# Super Animal Projekt

Marcin Zawada Paweł Żelazek Jakub Hawro

2 czerwca 2020

## Spis treści

# 1 Opis funkcjonalny projektu

Projekt o nazwie Super Animal ma za zadanie stworzyć platformę dla hodowców zwierząt, w której to hodowcy będą mogli znaleźć partnerkę bądź partnera dla swojego zwierzęcia, aby dokonać reprodukcji. W projekcie tym jest możliwość założenia własnego konta hodowcy, w którym to dodaje się swoje posiadane zwierzęta. Następnie można wyszukać odpowiednie zwierzę za pomocą prostej wyszukiwarki, a następnie skontaktować się z właścicielem danego zwierzęcia.

# 2 Wyszczególnione wdrożone kwalifikacje

## 2.1 HTML5

Szkielet aplikacji oparty jest o HTML5. Jednakże, ze względu na użycie w tym projekcie Asp.NET, pliki zawierające szkielet mają rozszerzenie *.cshtml* . Jest to wymóg, aby można było używać możliwości, jakie oferuje język C# w plikach HTML.

```
<div class="row">
    <div class="col-8">
        <h1>@Model.Pet.Name</h1>
    </div>
    <div class="col-2">
        @Html.ActionLink("Edit pet", "EditPet", "Pet", new { PetId = @Model.Pet.Id },
     new { @class = "text-decoration-none btn btn-info" })
    </div>
    <div class="col-2">
        @Html.ActionLink("Delete pet", "DeletePet", "Pet", new { PetId = @Model.Pet.
    Id }, new { @class = "text-decoration-none btn btn-danger" })
    </div>
</div>

```

Listing 1: Przykład użycia HTML w pliku *.cshtml*

## 2.2 CSS3

W aplikacji używamy frameworku CSS3, nazwanego Bootstrap.

## 2.3 Formularze

W aplikacji jest możliwość dodawania nowych zwierząt do swojej kolekcji, edytowania ich (np. nowe zdjęcie lub nazwa) a także usunięcia ich. Oto przykład formularza edytowania:

```
<form enctype="multipart/form-data" asp-controller="Pet" asp-action="EditPet" method="post">
    <hr />
    <div asp-validation-summary="All" class="text-danger"></div>

    <div class="form-group">
        <label asp-for="Pet.Name"></label>
        <input asp-for="Pet.Name" class="form-control" value="@Model.Pet.Name" />
        <span asp-validation-for="Pet.Name" class="text-danger"></span>
    </div>

    <div class="form-group">
        <label asp-for="Pet.Description"></label>
        <input asp-for="Pet.Description" class="form-control" value="@Model.Pet.Description" />
        <span asp-validation-for="Pet.Description" class="text-danger"></span>
    </div>

    <input asp-for="Pet.Id" type="hidden" value="@Model.Pet.Id" />
    <input asp-for="Pet.UserId" type="hidden" value="@Model.Pet.UserId" />
    <input asp-for="Pet.ProfilePhoto" type="hidden" value="@Model.Pet.ProfilePhoto" />
    <input asp-for="Pet.Genre" type="hidden" value="@Model.Pet.Genre" />

    <div class="form-group">
        <button type="submit" class="btn btn-default">Edit Pet </button>
    </div>
</form>

```

Listing 2: Przykład formularza w języku Asp.NET Core

## 2.4 Baza danych

Baza danych użyta w programie opiera się na MSSQL. Ze względu na specyfikę tej konkretnej bazy, jest ona skonstruowana tak, że jakiekolwiek spojrzenie na nią daje efekt, że jest to baza zewnętrzna (nawet jeśli serwer jest lokalny). Dzięki temu można łatwo stwierdzić, że baza danych jest typowo zewnętrzna.

## 2.5 Router

Inaczej przekierowania. W przypadku asp.NET są one obsługiwane poprzez klasy w folderze *Controllers*.

## 2.6 Uwierzytelnianie

Aby móc obsłużyć nasz program, jest wymagane zalogowanie się. W przypadku nowych osób, obsługiwana jest rejestracja.

## 2.7 MVC

Program jest stworzony według wzorca MVC, gdzie Models reprezentuje model, View widok a Controlles kontroler.

## 2.8 CRUD

Jest możliwość dodania, modyfikacji oraz usunięcia swoich posiadanych zwierząt na platformie. Wszystkie te atrybuty znajdują się w bazie danych.

## 2.9 ORM

W projekcie jest wykorzysany ORM Entity.

```
public ServiceResponse<PetIndexViewModel> GetPetIndexViewModelForPetId(AppUser
    loggedUser, int petId)
{
    var pet = Context.Pets.Include(x => x.User).FirstOrDefault(x => x.Id == petId);

    if (pet == null || pet.User.Id != loggedUser.Id)
        return ServiceResponse<PetIndexViewModel>.Error();
    else
        return ServiceResponse<PetIndexViewModel>.Ok(new PetIndexViewModel { Pet =
    pet });
}

```

Listing 3: Przykład zastosowania zapytania ORM Entity

## 2.10 Wystawienie API

W projekcie korzystamy również z API, które jest wystawione na zewnątrz. Domyślną obsługę znajdziemy pod adresem *../api/PetsApi*. Wszystkie wykorzystane API znajdują się w pliku *PetsApiController.cs*.

## 2.11 Konsumowanie API

W projekcie korzystamy z prostego zewnętrznego API dotyczącego statystyk zachorowań na Koronawirusa.

## 2.12 AJAX

AJAX został użyty podczas konstruowania wyświetlania zewnętrznego API.

## 2.13 Mail

Wykorzystaliśmy obsługę email do wysyłania wiadomości do innych hodowców. (z jakiegoś powodu nie działa i nie potrafimy tego naprawić).

## 2.14 Lokalizacja

Nasz projekt nie umożliwia lokalizacji na inne języki.

## 2.15 RWD

Ze względu na użycie frameworku Bootstrap, projekt jest automatycznie dostosowywany do innych urządzeń.

### 2.16 Logger

Istnieje logowanie akcji, które nie jest zapisywane do pliku, a wyświetlanie jedynie w konsoli. Aby się do niej dostać, w Visual Studio Code należy kliknąć: Debug, następnie Window, następnie Output.

### 2.17 Cache

W projekcie nie wykorzystujemy cache.

### 2.18 System zarządzania zależnościami

W projekcie został wykorzystany system **NuGet**.

### 2.19 Automatyzacja

Nie udało nam się zaimplementować żadnej przydatnej automatyzacji.

### 2.20 SEO

W projekcie wykorzystujemy słowa kluczowe zapisane w nagłówku głównego pliku HTML oraz plik *robots.txt*.

## 3 Streszczenie opisu technologicznego

W naszym projekcie wykorzystaliśmy technologię asp.NET Core 3.1. Jest to zbiór technologii opartych na frameworku zaprojektowanym przez firmę Microsoft. Przeznaczony jest do budowy różnorodnych aplikacji internetowych. Framework ten oparliśmy o C# w wersji 8. Sam projekt napisaliśmy w programie Visual Studio 2019.

W programie tym wykorzystaliśmy domyślny framwerok CSS3 nazwany Bootstrap (Visual Studio domyślnie dodaje ten framework do każdej tworzonej aplikacji internetowej opartej na asp.NET Core). Sam tekst HTML jest zapisany w plikach mających rozszerzenie .cshtml. Jest to spowodowane tym, aby można było wykorzystać możliwości C# typu pętle, zmienne i inne (domyślnie HTML nie obsługuje tego typu instrukcji).

Nasza aplikacja korzysta także z bazy danych MSSQL w wersji SQL Server 2019. Używamy także paczki Entity w najnowszej wersji, aby móc w łatwy sposób obsłużyć zapytania do tej bazy.

## 4 Instrukcja lokalnego i zdalnego uruchomienia systemu

Aby uruchomić program lokalnie, najprościej będzie zainstalować Visual Studio 2019 oraz zainstalować pakiet roboczy *Opracowywanie zawartości dla platformy ASP.NET i sieci web*. Z drugiej strony, będziemy potrzebowali także serwera baz danych. Będziemy musieli zainstalować MSSQL w wersji 2019. Projekt, który znajduje się na githubie wykorzystuje lokalną instalację MSSQL, także jedyne co będziemy musieli edytować to plik *appsettings.json* i ustawić DefaultConnection.

Aby uruchomić program zdalnie, wystarczy wyeksportować za pomocą Visual Studio projekt na platformę Azure. Instrukcje dotyczące obsługi tej platformy znajdują się właśnie w zasobach tej platformy. Istnieje jeszcze możliwość wyeksportowania tej aplikacji jako plik Docker. Taki plik można wtedy użyć także w zdalnych serwerach. Instrukcję wykorzystania tego pliku także można znaleźć łatwo w Internecie i będą na pewno bardziej dokładne niż opis tutaj.

# 5 Wnioski projektowe

Na początku chcieliśmy stworzyć prostą platformę wzorując się na Facebooku. Miałą być to platforma dla hodowców rasowych zwierząt. Chcieliśmy, aby ci hodowcy mogli znajdować odpowiednie pupile do rozmnażania, gdyż o to ciężko jest. Ustaliliśmy, że projekt będzie napisany w asp.NET. Większość pracy, ze względu na biegłą znajomość języka, wykonał Marcin. Na początku Jakub próbował też coś napisać, jednakże rosło napięcie w zespole i Jakub skupił się na innych rzeczach (jak choćby napisanie tego sprawozdania). Paweł z kolei był odpowiedzialny za przetestowanie tej aplikacji (w wyniku testów wyszły problemy z emailem, których nie potrafimy jeszcze rozwiązać). Obawialiśmy się, że może nam się nie udać ukończyć projektu na czas, jednakże dzięki zwięzłej pracy udało nam się ukończyć projekt. W przyszłości być może planujemy rozbudowę tego projektu o dodatkowe funkcje.