

Projekt zaliczeniowy – Instrukcja

Aplikacja została podzielona na 2 części (warstwy). Pierwsza to “damkorki_app”, która obejmuje aplikację SPA utworzoną przy pomocy Angular 5 oraz ASP.NET Core 2. Głównie kod frontendowy. Druga “damkorki_web_api” to aplikacja implementująca web service w ASP.NET Core z wykorzystaniem Entity Framework Core i Code First Migrations do bazy danych MS SQL. Aplikacje zostały deployowane na Microsoft Azure/Hosting odpowiednio pod adresem: <http://wojciechowska-001-site1.itemurl.com> oraz <https://damkorki-web-api.azurewebsites.net>. Aplikacji DamkorkiApp (Angular) nie udało mi się opublikować na Microsoft Azure z powodu jakiegoś błędu dlatego też wrzuciłam ją przez FTP na darmowy hosting.

Cały projekt dostępny jest na github pod adresem:
<https://github.com/iwona-wojciechowska/damkorki>

Omówienie aplikacji **Damkorki Web API**:

W **appsettings.json** znajdują się ConnectionStringi do bazy danych na MacOS (MS SQL uruchamiany na Dockerze), do lokalnej bazy danych uruchamianej na Windowsie w Visual Studio, oraz do bazy danych w Microsoft Azure.

Znajdują się tam również dane do autentykacji użytkownika z wykorzystaniem tokenów JWT (JSON Web Token): SecretKey, jak również dane do uwierzytelnienia mechanizmów logowania w Facebook, Google. Strona posiada autentykację tokenową z danymi użytkownika przechowywanymi w lokalnej bazie danych SQL Server przy użyciu ASP.NET Identity. Dodatkowo zaimplementowana została możliwość logowania z Facebookiem i Googlem, jest ona zintegrowana z ASP.NET Identity.

Aplikacja wykorzystuje Entity Framework Core:

- ! Encje oraz DbContext znajdują się w katalogu Models
- ! Aplikacja korzysta z podejścia Code First Migrations. W pliku Startup.cs odbywa się automatyczne migrowanie modelu do bazy danych przy pomocy:

// Automatically perform database migration (useful when on Azure)

```
services.BuildServiceProvider().GetService<DbContext>().Database.Migrate();
```

- ! Ponad warstwą modelu danych utworzone zostało repozytorium stosując Generic Repository Pattern (/Repositories/Repository.cs), poszczególne repozytoria dziedziczą po jednym abstrakcyjnym repozytorium parametryzowanym typem TEntity oraz implementują własny interfejs dodatkowych metod dostępu do danych. Kolekcje danych z repozytoriów są zwracane jako IEnumerable<TEntity> aby uniemożliwić dodatkowy dostęp do interfejsu IQueryable<TEntity> w wyższych warstwach aplikacji np. w implementacji Controllerów Web Api.

Web API zostało zaimplementowane przy pomocy klas dziedziczących po Controller które znajdują się w katalogu Controllers. Metody Controllerów oznaczone są atrybutami [Route] oraz [HttpGet], [HttpPost], etc. Pobierają i zwracają dane w postaci obiektów ViewModel.

Klasy ViewModel znajdują się w katalogu ViewModels.

Omówienie aplikacji **Damkorki App**

To aplikacja webowa utworzona z wykorzystaniem technologii ASP.NET Core 2.0 oraz Angular 5.0. Aplikacja bazuje na projekcie startowym [aspnetcore-angular2-universal](#).

Główne zaimplementowane funkcjonalności aplikacji:

- ! Cała strona podzielona na komponenty znajdujące się w Client/app/components. Komponent aplikacji składa się z komponentów Headera, Footera, MainContent.
- ! strona logowania w tym logowanie z Facebook i Google
- ! strona rejestracji

- ! resetowanie hasła
- ! wysyłanie linku aktywacyjnego i linku do resetowania hasła (nie działa na Microsoft Azure, problem z SMTPClient)
- ! profil użytkownika z podstawowymi danymi (walidacja danych formularza)
- ! strona z formularzem dodawania adresu użytkownika
- ! strona tworzenia profilu nauczyciela (Tutor Profile), opcje dodawania Doświadczeń zawodowych (Experiences) oraz umiejętności (Skills)
- ! strona dodawania oferty lekcji (Add Lesson Offer, lesson-offers/add)
- ! strona wyświetlania oferty lekcji (linki na dole strony, /lesson-offers)
- ! podstrony edycji profilu w oddzielnym module @NgModule /Client/app/modules/profile
- ! podstrony ofert lekcji w oddzielnym module @NgModule /Client/app/modules/lesson-offers
- ! formularze np. edycji profilu posiadają CanDeactivateGuard, który pyta użytkownika czy chce odrzucić zmiany dokonane w formularzu.
- ! AuthGuard zabezpiecza strony tylko dla zalogowanych użytkowników.
- ! Strona dodawania zdjęcia profilowego, pozwala poprzez Drag&Drop przesłać zdjęcie na serwer web api.
- ! Rozpoczęto implementację map Leaflet, przetestowano ustawianie obszaru mapy i dodawanie Placemarków. Implementacja komponentu nie dokończona.
- ! Komponent wyświetlania przykładowych ofert, który można wykorzystać do wyświetlania siatki bloków z ofertami w dowolnej konfiguracji (liczba kolumn, wierszy), korzystający z Responsive Web Design (CSS).
- ! Komponenty LessonOfferGridCard wyświetlają poszczególne oferty lekcji z bazy danych.

Utworzono również szablon dla podstrony /lesson-offers/{lessonOfferId} plik: damkorki_user.html.

Testowanie Web API przy pomocy Google Chrome POSTMAN:

Testowanie API wymaga wygenerowania Tokena JWT:

// POST: <https://damkorki-web-api.azurewebsites.net/auth/token>

Headers:

Accept: application/json

ContentType: application/x-www-form-urlencoded

Body (x-www-form-urlencoded):

userName: tkobus

password: Pass4user3!

grant_type: password

client_id: DamkorkiApp

client_secret: secret_123456789

Następnie otrzymany access_token

```
{ "access_token":
```

```
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDAxLzA1L2lkZW50aXR5L2NsYWVtcy9uYW1lIjoidGtvYnVzIiwiaWF0Ij0b21hc3oua29idXNAZ21haWwY29tIiwidWlkIjoiMzU3YWJhYjltNWYwNC00Y2RkLWI5MGEtZGJhNzc3NDU3ZTU0Iiwicm9sZXMiOiJVe2VyIiwiaXNzIjoiRGFta29ya2lXZWJBcGkiLCJhdWQiOiJFYW1rb3JraUFwcCI5InN1YiI6InRrb2J1cyIsImp0aSI6IjQzMmVIMmY1LWlzMzYtNDE5YS1iOQQ0LTU0YzIwYTU3NTZlZCIsImhhdCI6MTUxNjU0OTU3MywibmJmIjoxNTE2NTQ5NTczLCJleHAiOiJlMTY1NTAxNzN9.8a4lQdqYFc00Cu9M0XI_8ssI64qoKcr86gluMMHEHxU",
```

```
  "expiration": 600,
```

```
  "refresh_token": "0d38d5fb0c294cd4900a8447bdfa9eb8"
```

```
}
```

używamy w kolejnych zapytaniach wklejając w nagłówek Authorization : Bearer <access_token>.