AKADEMIA
im. JAKUBA Z PARADYŻA
w GORZOWIE WIELKOPOLSKIM

Dokumentacja

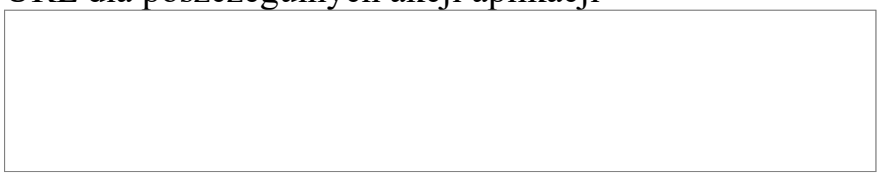
Przedmiot:	**Usługi Webow**
Temat zajęć:	**Projekt Api**
Data:	18 styczeń 2018 roku
Imię i nazwisko:	Maciej Radzio
Rok, grupa:	3 rok, informatyka, studia stacjonarne

Cel Ćwiczenia:

Celem ćwiczenia było wykonanie usługi webowej która wykonuje podstawowe fuknicje CRUD czyli utworzenie, wyświetlenie, edycja oraz usuwanie danych za pomocą zapytać HTTP i modelu RESTful.
Do wykonania został użyty język PHP z frameworkiem Laravel. Do przechowywania danych został użyty system zarządzania bazami danych SQLite

Routing

Aplikacja wykokżystuje element Resources znajdującego się w frameworku Laravel. Odpowiedzialny jest za automatyczne przydzielenie reguł routingu i buduje adresy URL dla poszczegulnych akcji aplikacji



Po wpisaniu php artisan route:list zostaną wypisane reguł routingu aplikacji

```
$ php artisan route:list
+--------+----------+-------------------------+----------------+--------------
------------------------------+--------------+
| Domain | Method   | URI                     | Name           | Action
                              | Middleware   |
+--------+----------+-------------------------+----------------+--------------
------------------------------+--------------+
|        | GET|HEAD | /                       |                | App\Http\Contr
ollers\MovieController@index  | web          |
|        | GET|HEAD | api/movies              | movies.index   | App\Http\Contr
ollers\MovieController@index  | api          |
|        | POST     | api/movies              | movies.store   | App\Http\Contr
ollers\MovieController@store  | api          |
|        | GET|HEAD | api/movies/create       | movies.create  | App\Http\Contr
ollers\MovieController@create | api          |
|        | GET|HEAD | api/movies/{movie}      | movies.show    | App\Http\Contr
ollers\MovieController@show   | api          |
|        | PUT|PATCH| api/movies/{movie}      | movies.update  | App\Http\Contr
ollers\MovieController@update | api          |
|        | DELETE   | api/movies/{movie}      | movies.destroy | App\Http\Contr
ollers\MovieController@destroy | api         |
|        | GET|HEAD | api/movies/{movie}/edit | movies.edit    | App\Http\Contr
ollers\MovieController@edit   | api          |
|        | GET|HEAD | api/user                |                | Closure
                              | api,auth:api |
|        | GET|HEAD | movies                  | movies.index   | App\Http\Contr
ollers\MovieController@index  | web          |
|        | POST     | movies                  | movies.store   | App\Http\Contr
ollers\MovieController@store  | web          |
|        | GET|HEAD | movies/create           | movies.create  | App\Http\Contr
ollers\MovieController@create | web          |
|        | GET|HEAD | movies/{movie}          | movies.show    | App\Http\Contr
ollers\MovieController@show   | web          |
|        | PUT|PATCH| movies/{movie}          | movies.update  | App\Http\Contr
ollers\MovieController@update | web          |
|        | DELETE   | movies/{movie}          | movies.destroy | App\Http\Contr
ollers\MovieController@destroy | web         |
|        | GET|HEAD | movies/{movie}/edit     | movies.edit    | App\Http\Contr
ollers\MovieController@edit   | web          |
+--------+----------+-------------------------+----------------+--------------
------------------------------+--------------+
```

W aplikacji znajdziemy 2 pliki odpowiedzialne za routing web.php oraz api.php odpowiedzialne odpowiednio za reguły routingu dla interfejsu webowego oraz API aplikacji
Zapytania wykonane z przegladarki zostaną zrealizowane poprzez reguł routingu zawarte w pliku web.php, zapytania wykonane z prograów do zapytań JSONowych np. POSTman w momencie gdy ustawimy nagłówek: Accept application/json i dodamy do adresu /api zaraz po adresie zostaną wykonane poprzez reguły routingu zawarte w api.php

Przykładowe reguły routigu
- web – 127.0.0.1:8000/movies/28
- api – 127.0.0.1:8000/api/movies/28

obie służą do wyświetlenia filmu o ID 28

Kontroler, Metody i Dodatkowe Funkcje

W aplikacji znajdziemy Kontroler MoviesConttroler.php w którym zostaly wykorzystane metody takie jak :

```php
public function index(Request $request)
{
    $movies = MovieResource::collection(Movie::paginate(20));
    if ($request->wantsJson()) {
        return $movies;
    } else {
        return view( view: 'movies.index', ['movies' => $movies]);
    }
}
```

Metoda ta jest odpowiedzialna za listowanie elementów z klasy Movies.
Dodatkowo metoda paginate jest odpowiedzialna za Paginacje elemenetów po 20 na strone (zależnie od wpisanej wartości).
MoviesResource:collection jest odpowiedzialne za utworzenie konwertera z obiektu na postać JSON oraz przetworzeniem tablicy z obiektami klasy Resoucre

```php
public function create()
{
    return view( view: 'movies.create', [
        'movie' => new Movie()
    ]);
}
```

Funkcja creata jest odpowiedzialna za przekierowanie do formularza dodawania filmu do bazy

```php
public function store(Request $request)
{
    $this->validate($request, [
        'title' => 'required',
        'director' => 'required',
        'year' => 'required',
        'genre' => 'required'
    ]);

    $movie = new Movie([
        'title' => $request->input( key: 'title'),
        'director' => $request->input( key: 'director'),
        'year' => $request->input( key: 'year'),
        'duration_in_minutes' => $request->input( key: 'duration_in_minutes'),
        'genre' => $request->input( key: 'genre'),
        'rating' => $request->input( key: 'rating'),
        'country' => $request->input( key: 'country'),
    ]);

    $movie->saveOrFail();

    $message = 'Dodano film do bazy.';

    if ($request->wantsJson()) {
        $response = [
            'message' => $message,
            'movie' => $movie
        ];
        return response()->json($response, status: 200);
    } else {
        flash($message)->success();
        return view( view: 'movies.show', [
            'movie' => $movie
        ]);
    }
}
```

Funkcja store jest odpowiedzialna za dodanie filmu do bazy poprzez wypełnienie formularza dodawania gdzie wymagane są pola Tytuł, Reżyser, Rok produkcji i Gatunek

```php
public function show(Request $request, $id)
{
    $movie = Movie::find($id);
    if ($request->wantsJson()) {
        $response = [
            'movie' => $movie
        ];
        return response()->json($response, status: 200);
    } else {
        return view( view: 'movies.show', [
            'movie' => $movie
        ]);
    }
}
```

Funkcja show jest odpowiedzialna za wyświetlenie filmu

```php
public function edit($id)
{
    return view( view: 'movies.edit', [
        'movie' => Movie::find($id)
    ]);
}
```

Funkcja edit jest odpowiedzialna za przekierowanie do formularza edycji filmu

```php
public function update(Request $request, $id)
{
    $this->validate($request, [
        'title' => 'required',
        'director' => 'required',
        'year' => 'required',
        'genre' => 'required'
    ]);

    $movie = Movie::findOrFail($id);

    $movie->title = $request->input( key: 'title');
    $movie->director = $request->input( key: 'director');
    $movie->year = $request->input( key: 'year');
    $movie->duration_in_minutes = $request->input( key: 'duration_in_minutes');
    $movie->genre = $request->input( key: 'genre');
    $movie->rating = $request->input( key: 'rating');
    $movie->country = $request->input( key: 'country');

    $movie->saveOrFail();

    $message = 'Zaktualizowano film.';

    if ($request->wantsJson()) {
        $response = [
            'message' => $message,
            'movie' => $movie
        ];
        return response()->json($response, status: 200);
    } else {
        flash($message)->success();
        return view( view: 'movies.show', [
            'movie' => $movie
        ]);
    }
}
```

Funkcja update jest odpowiedzialna za edycja filmu z bazy poprzez pobranie aktualnych wartości, wypełnienie formularza edytowania gdzie wymagane są pola Tytuł, Reżyser, Rok produkcji i Gatunek

```php
public function destroy(Request $request, $id)
{
    $movie = Movie::findOrFail($id);
    $movie->delete();

    $message = 'Usunięto film z bazy.';

    if ($request->wantsJson()) {
        $response = [
            'message' => $message,
            'movie' => $movie
        ];
        return response()->json($response, status: 200);
    } else {
        flash($message)->success();
        return redirect()->route( route: 'movies.index');
    }
}
```

Funkcja destroy jest odpowiedzialna za usuwanie filmu z bazy

```php
public function run()
{
    $faker = Faker::create();
    foreach (range( low: 1, high: 1000) as $index) {
        DB::table('movies')->insert([
            'title' => $faker->word,
            'year' => $faker->year,
            'director' => $faker->firstName . " " . $faker->lastName,
            'genre' => $faker->word,
            'country' => $faker->country,
            'duration_in_minutes' => $faker->randomNumber( nbDigits: 3),
            'rating' => $faker->word,
            'created_at'=>new \DateTime(),
            'updated_at'=>new \DateTime(),
        ]);
    }
}
```

Funkcja run jest odpowiedzialna za wypelnienie bazy filmami
Faker wypełnia formularze create losowymi danymi powtarza to 1000 razy
Dla kazdego rekordu jest zdefiniowane jaki typ zmiennej ma być zastosowany