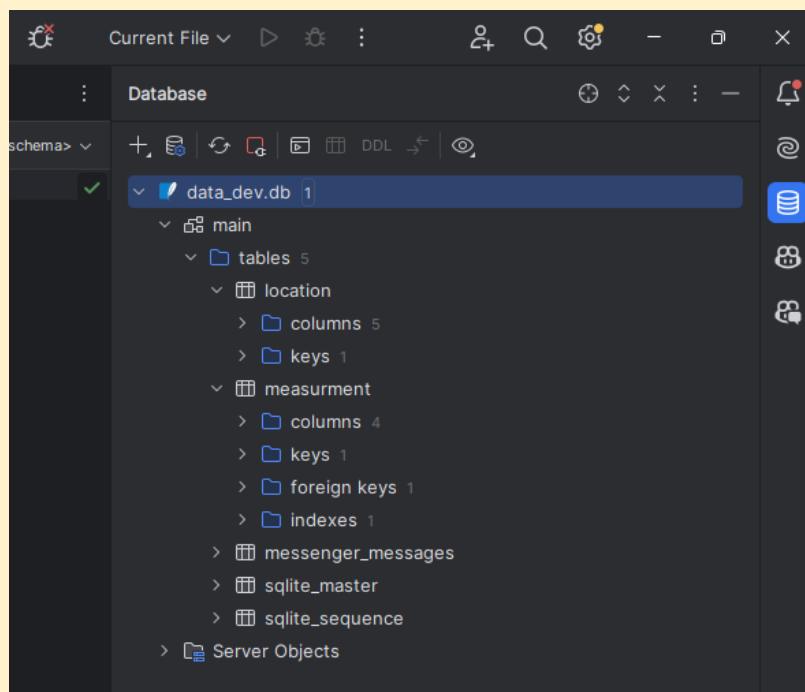
PS D:\Semestr 7\ai2-labs\pogodynka> php bin\console doctrine:schema:update --dump-sql --force
CREATE TABLE measurement (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, location_id INTEGER NOT NULL, date DATE NOT NULL, celsius NUMERIC(3, 0) NOT NULL, CONSTRAINT FK_E31DB30A64D218E FOREIGN KEY (location_id) REFERENCES location (id) NOT DEFERRABLE INITIALLY IMMEDIATE);
CREATE INDEX IDX_E31DB30A64D218E ON measurement (location_id);

Updating database schema...
2 queries were executed

[OK] Database schema updated successfully!

PS D:\Semestr 7\ai2-labs\pogodynka> |
```

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]

AKCJA KONTROLERA

Utwórz pusty kontroler z wykorzystaniem komendy:

```
pogodynka> php .\bin\Console make:controller
Choose a name for your controller class (e.g. GentleKangarooController):
> WeatherController

Do you want to generate PHPUnit tests? [Experimental] (yes/no) [no]:
>

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  no usages
8
9  final class WeatherController extends AbstractController
10 {
11     #[Route('/weather', name: 'app_weather')]
12     public function index(): Response
13     {
14         return $this->render('weather/index.html.twig', [
15             'controller_name' => 'WeatherController',
16         ]);
17     }
18 }
```

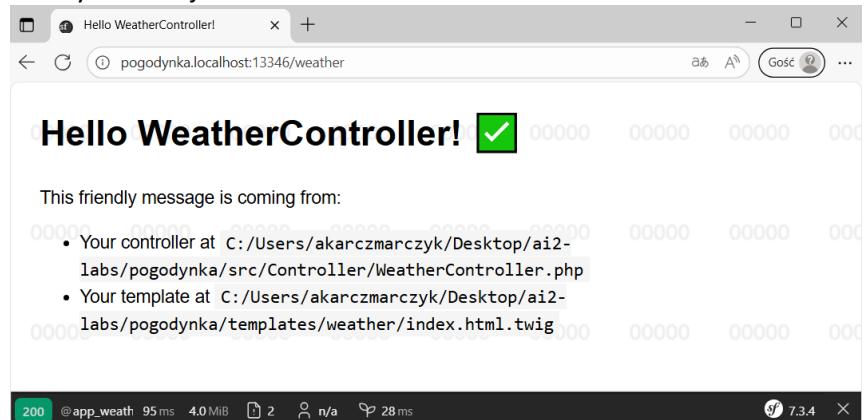
Utworzone zostały także pliki widoku:

- `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:

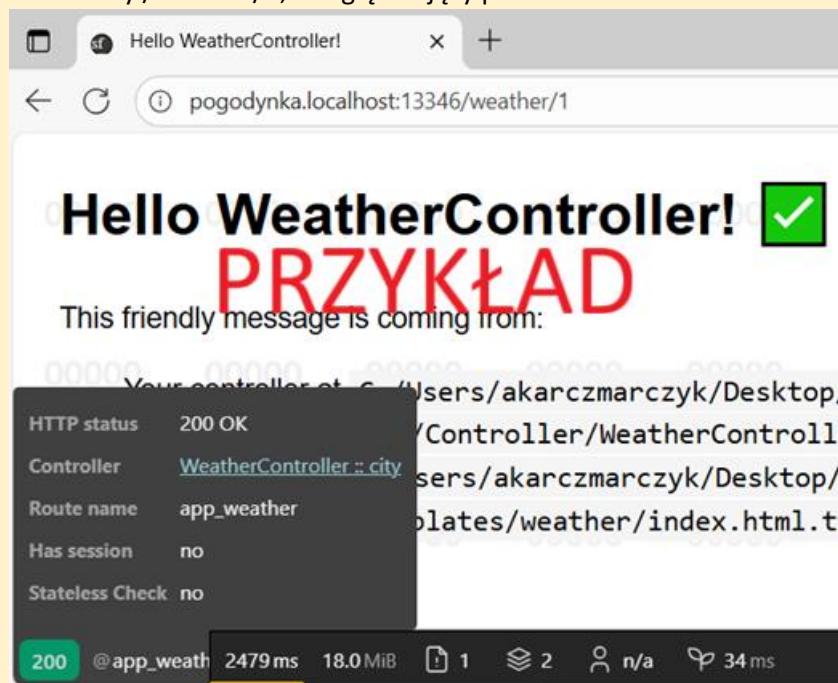
```
<style>
    body {
        background: url("https://placehold.co/100x100/FFFFFF/EFEFEF/png?text=0000");
    }
</style>
```

Akcję kontrolera można podejrzeć teraz w przeglądarce pod adresem `http://pogodynka.localhost:00000/weather`:

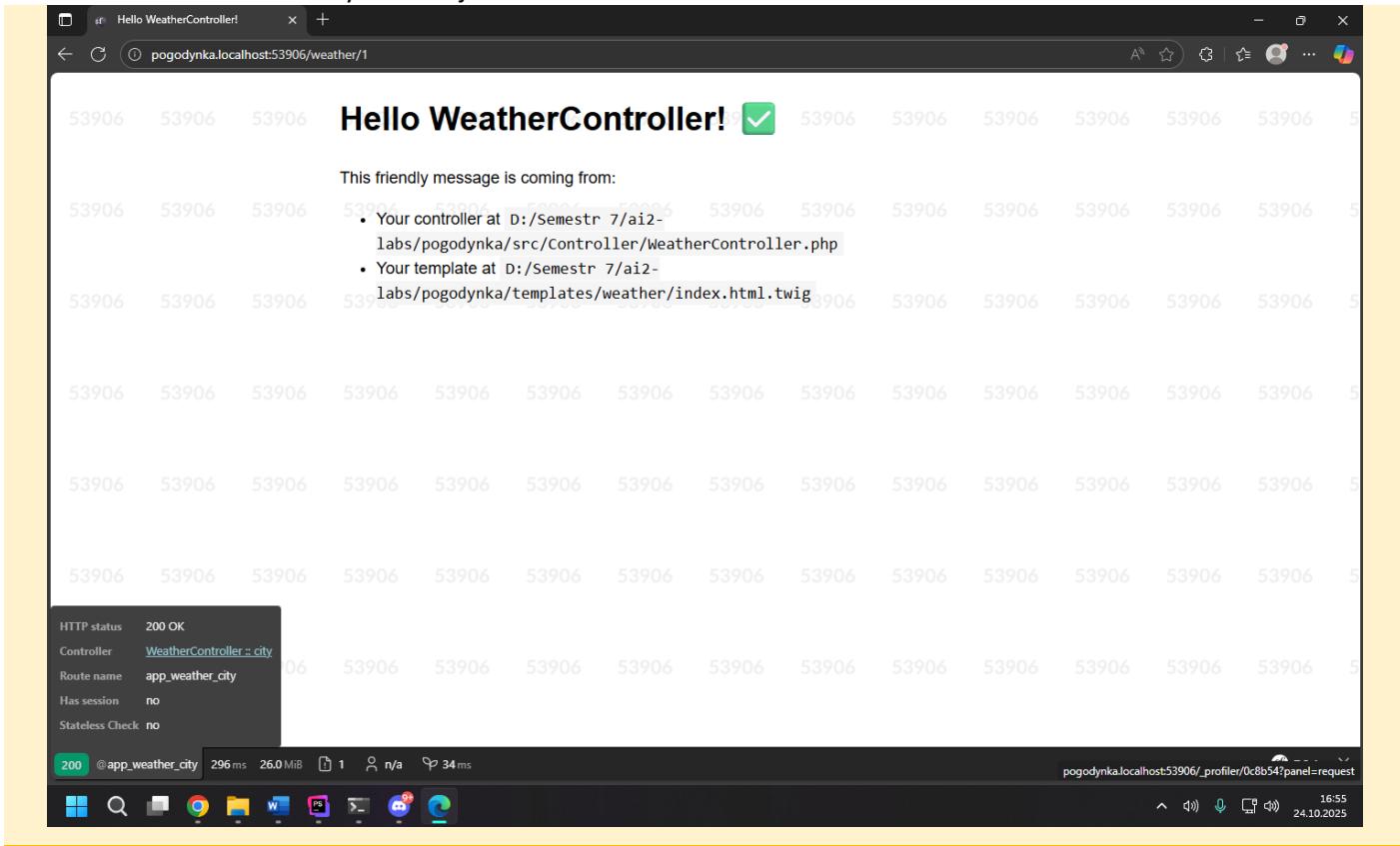


Na koniec zmień nazwę akcji kontrolera z `index` na `city`, a ścieżkę z `/weather` na `/weather/{id}`. Na ten moment wymuś, aby parametr `id` mógł być wyłącznie `\d+`.

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



Wstaw poniżej zrzut ekranu kodu kontrolera:



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

POBIERANIE DANYCH PRZEZ REPOZYTORIUM

Zmodyfikujmy 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;
}
```

Uwaga! Jeśli po *przepisaniu* powyższego kodu w IDE `Location` oznaczone jest jako błąd, najprawdopodobniej nie zimportowano klasy `Location` do pliku. Wystarczy najechać myszką na podświetlony fragment, a następnie kliknąć **Import class**:

The screenshot shows a portion of a PHP file in PhpStorm. A tooltip is displayed over the word 'Location' at line 19, which is part of the method signature 'public function findByLocation (Location \$location)'. The tooltip contains the message 'Undefined class 'Location''. Below this, there is an 'Import class' button with a keyboard shortcut 'Alt+Shift+Enter'. The tooltip also shows the full path 'Location: \App\Repository\Location' and the source file 'Source: .../src/Repository/MeasurementRe...'. The code block itself includes several annotations and imports.

```

19     public function findByLocation (Location $location)
20     {
21         $qb = $this->createQueryBuilder('m');
22         $qb->where('m.location = :location')
23             ->setParameter('location', $location);
24         $qb->andWhere('m.date > :now')
25             ->setParameter('now', date('Y-m-d'));
26
27         $query = $qb->getQuery();
28         $result = $query->getResult();
29
30         return $result;
31     }

```

Zmodyfikuj 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 do widoku.

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 %}

```

Porada. To świetny moment, żeby w PhpStorm zainstalować i włączyć dla projektu plugin Symfony. Dzięki temu m.in. uzyskamy podpowiadanie składni obiektów w Twigu:

```
3  #@var location \App\Entity\Location #}
4  #@var weather \App\Entity\Weather #}
5
6 @@ {%- block title %}Weather in {{ location.city }}, {{ lo
7
8 @@ {%- block body %}
```

(m) city()

Press Enter to insert, Tab to move

Oczekiwany efekt:

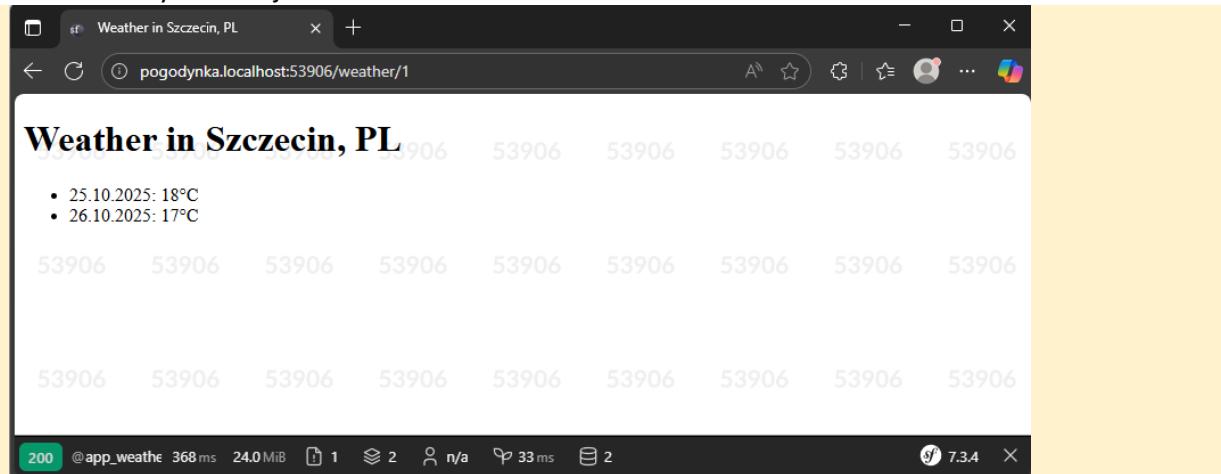


Jak rozwiązać **Could not convert database value "2025-10-05 00:00:00.000" to Doctrine Type date. Expected format: Y-m-d**? W ustawieniach połączenia z DB w PhpStorm, zakładka Advanced, zmienić `date_string_format` na `yyyy-MM-dd` i ponownie zapisać wszystkie daty w tabeli `measurement` (lub odpowiedniku).

Name	Value
case_sensitive_like	
count_changes	
date_class	TEXT
date_precision	
date_string_format	yyyy-MM-dd

Nie wyświetla się żadne pomiary temperatury? A czy daty wpisów do tabeli `measurement` (lub odpowiedniku) są w przyszłości? Por. `MeasurementRepository::findByLocation()`.

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



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

```
%> extends 'base.html.twig' %>
{# @var location \App\Entity\Location #}
{# @var measurement \App\Entity\Measurement #}

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

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

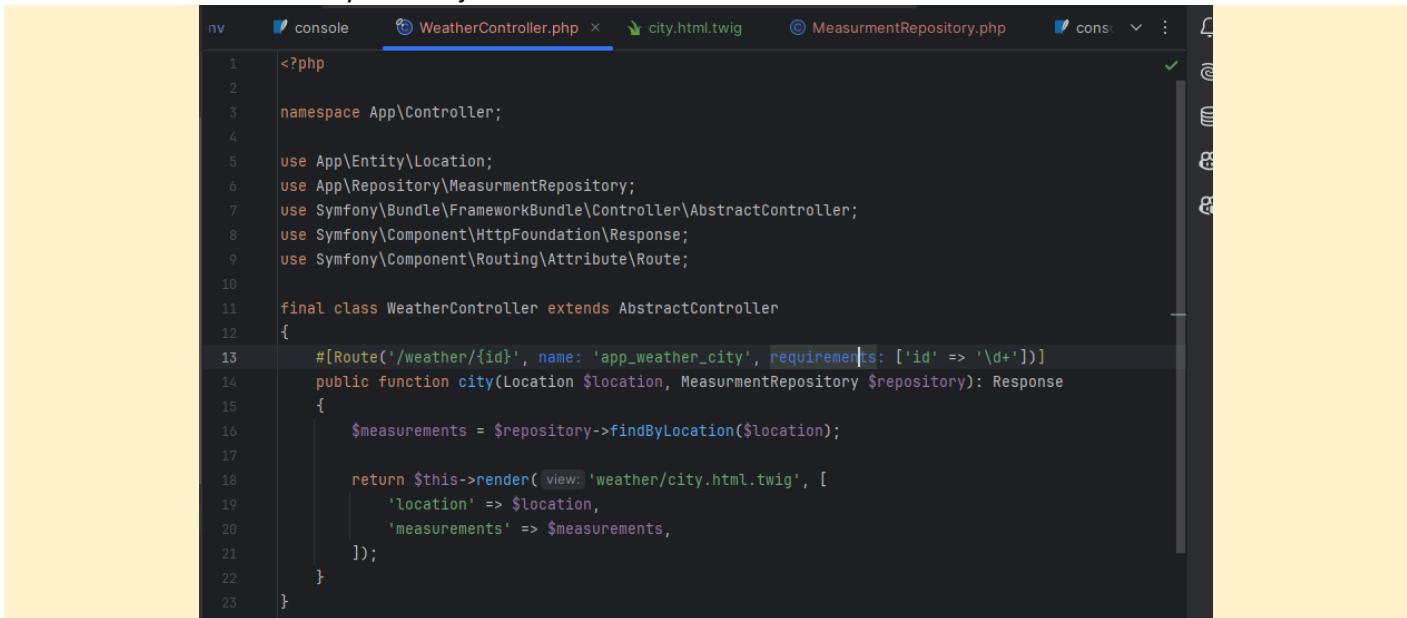
        <ul>
            {% for measurement in measurements %}
                <li>{{ measurement.date|date('d.m.Y') }}: {{ measurement.celsius }}&deg;C</li>
            {% else %}
                <li>No weather data available.</li>
            {% endif %}
        </ul>
    </main>
{%- endblock %}
```

Punkty:

0

1

Wstaw zrzut ekranu kodu kontrolera `WeatherController`:

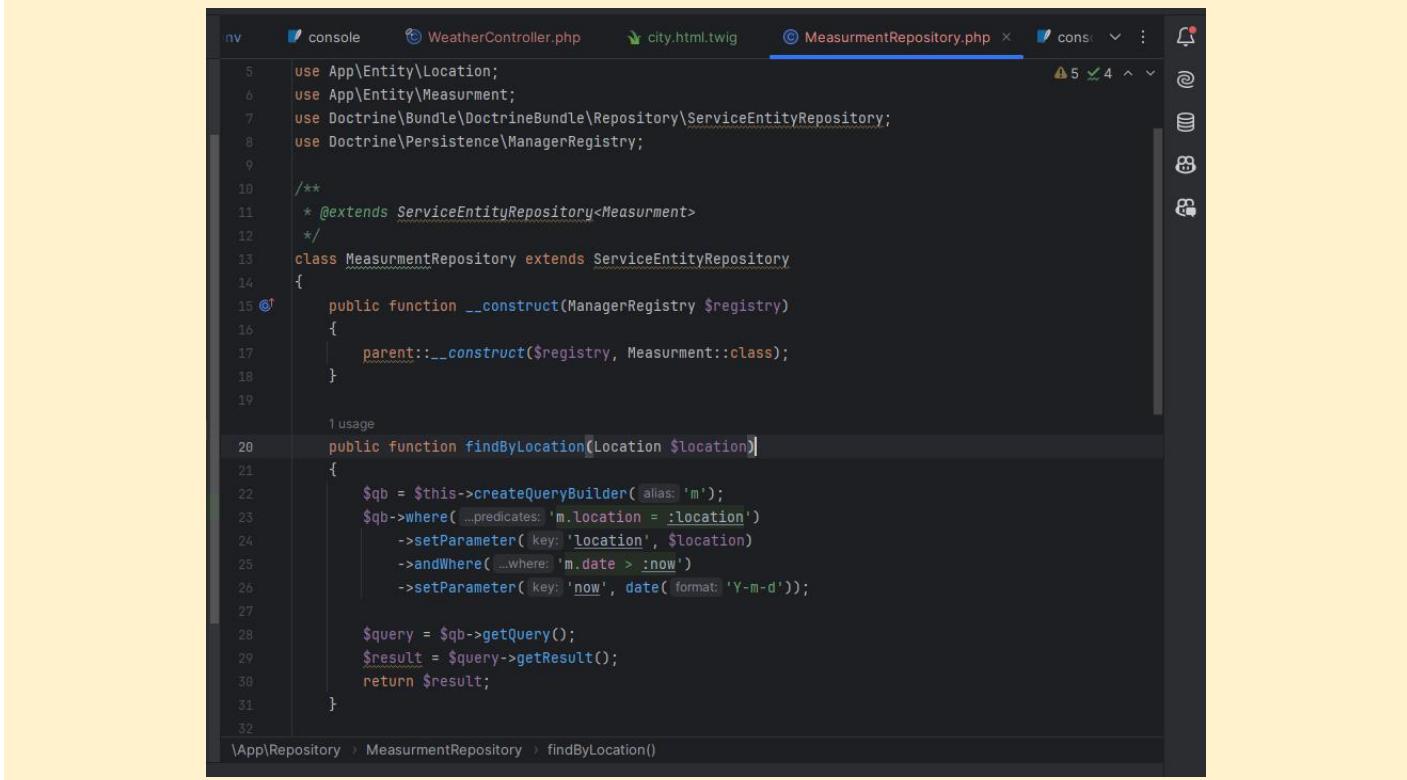


```

1 <?php
2
3 namespace App\Controller;
4
5 use App\Entity\Location;
6 use App\Repository\MeasurmentRepository;
7 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
8 use Symfony\Component\HttpFoundation\Response;
9 use Symfony\Component\Routing\Annotation\Route;
10
11 final class WeatherController extends AbstractController
12 {
13     #[Route('/weather/{id}', name: 'app_weather_city', requirements: ['id' => '\d+'])]
14     public function city(Location $location, MeasurmentRepository $repository): Response
15     {
16         $measurements = $repository->findByLocation($location);
17
18         return $this->render('weather/city.html.twig', [
19             'location' => $location,
20             'measurements' => $measurements,
21         ]);
22     }
23 }

```

Wstaw zrzut ekranu kodu repozytorium MeasurementRepository:



```

5 use App\Entity\Location;
6 use App\Entity\Measurment;
7 use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
8 use Doctrine\Persistence\ManagerRegistry;
9
10 /**
11 * @extends ServiceEntityRepository<Measurment>
12 */
13 class MeasurmentRepository extends ServiceEntityRepository
14 {
15     public function __construct(ManagerRegistry $registry)
16     {
17         parent::__construct($registry, Measurment::class);
18     }
19
20     1 usage
21     public function findByLocation(Location $location)
22     {
23         $qb = $this->createQueryBuilder('m');
24         $qb->where(...predicates: 'm.location = :location')
25             ->setParameter('location', $location)
26             ->andWhere(...where: 'm.date > :now')
27             ->setParameter('now', date(format: 'Y-m-d'));
28
29         $query = $qb->getQuery();
30         $result = $query->getResultSet();
31         return $result;
32     }
33 }

```

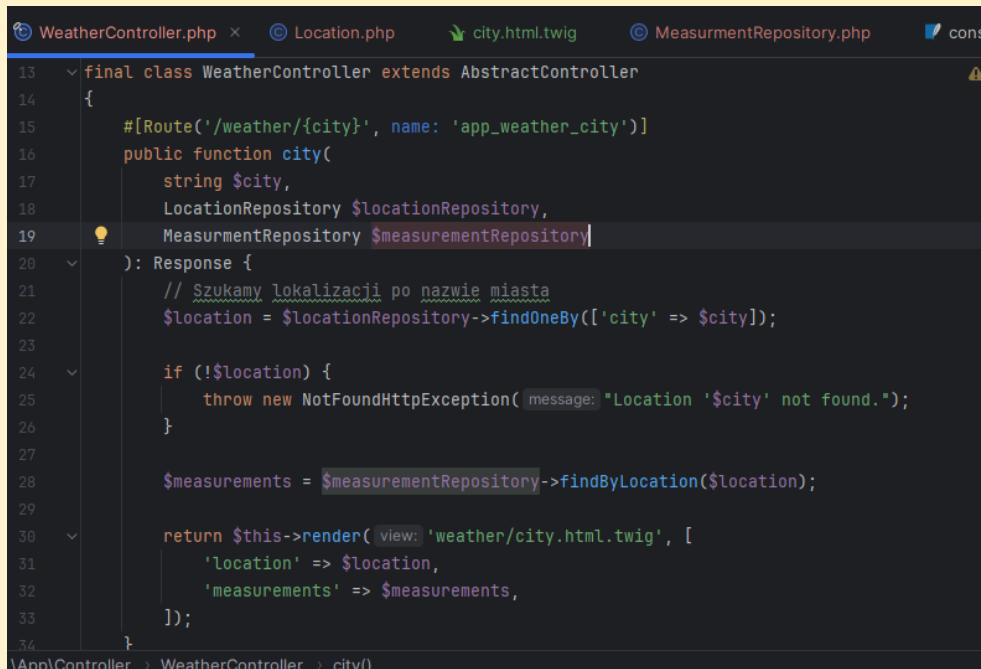
Punkty:	0	1
---------	---	---

WYSZUKIWANIE LOKACJI PO NAZWIE MIASTA

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

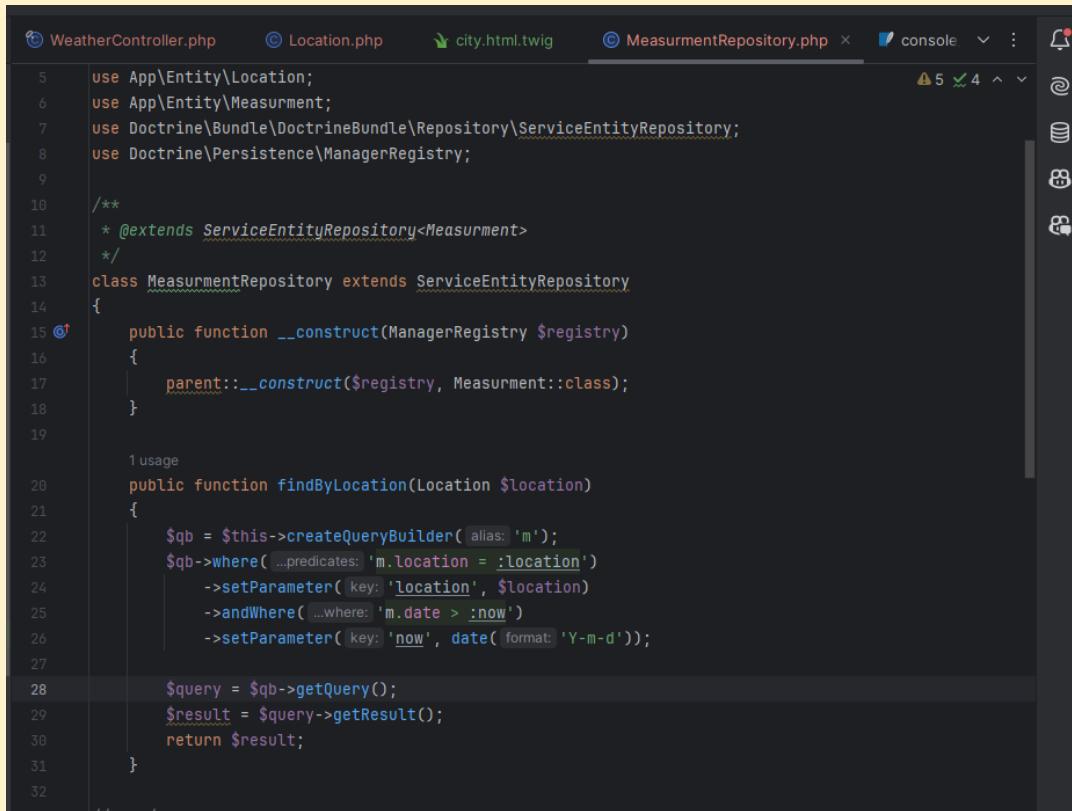
Warto poczytać: <https://symfony.com/doc/current/doctrine.html#automatically-fetching-objects-entity-value-resolver>

Wstaw zrzut ekranu kodu zmodyfikowanego kontrolera:



```
13 final class WeatherController extends AbstractController
14 {
15     #[Route('/weather/{city}', name: 'app_weather_city')]
16     public function city(
17         string $city,
18         LocationRepository $locationRepository,
19         MeasurmentRepository $measurementRepository
20     ): Response {
21         // Szukamy lokalizacji po nazwie miasta
22         $location = $locationRepository->findOneBy(['city' => $city]);
23
24         if (!$location) {
25             throw new NotFoundHttpException('Location "'.$city.'" not found.');
26         }
27
28         $measurements = $measurementRepository->findByLocation($location);
29
30         return $this->render('weather/city.html.twig', [
31             'location' => $location,
32             'measurements' => $measurements,
33         ]);
34     }
35 }
```

Wstaw zrzuty ekranu kodu zmodyfikowanych repozytoriów:



```
5 use App\Entity\Location;
6 use App\Entity\Measurment;
7 use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
8 use Doctrine\Persistence\ManagerRegistry;
9
10 /**
11 * @extends ServiceEntityRepository<Measurment>
12 */
13 class MeasurmentRepository extends ServiceEntityRepository
14 {
15     public function __construct(ManagerRegistry $registry)
16     {
17         parent::__construct($registry, Measurment::class);
18     }
19
20     /**
21      * @usage
22      */
23     public function findByLocation(Location $location)
24     {
25         $qb = $this->createQueryBuilder('m');
26         $qb->where('m.location = :location')
27             ->setParameter('location', $location)
28             ->andWhere('m.date > :now')
29             ->setParameter('now', date('Y-m-d'));
30
31         $query = $qb->getQuery();
32         $result = $query->getResult();
33         return $result;
34     }
35 }
```

Wstaw zrzut ekranu wynikowej strony pod adresem uwzględniającym nazwę miasta:

Weather in Szczecin, PL

25.10.2025: 18°C
26.10.2025: 17°C

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

COMMIT PROJEKTU DO GIT

Zacommittuj 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/xDziDza/ai2-labs/tree/lab-c>

PODSUMOWANIE

W kilku słowach/zdaniach napisz swoje przemyślenia odnośnie tego laboratorium. Co było największym wyzwaniem? Co było najciekawsze? Co mogło być lepsze? Nie używaj LLM.



Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.