Contman Task

Robert Walicki

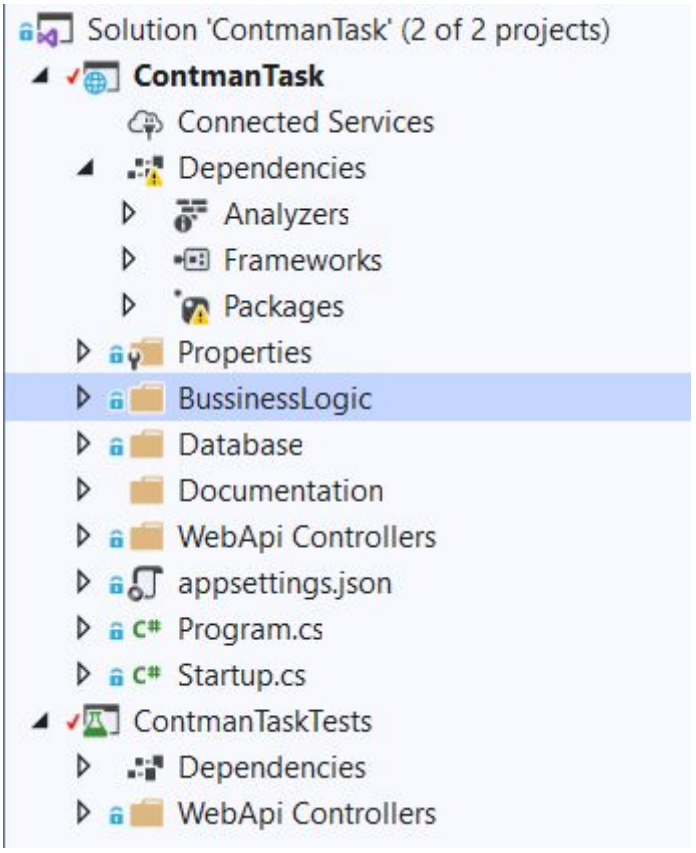
1. **Zadanie**

Projekt po uruchomieniu można odpytywać z adresu **https://localhost:5001/**. Dokumentacja API zgodna ze standardem OAS jest dostępna po uruchomieniu serwisu i znajduje się pod adresem **https://localhost:5001/swagger** lub w wersji offline w katalogu Documentation projektu ContmanTask. Poniższa dokumentacja zawiera dodatkowe informacje dotyczące zadania m.in. architekturę i budowę projektu, stack technologiczny, oraz schemat budowy bazy danych

2. **Architektura**

Zaprojektowane API jest oparte na architekturze RESTful. Do obsługi maili oraz grup mailowych zostały stworzone endpointy zgodnie z konceptem CRUD, a dla kont użytkowników powstały jedynie endpointy z tworzeniem konta oraz logowaniem, aby uniemożliwić podgląd danych innych użytkowników. API jest zabezpieczone tokenem uwierzytelniającym otrzymywanym po zalogowaniu użytkownika, który pozwala na dostęp do funkcjonalności związanych z mailami oraz grupami mailowymi.

3. **Budowa projektu**



Projekt ContmanTask zostały podzielone zgodnie z przeznaczeniem na podkatalogi:

- BussinessLogic – zawiera logikę biznesową używana przez API i nie tylko – tu odbywa się pobieranie danych z bazy oraz ich zapis
- Database - klasy używane do łączenia z bazą danych MySQL
- Documentation - dokumentacja zadania
- WebApi Controllers - klasy stanowiące warstwę sieciową API

Project ContmanTaskTests jest projektem testowym zawierającym testy jednostowe przeprowadzane na API. Niestety ograniczają się jedynie do testowania mechanizmu autoryzacji, gdyż biblioteka Castle.Windsor wykorzystana do stworzenia kontenera IoC nie została jeszcze dobrze zintegrowana z .NET Core i nie ma jeszcze opcji jawnego wstrzykiwania tego serwisu do testowego serwera. Pozostałe testy API zostały wykonane manualnie za pomocą oprogramowania Postman.

4. **Stos technologiczny**

- ASP.NET Core 3.1
- My SQL Connector
- Pakiet Castle (Castle.Core,Castle.Windsor - kontener IoC, Castle.Windsor.Extensions.DependencyInjection - do wykorzystania zaimplementowane kontenera Windsor zamiast podstawowego kontenera dostarczanego przez .NET Core).
- OAuth (usługa pozwalająca m.in. na zabezpieczenie API za pomocą tokenów autoryzacyjnych)
- Microsoft.AspNetCore.Authentication.JwtBearer (do obsługi tokenów autoryzacyjnych przez API)
- Entity Framework Core
- Microsoft.Extensions.DependencyInjection
- MySql.Data.EntityFrameworkCore (interpreter zapytań zgodnych z Entity Framework Core, tłumaczący na zapytania MySQL)
- RestSharp (wykorzystywany do testów jednostkowych i do komunikacji z OAuth)
- Swashbuckle.AspNetCore (biblioteka służąca do stworzenia dokumentacji Swagger)
- Z.EntityFramework.Extensions.EFCore, Z.EntityFramework.Plus.EFCore (biblioteki rozszerzające możliwości Entity Framework Core)

5. **Schemat bazy danych**