

Patryk Jugowiec

Frameworki PHP

**Informatyka
Niestacjonarne
semestr 6**

BACKEND

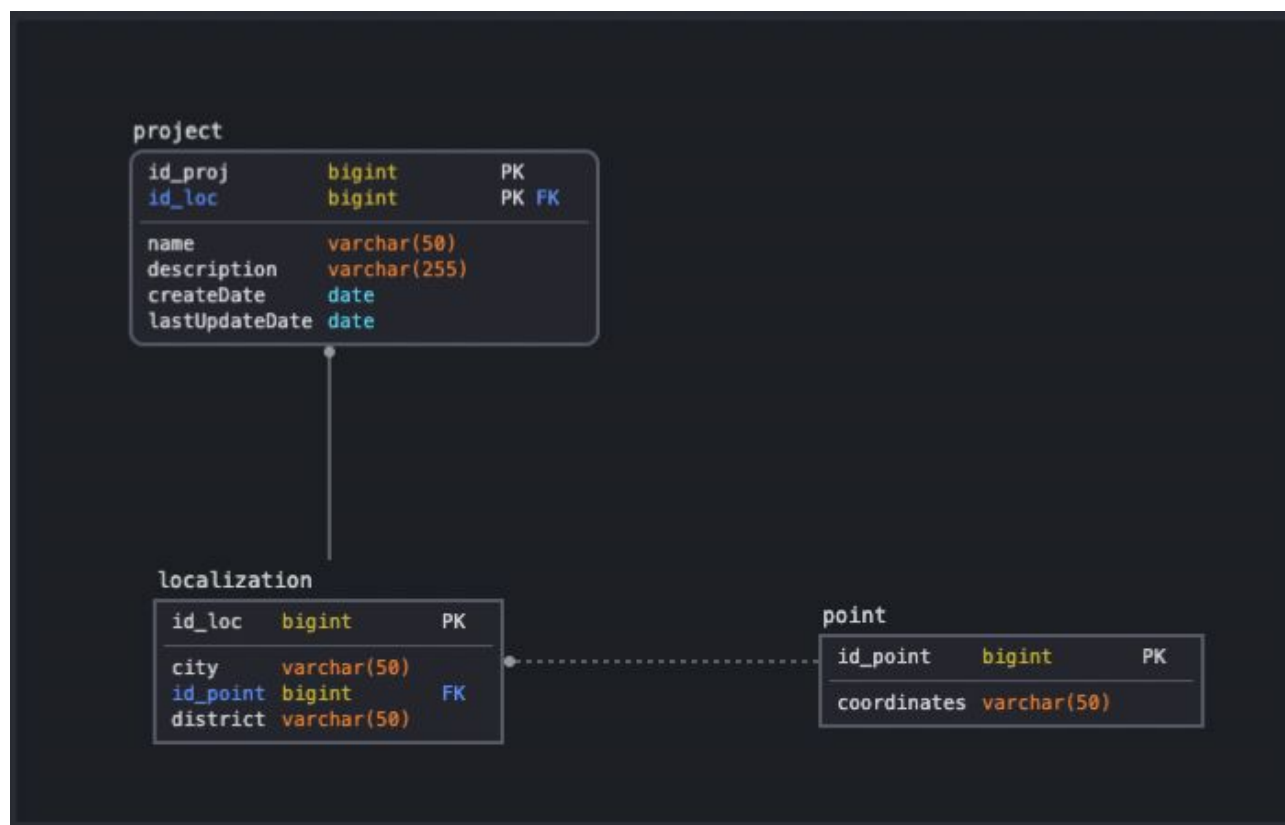
Backend (serwer aplikacji) został zaimplementowany w języku PHP przy pomocy frameworku Symfony w wersji 5. Baza została wygenerowana na podstawie modeli (ORM) w w/w frameworku.

PHP – interpretowany, skryptowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji webowych w czasie rzeczywistym.

Symfony – framework dla aplikacji internetowych napisany w języku PHP bazujący na wzorcu projektowym MVC.

ORM - W informatyce jest techniką programowania służącą do konwersji danych między niekompatybilnymi systemami typów przy użyciu zorientowanych obiektowo języków programowania. W efekcie tworzy się „wirtualna baza danych obiektów”, z której można korzystać z poziomu języka programowania. Dostępne są zarówno darmowe, jak i komercyjne pakiety, które wykonują mapowanie obiektowo-relacyjne, chociaż niektórzy programiści decydują się na tworzenie własnych narzędzi ORM.

Model bazy danych przygotowany w Data Modelerze:



ENCJE:

Encja Projektu na podstawie której framework Symfony wygeneruje odpowiednią tabelę. Jak widać na poniższym screenie został wykorzystany **ORM** w którym definiujemy tabelę. Widać także połączenie relacyjne do encji "Lokalizacja". ManyToOne ponieważ jedna lokalizacja może mieć wiele projektów.

```
/**
 * @ORM\Entity(repositoryClass=ProjectRepository::class)
 */
class Project
{
    /**
     * @var int|null
     *
     * @ORM\Id
     * @ORM\GeneratedValue
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     *
     * @ORM\Column(type="string", length=255)
     */
    private $name;

    /**
     *
     * @ORM\Column(type="string", length=255)
     */
    private $description;

    /**
     *
     * @ORM\ManyToOne(targetEntity="App\Entity\Localization", inversedBy="projects")
     * @ORM\JoinColumn(name="localization_id", nullable=false, referencedColumnName="id")
     */
    private $localization;
```

Po stronie serwera zostały konkretne kontrolery który udostępniają połączenie poprzez REST API.

```
routes.yml
1  • getAllProjects:
2    path: /api/project
3    controller: App\Controller\ProjectController::getAllProjects
4    methods: [GET]
5
6  • createProject:
7    path: /api/project
8    controller: App\Controller\ProjectController::createProject
9    methods: [POST]
10
11 • updateProject:
12   path: /api/project/{id}
13   controller: App\Controller\ProjectController::updateProject
14   methods: [ PUT ]
15
16 • deleteProject:
17   path: /api/project/{id}
18   controller: App\Controller\ProjectController::deleteProject
19   methods: [ DELETE ]
20
21 • getAllLocalizations:
22   path: /api/localization
23   controller: App\Controller\LocalizationController::getAllLocalizations
24   methods: [GET]
25
26 • createLocalization:
27   path: /api/localization
28   controller: App\Controller\LocalizationController::createLocalization
29   methods: [POST]
30
31 • deleteLocalization:
32   path: /api/localization/{id}
33   controller: App\Controller\LocalizationController::deleteLocalization
34   methods: [ DELETE ]
35
36 • updateLocalization:
37   path: /api/localization/{id}
38   controller: App\Controller\LocalizationController::updateLocalization
39   methods: [ PUT ]
```

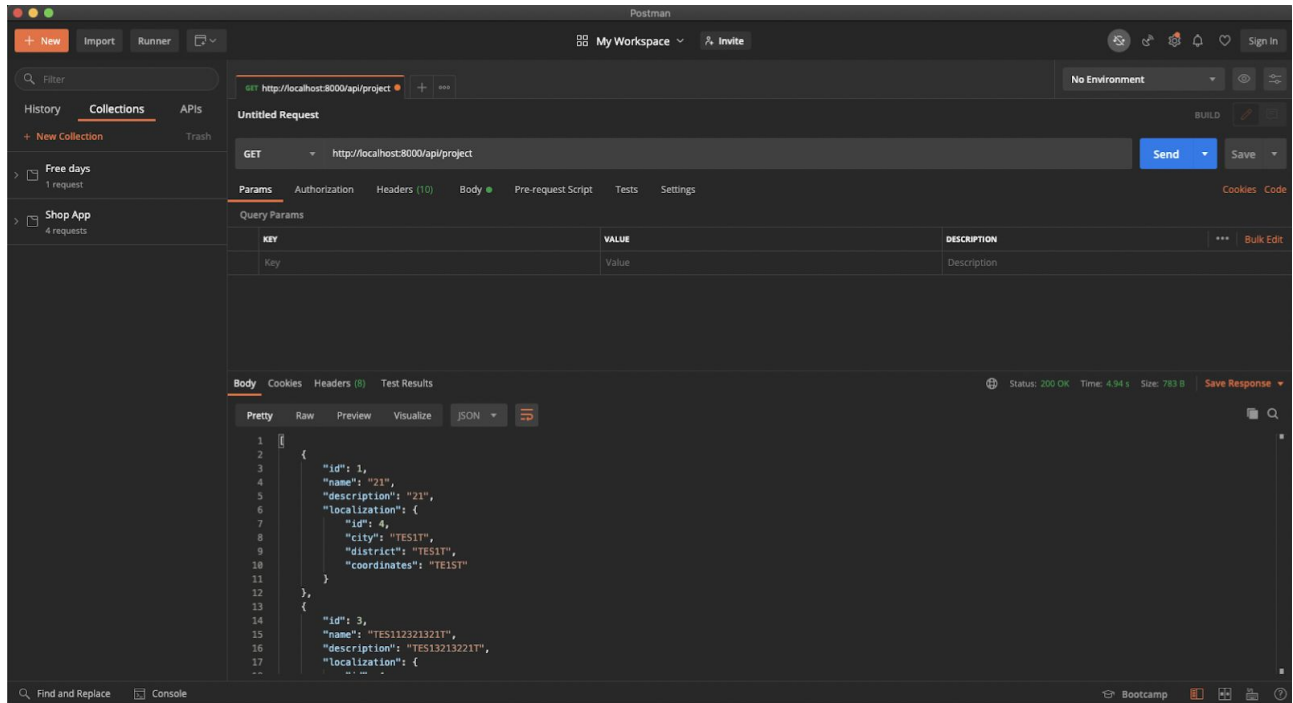
REPOZYTORIUM :

W repozytorium (część logiki odpowiedzialnej za komunikację z bazą danych), zostały dodane mappery które są odpowiedzialne za dostarczenie do kontrolerów gotowego modelu danych.

```
23 public function transform(Project $project): array
24 {
25     return [
26         'id' => (int) $project->getId(),
27         'name' => (string) $project->getName(),
28         'description' => (string) $project->getDescription(),
29         'localization' => $this->transformLocalization($project->getLocalization())
30     ];
31 }
32
33
34 public function transformAll(): array
35 {
36     $projects = $this->findAll();
37     $returnArray = [];
38
39     foreach ($projects as $project) {
40         $returnArray[] = $this->transform($project);
41     }
42
43     return $returnArray;
44 }
45
46 public function transformLocalization(Localization $localization): array
47 {
48     return [
49         'id' => (int) $localization->getId(),
50         'city' => (string) $localization->getCity(),
51         'district' => (string) $localization->getDistrict(),
52         'coordinates' => (string) $localization->getCoordinates()
53     ];
54 }
55 }
```

Przykłady wykorzystania komunikacji REST w praktyce:

/api/project GET



/api/project POST

