

**Patryk Jugowiec**

**Frameworki PHP**

**Informatyka  
Niestacjonarne  
semestr 6**

# BACKEND

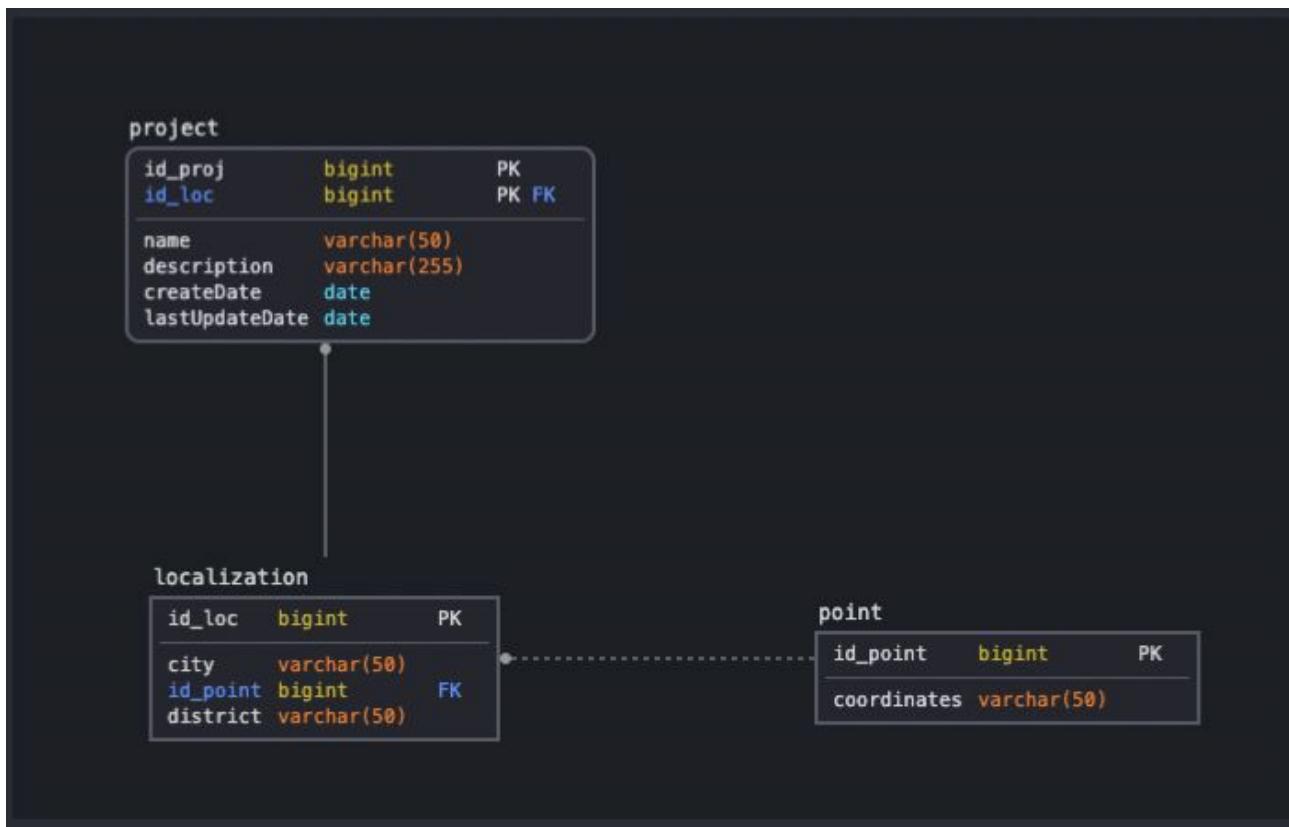
**Backend** ( serwer aplikacji) został zaimplementowany w języku PHP przy pomocy frameworku Symfony w wersji 5. Baza została wygenerowana na podstawie modeli (ORM) w w/w frameworku.

**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:

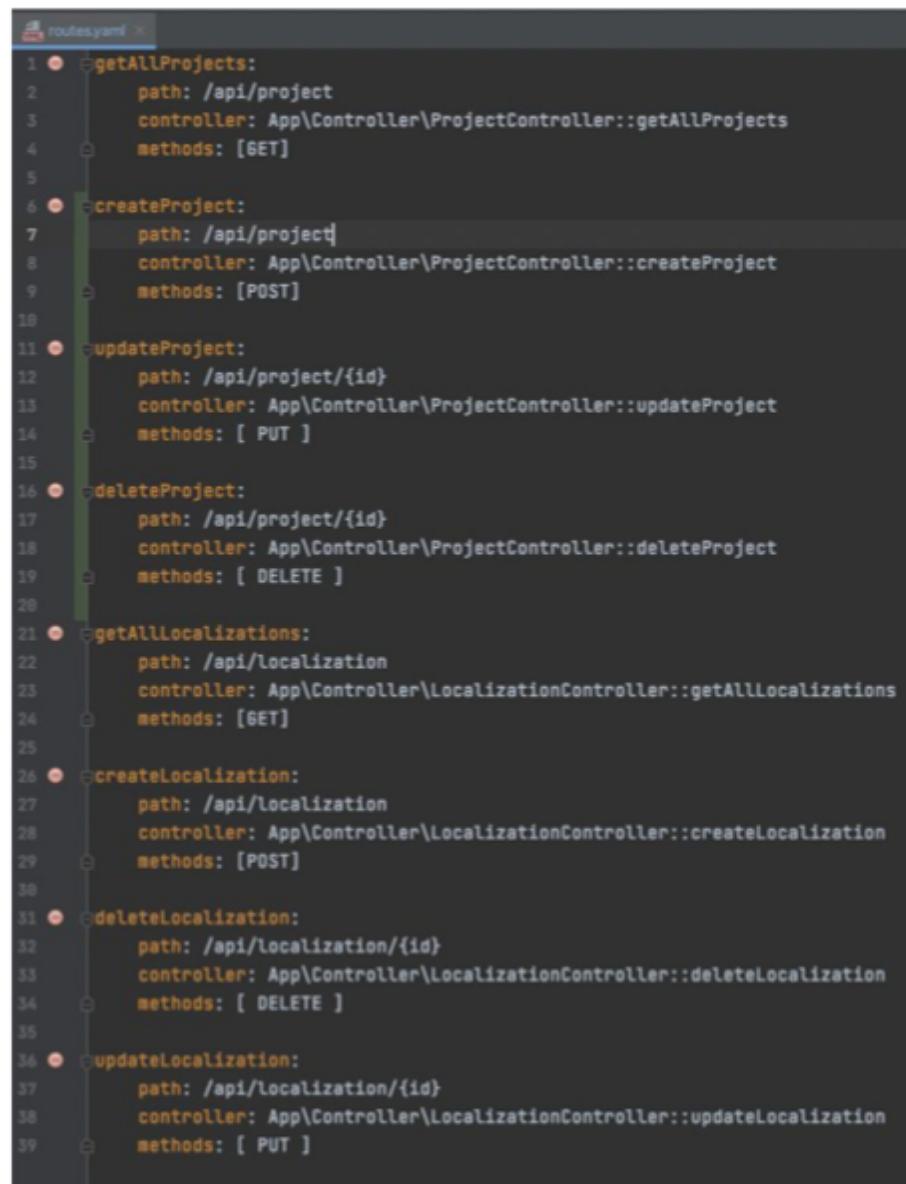


## ENCJE:

Encja Projektu na podstawie której framework Symfony wygeneruje odpowiednią tabelę. Jak widać na poniższym screenie został wykorzystany **ORM** w którym definiujemy tabele. 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.yaml
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 komunikacje 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

The screenshot shows the Postman application interface. A collection named "Free days" is selected on the left. In the main area, a new request is being prepared for "http://localhost:8000/api/project". The method is set to GET. Under the Params tab, there is a single parameter named "Key" with the value "Value". The Body tab shows a JSON response with two project objects. The first object has an id of 1, name "21", description "21", localization with id 4, city "TES1T", district "TES1T", and coordinates "TE1ST". The second object has an id of 3, name "TES112321321T", description "TES13213221T", and localization with id 5. The status bar at the bottom indicates a 200 OK response.

```
[{"id": 1, "name": "21", "description": "21", "localization": {"id": 4, "city": "TES1T", "district": "TES1T", "coordinates": "TE1ST"}}, {"id": 3, "name": "TES112321321T", "description": "TES13213221T", "localization": {"id": 5}}]
```

## /api/project POST

The screenshot shows the Postman application interface. A collection named "Shop App" is selected on the left. In the main area, a new request is being prepared for "http://localhost:8000/api/project". The method is set to POST. Under the Body tab, the content type is set to JSON, and the body contains a single project object with name "21", description "21", and localization\_id "5". The status bar at the bottom indicates a 201 Created response.

```
{"name": "21", "description": "21", "localization_id": 5}
```