

BAZA DANYCH I OPERACJE ODCZYTU

SPIS TREŚCI

Spis treści	1
Cel zajęć.....	1
Rozpoczęcie	1
Uwaga	1
Encja Location	2
Pozostałe encje.....	5
Kontroler	9
Repozytorium.....	12
Wyszukiwanie lokacji po nazwie miasta.....	16
Commit projektu do GIT.....	21
Podsumowanie.....	22

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?

...omówienie różnicy...

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, logitude 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);
```

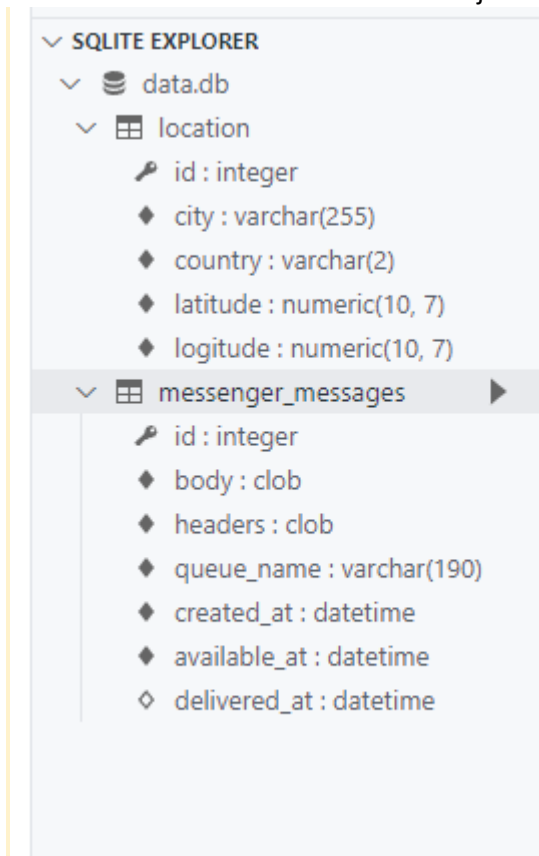
UŻYCIE FORCE robi update na bazie danych

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-lab\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

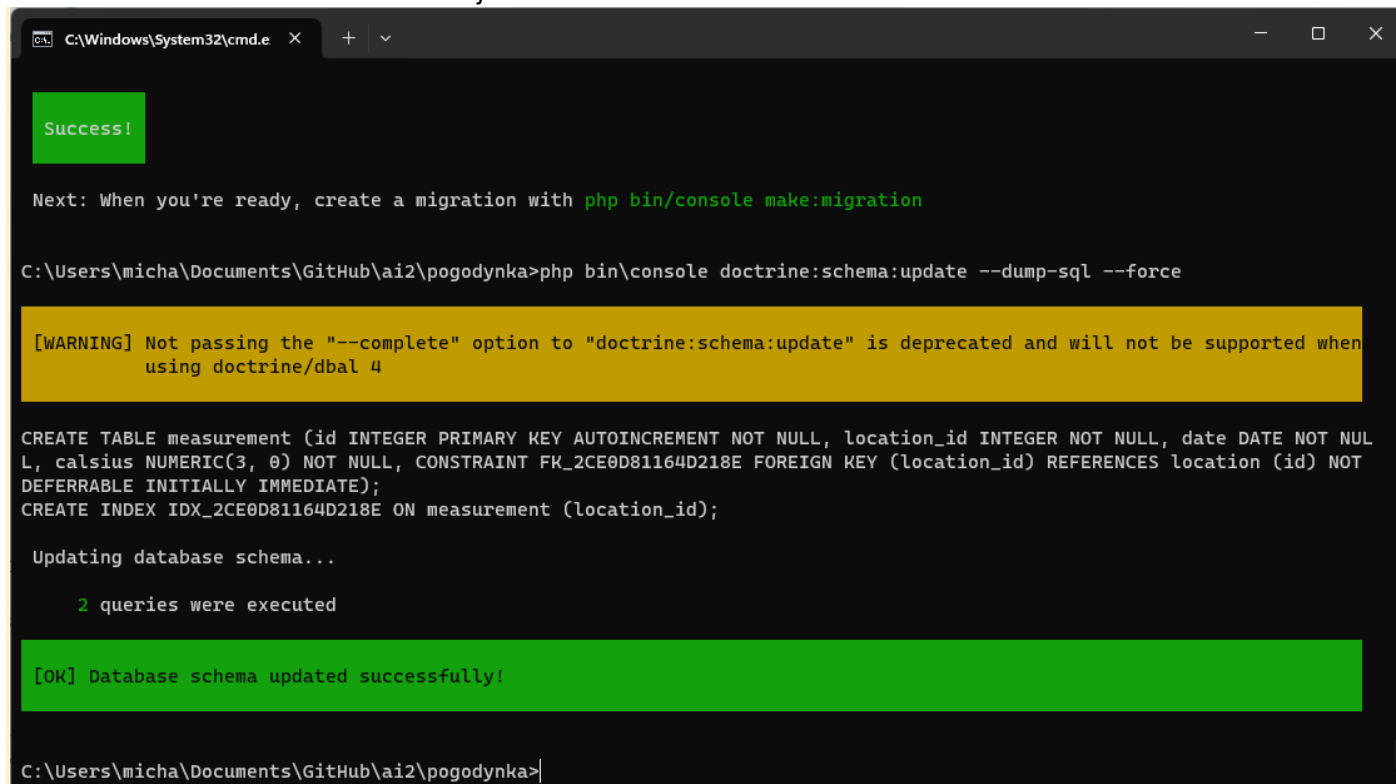
Can this field be null in the database (nullable) (yes/no) [no]:
>

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:



```
C:\Windows\System32\cmd.e x + v

Success!

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

C:\Users\micha\Documents\GitHub\ai2\pogodynka>php bin\console doctrine:schema:update --dump-sql --force

[WARNING] Not passing the "--complete" option to "doctrine:schema:update" is deprecated and will not be supported when
using doctrine/dbal 4

CREATE TABLE measurement (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, location_id INTEGER NOT NULL, date DATE NOT NUL
L, calsius NUMERIC(3, 0) NOT NULL, CONSTRAINT FK_2CE0D81164D218E FOREIGN KEY (location_id) REFERENCES location (id) NOT
DEFERRABLE INITIALLY IMMEDIATE);
CREATE INDEX IDX_2CE0D81164D218E ON measurement (location_id);

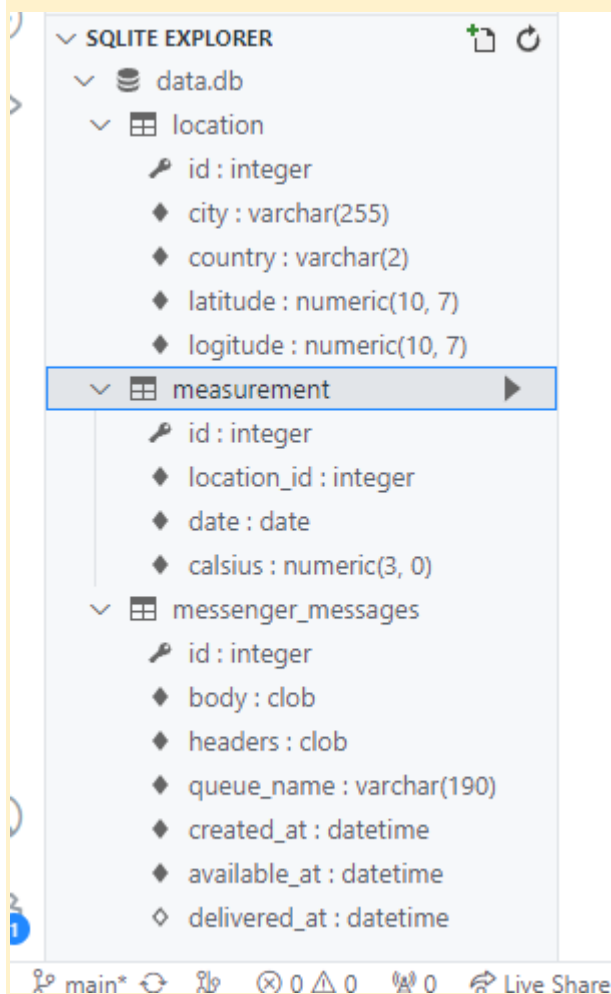
Updating database schema...

 2 queries were executed

[OK] Database schema updated successfully!

C:\Users\micha\Documents\GitHub\ai2\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]

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

Utworzone zostały także pliki widoku:

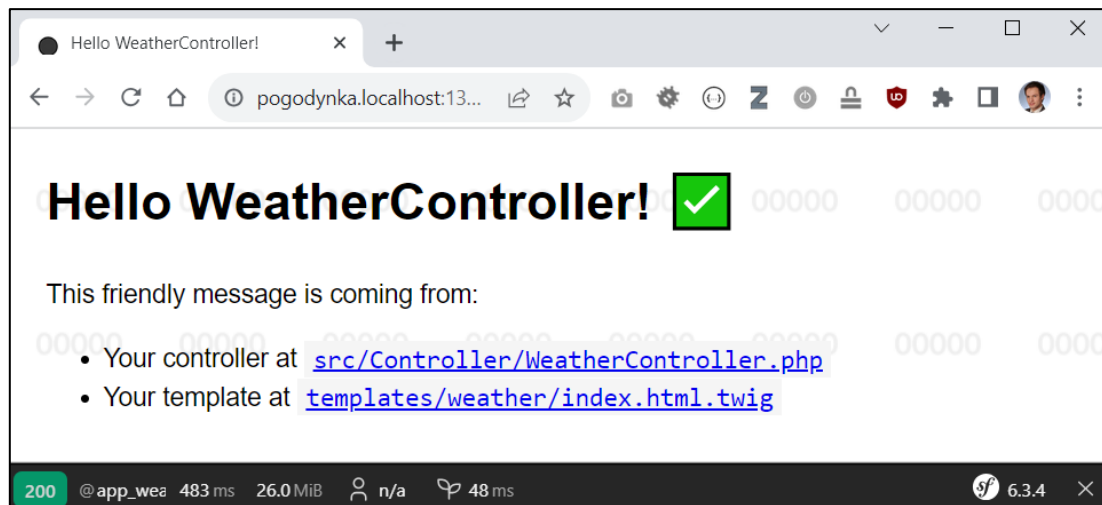
- `templates/base.html.twig`
- `templates/weather/index.html.twig`

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

```
<style>
```

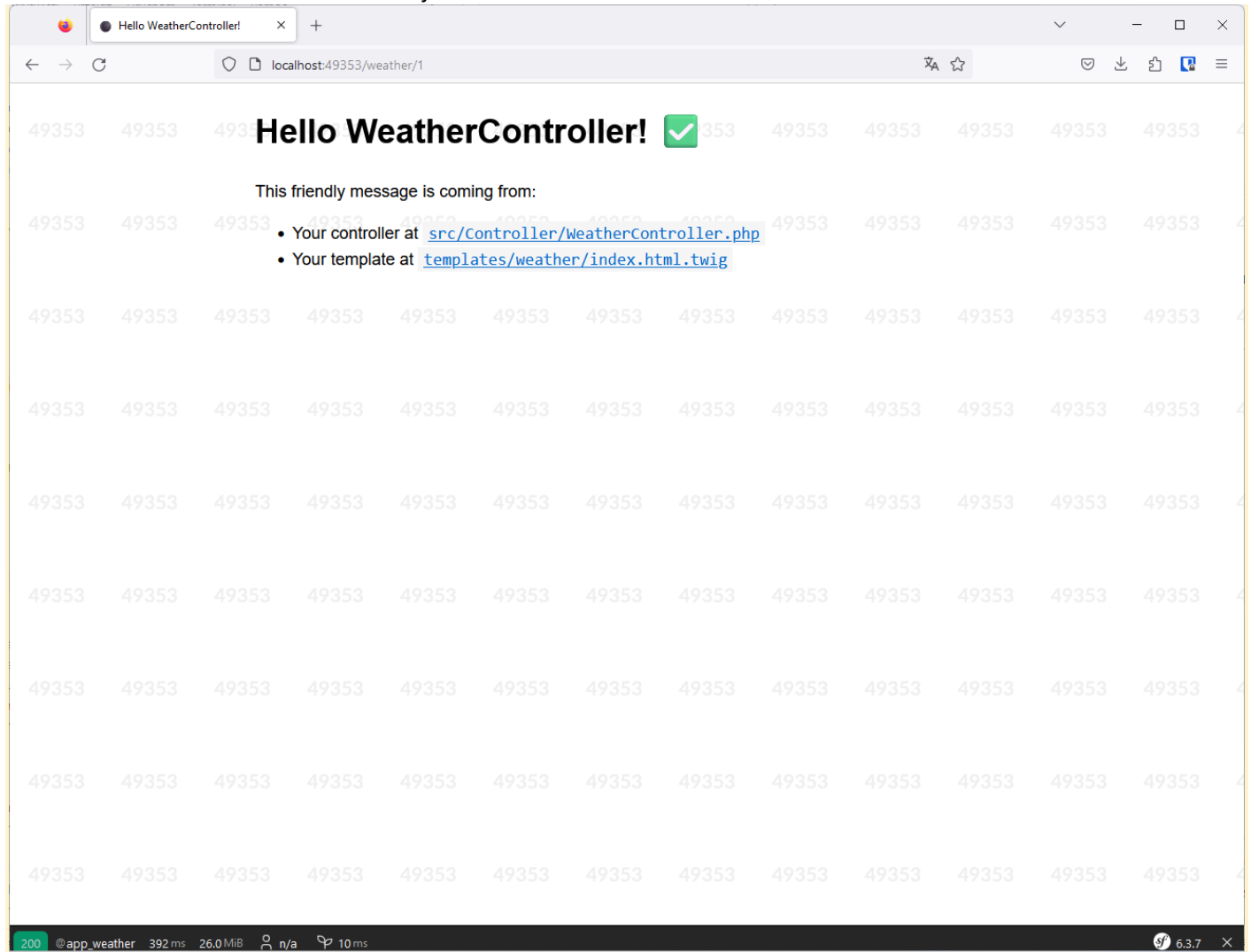
```
body {  
  background: url("https://placeholder.co/100x100/FFFFFF/EFEFEF/png?text= ");  
}  
</style>
```

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:

```

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\Annotation\Route;
8  class WeatherController extends AbstractController
9  {
10     #[Route('/weather/{id}', name: 'app_weather', requirements: ['id' => '\d+'])]
11     public function city($id): Response
12     {
13         return $this->render('weather/index.html.twig', [
14             'controller_name' => 'WeatherController',
15         ]);
16     }
17 }

```

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:

AI2 LAB C – Nowakowski Michał – Wersja 1

```
#[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 %}

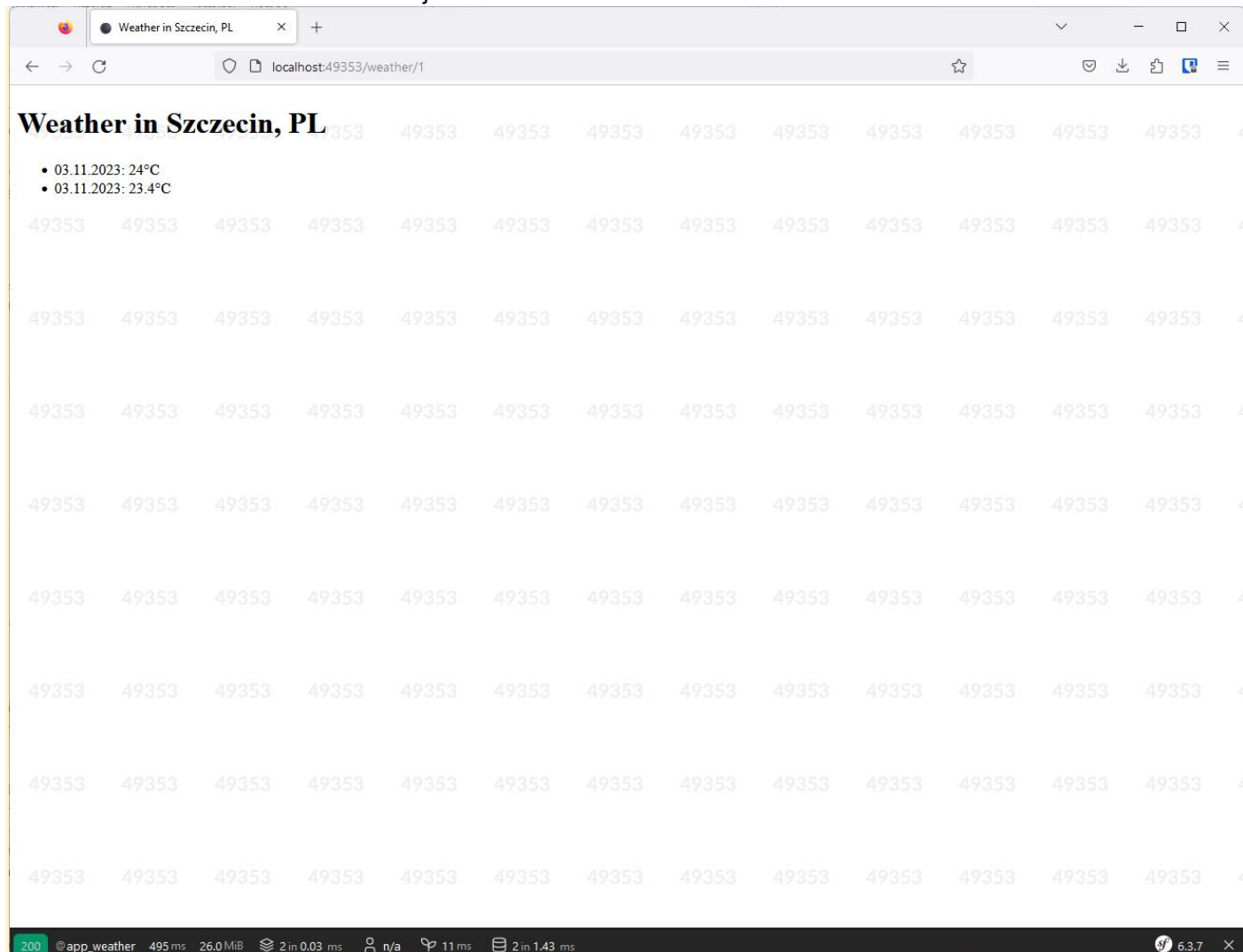
{% block body %}
<main>
    <h1>Weather in {{ location.city }}, {{ location.country }}</h1>

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

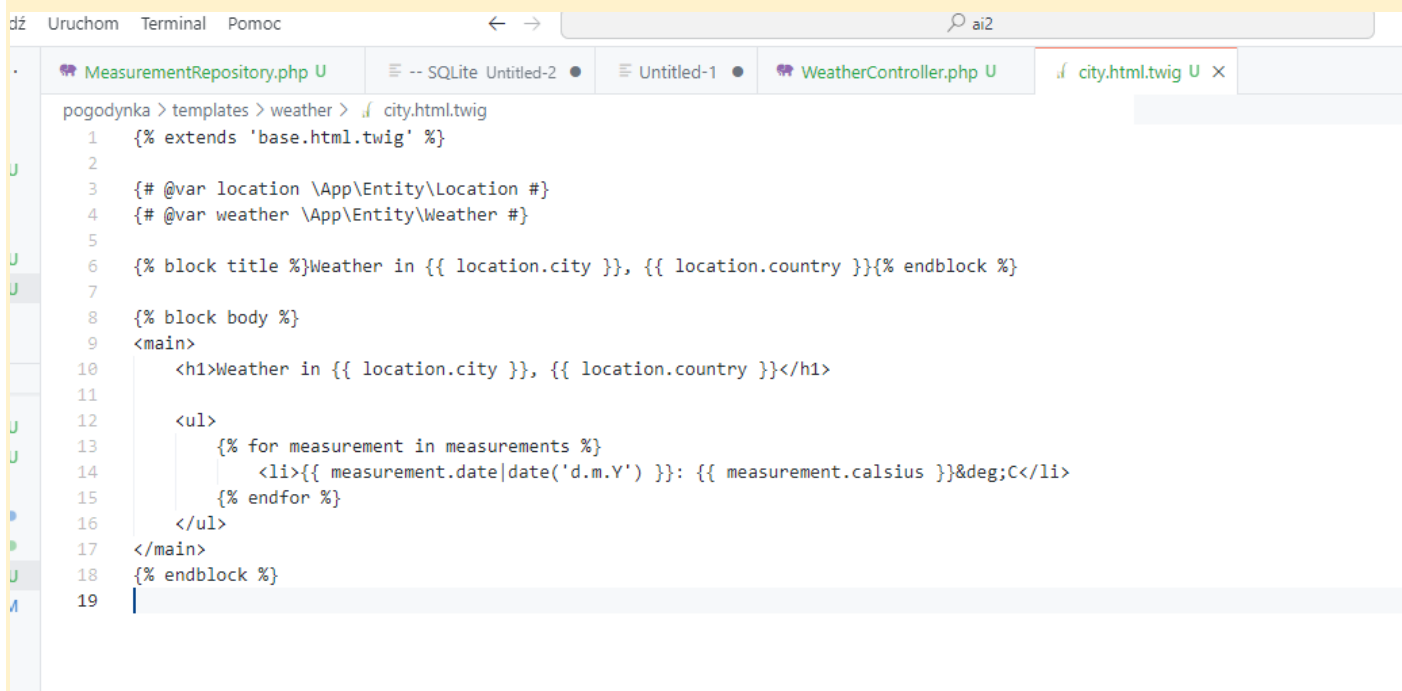
Oczekiwany efekt:



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



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



Punkty:

0

1

Wstaw zrzut ekranu kodu kontrolera WeatherController:

pogodynka > src > Controller > WeatherController.php

```
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\Annotation\Route;
8  use App\Entity\Location;
9  use App\Repository\MeasurementRepository;
10 use App\Entity\Measurement;
11 class WeatherController extends AbstractController
12 {
13     #[Route('/weather/{id}', name: 'app_weather', requirements: ['id' => '\d+'])]
14     public function city(Location $location, MeasurementRepository $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:

```

pogodynka > src > Repository > MeasurementRepository.php
1  <?php
2
3  namespace App\Repository;
4
5  use App\Entity\Measurement;
6  use Doctrine\Bundle\DoctrineBundle\Repository\ServiceEntityRepository;
7  use Doctrine\Persistence\ManagerRegistry;
8  use App\Entity\Location;
9
10 /**
11  * @extends ServiceEntityRepository<Measurement>
12  *
13  * @method Measurement|null find($id, $lockMode = null, $lockVersion = null)
14  * @method Measurement|null findOneBy(array $criteria, array $orderBy = null)
15  * @method Measurement[]    findAll()
16  * @method Measurement[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
17  */
18 class MeasurementRepository extends ServiceEntityRepository
19 {
20     public function __construct(ManagerRegistry $registry)
21     {
22         parent::__construct($registry, Measurement::class);
23     }
24     public function findByLocation(Location $location)
25     {
26         $qb = $this->createQueryBuilder('m');
27         $qb->where('m.location = :location')
28             ->setParameter('location', $location)
29             ->andWhere('m.date > :now')
30             ->setParameter('now', date('Y-m-d'));
31         $query = $qb->getQuery();
32         $result = $query->getResult();
33         return $result;
34     }
35 }

```

DANE WYJŚCIOWE KONSOLA DEBUGOWANIA TERMINAL PORTY GITLENS PROBLEMY

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.

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

Wstaw zrzut ekranu kodu zmodyfikowanego 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\Annotation\Route;
8  use App\Entity\Location;
9  use App\Repository\MeasurementRepository;
10 use App\Entity\Measurement;
11 class WeatherController extends AbstractController
12 {
13     #[Route('/weather/{city}', name: 'app_weather')]
14     public function city(Location $location, MeasurementRepository $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 zrzuty ekranu kodu zmodyfikowanych repozytoriów:

KONTROLA ŹRÓDŁA

LabC Modified

✓ Zatwierdź

✓ Przygotowane zmiany 7

- WeatherController.php pogody... A
- Location.php pogodynka\src\Ent... A
- Measurement.php pogodynka\s... A
- LocationRepository.php pogod... A
- MeasurementRepository.php p... A
- base.html.twig pogodynka\tem... M
- city.html.twig pogodynka\templ... A

✓ Zmiany 0

The image shows a code editor with two files open: `WeatherController.php` and `Location.php`. The `WeatherController.php` file contains the following code:

```
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\Annotation\Route;
8+ use App\Entity\Location;
9+ use App\Repository\MeasurementRepository;
10+ use App\Entity\Measurement;
11+ class WeatherController extends AbstractController
12+ {
13+     #[Route('/weather/{city}', name: 'app_weather')]
14+     public function city(Location $location, MeasurementRepository $repository)
15+     {
16+         $measurements = $repository->findByLocation($location);
17+
18+         return $this->render('weather/city.html.twig', [
19+             'location' => $location,
20+             'measurements' => $measurements,
21+         ]);
22+     }
23+ }
```

The `Location.php` file contains the following code:

```
1+ <?php
2+
3+ namespace App\Entity;
4+
5+ use App\Repository\LocationRepository;
6+ use Doctrine\Common\Collections\ArrayCollection;
7+ use Doctrine\Common\Collections\Collection;
8+ use Doctrine\DBAL\Types\Types;
9+ use Doctrine\ORM\Mapping as ORM;
10+
11+ #[ORM\Entity(repositoryClass: LocationRepository::class)]
12+ class Location
13+ {
14+     #[ORM\Id]
15+     #[ORM\GeneratedValue]
16+     #[ORM\Column]
17+     private ?int $id = null;
18+
19+     #[ORM\Column(length: 255)]
20+     private ?string $city = null;
21+
22+     #[ORM\Column(length: 2)]
23+     private ?string $country = null;
24+
25+     #[ORM\Column(type: Types::DECIMAL, precision: 10, scale: 7)]
26+     private ?string $latitude = null;
27+
28+     #[ORM\Column(type: Types::DECIMAL, precision: 10, scale: 7)]
29+     private ?string $longitude = null;
30+
31+     #[ORM\OneToMany(mappedBy: 'location', targetEntity: Measurement::class)]
32+     private Collection $measurements;
33+
34+     public function __construct()
35+     {
36+         $this->measurements = new ArrayCollection();
37+     }
38+
39+     public function getId(): ?int
40+     {
41+         return $this->id;
42+     }
43+
44+     public function getCity(): ?string
45+     {
46+         return $this->city;
47+     }
48+
49+     public function setCity(string $city): static
```

The Git Graph sidebar on the right shows the commit history for the repository. The main branch is `main`, and the current commit is `7f2291e4`. The sidebar also shows the file structure of the repository, including the `pogodynka` directory and the `src` directory.

Git Graph
Measurement.php (Added in 7f2291e4)
1-

```

1+ <?php
2+
3+ namespace App\Entity;
4+
5+ use App\Repository\MeasurementRepository;
6+ use Doctrine\DBAL\Types\Types;
7+ use Doctrine\ORM\Mapping as ORM;
8+
9+ #[ORM\Entity(repositoryClass: MeasurementRepository::class)]
10+ class Measurement
11+ {
12+     #[ORM\Id]
13+     #[ORM\GeneratedValue]
14+     #[ORM\Column]
15+     private ?int $id = null;
16+
17+     #[ORM\ManyToOne(inversedBy: 'measurements')]
18+     #[ORM\JoinColumn(nullable: false)]
19+     private ?Location $location = null;
20+
21+     #[ORM\Column(type: Types::DATE_MUTABLE)]
22+     private ?\DateTimeInterface $date = null;
23+
24+     #[ORM\Column(type: Types::DECIMAL, precision: 3, scale: '0')]
25+     private ?string $celsius = null;
26+
27+     public function getId(): ?int
28+     {
29+         return $this->id;
30+     }
31+
32+     public function getLocation(): ?Location
33+     {
34+         return $this->location;
35+     }
36+
37+     public function setLocation(?Location $location): static
38+     {
39+         $this->location = $location;
40+
41+         return $this;
42+     }
43+
44+     public function getDate(): ?\DateTimeInterface
45+     {
46+         return $this->date;
47+     }
48+
49+     public function setDate(\DateTimeInterface $date): static

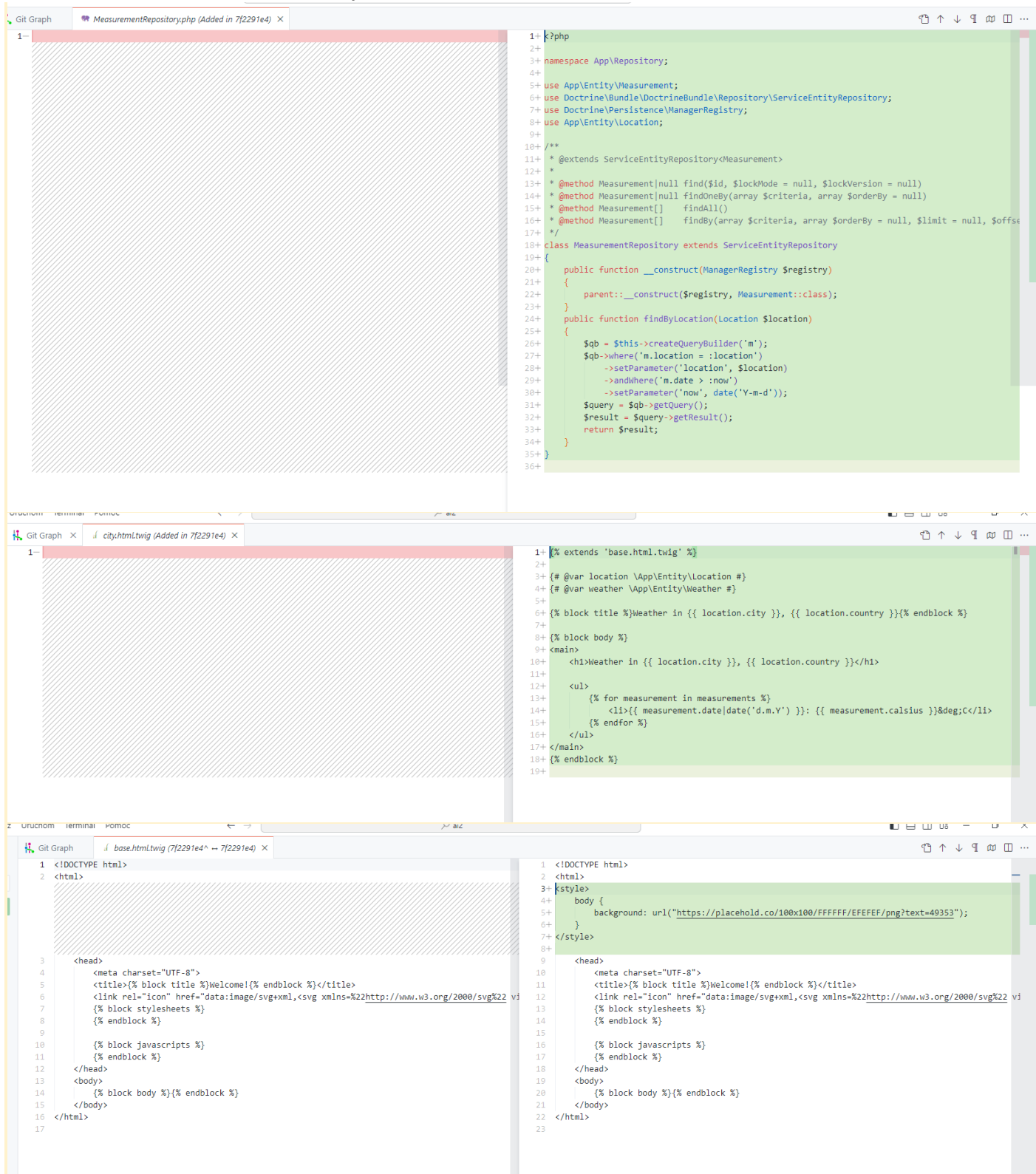
```

Git Graph
LocationRepository.php (Added in 7f2291e4)
1-

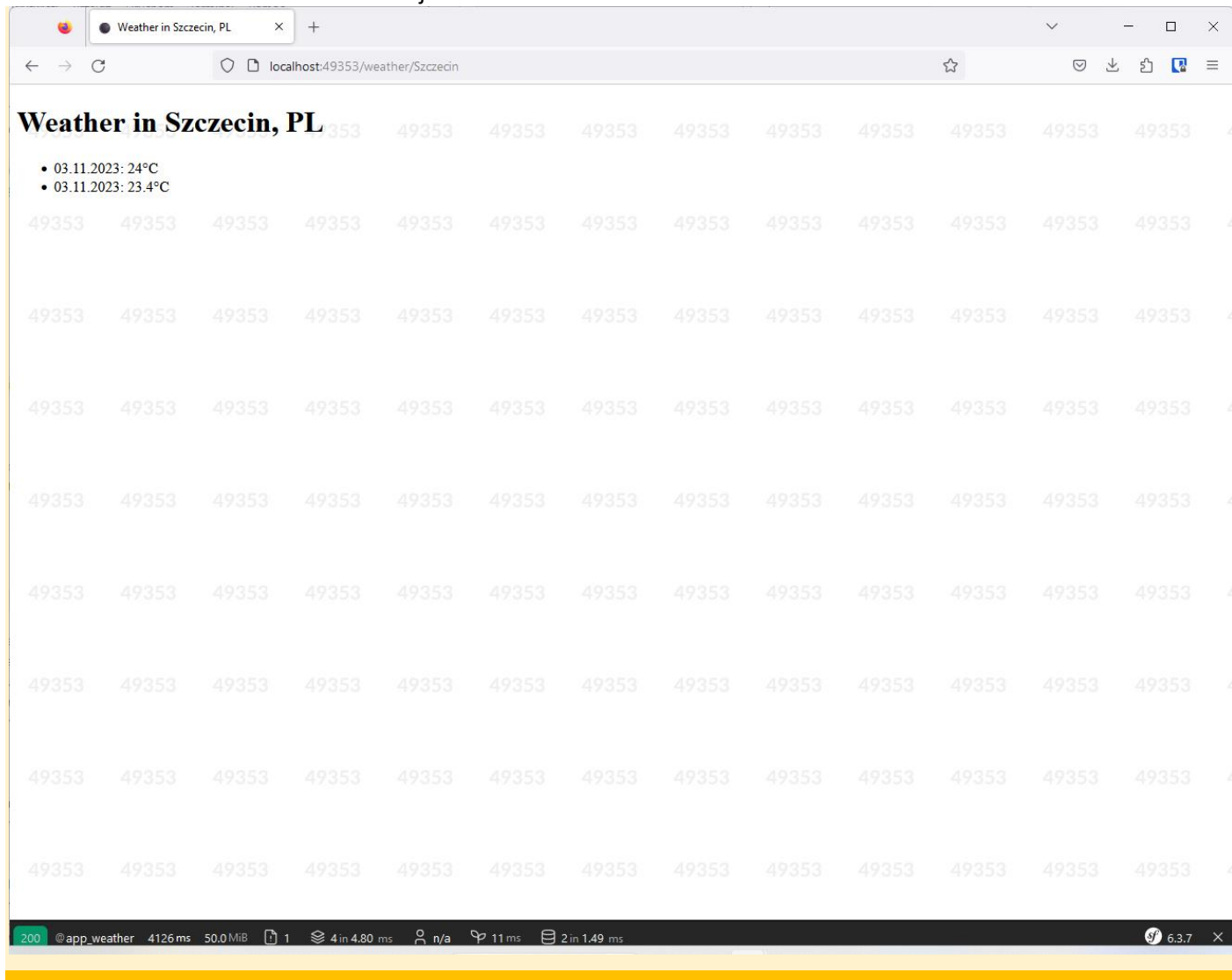
```

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+  * @method Location|null find($id, $lockMode = null, $lockVersion = null)
13+  * @method Location|null findOneBy(array $criteria, array $orderBy = null)
14+  * @method Location[]    findAll()
15+  * @method Location[]    findBy(array $criteria, array $orderBy = null, $limit = null, $offset = null)
16+  */
17+ class LocationRepository extends ServiceEntityRepository
18+ {
19+     public function __construct(ManagerRegistry $registry)
20+     {
21+         parent::__construct($registry, Location::class);
22+     }
23+
24+     /**
25+      * @return Location[] Returns an array of Location objects
26+      */
27+     public function findByExampleField($value): array
28+     {
29+         return $this->createQueryBuilder('l')
30+             ->andWhere('l.exampleField = :val')
31+             ->setParameter('val', $value)
32+             ->orderBy('l.id', 'ASC')
33+             ->setMaxResults(10)
34+             ->getQuery()
35+             ->getResult();
36+     }
37+
38+     public function findOneBySomeField($value): ?Location
39+     {
40+         return $this->createQueryBuilder('l')
41+             ->andWhere('l.exampleField = :val')
42+             ->setParameter('val', $value)
43+             ->getQuery()
44+             ->getOneOrNullResult();
45+     }
46+
47+ }
48+
49+

```



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/Michaldariusznnowakowski/ai2/tree/lab-c>

...link, np. <https://github.com/ideaspot-pl/ai2-pogodynka-202310/tree/lab-c...>

PODSUMOWANIE

W kilku zdaniach podsumuj zdobyte podczas tego laboratorium umiejętności.

W trakcie tego laboratorium zdobyłem umiejętności związane z Symfony Framework, w tym:

- Konfiguracją tras w Symfony za pomocą adnotacji i określeniem wymagań dotyczących parametrów tras.
- Rozumieniem różnicy między parametrami wymaganymi, a opcjonalnymi w kontrolerach.
- Wykorzystaniem Dependency Injection do wstrzykiwania zależności w kontrolerze.
- Przetwarzaniem parametrów przekazywanych w adresie URL, zarówno dla stringów, jak i liczb.

Te umiejętności pozwalają na skuteczne tworzenie aplikacji webowych w Symfony, zarządzanie trasami, przetwarzanie parametrów i komunikację asynchroniczną w ramach aplikacji.

Zweryfikuj kompletność sprawozdania. Utwórz PDF i wyślij w terminie.