Techniczne aspekty zrealizowanych funkcjonalności projektu SmartSchedule

1 Zastosowane technologie i biblioteki

1.1 Backend

- C# 7.3,
- .NET Core 2.2,
- ASP.NET Core
- SQLServer 2016 R2,
- Entity Framework Core,
- MediatR,
- FluentValidation,
- AutoMapper,
- Swagger (Swashbuckle),
- Serilog,
- xUnit,
- Shouldly,
- SonarCloud

1.2 Frontend

- Vue 2,
- Bootstrap Vue
- ECMAScript6

2 Zrealizowane funkcjonalności w backendzie

[1.] Dokumentacja i możliwość testowania API za pomocą Swaggera
[2.] Logger do pliku i/lub konsoli (w zależności od trybu – DEV czy produkcja)
[3.] API uruchomione na platformie Azure przy użyciu wirtualnej maszynie (VM) opartej o system Linux i Dockera
[4.] Continuous Integration do sprawdzania poprawności solucji przy użyciu Azure DevOps
[5.] Testy jednostkowe z wykorzystaniem xUnit
[6.] Kontrola jakości kodu przy użyciu SonarCloud
[7.] Dane zapisywane w bazie danych
[8.] Konta użytkowników:
    a. Autoryzacja (logowanie) oparta o JWT
    b. Rejestracja
    c. Reset hasła oparty o JWT
    d. Dwa typy kont: użytkownika i administratora
[9.] Kalendarz i wydarzenia w kalendarzu:
    a. Zalogowany użytkownik może dodawać, usuwać i edytować kalendarze

b. Zalogowany użytkownik może dodawać, usuwać i edytować wydarzenia w kalendarzu
c. Zalogowany użytkownik może ustalić lokalizację wydarzenia
d. Zalogowany użytkownik może dodać i usunąć znajomego z kalendarza
e. Zalogowany użytkownik może usunąć wszystkie wydarzenia z kalendarza
f. Zalogowany użytkownik może pobrać listę swoich wydarzeń ze wszystkich kalendarzy
g. Zalogowany użytkownik może pobrać listę wydarzeń z wybranego kalendarza

[10.] Znajomi:
a. Zalogowany użytkownik może pobrać listę znajomych, zablokowanych użytkowników, wysłanych i odebranych zaproszeń
b. Zalogowany użytkownik może wysyłać prośby o zostanie znajomym
c. Zalogowany użytkownik może akceptować i odrzucać zaproszenia
d. Zalogowany użytkownik może zablokować i odblokować innego użytkownika
e. Zalogowany użytkownik może usunąć znajomego ze swojej listy znajomych

[11.] Administrator może:
a. dodać kalendarz dowolnemu użytkownikowi
b. wyświetlić listę wszystkich kalendarzy
c. wyświetlić listę wszystkich kalendarzy konkretnego użytkownika
d. wyświetlić listę wyświetlić listę znajomych, zablokowanych użytkowników, wysłanych i odebranych zaproszeń konkretnego użytkownika
e. wyświetlić szczegóły konta konkretnego użytkownika
f. wyświetlić wydarzenia konkretnego użytkownika

[12.] Przeglądarkowy interfejs użytkownika

3 ZREALIZOWANE ZAŁOŻENIA TECHNICZNE REST API

[1.] Clean Architecture REST API inspirowany podejściem DDD (Domain Driven Design) przy użyciu wzorców takich jak CQRS, Mediator, Repozytorium, Unit of Work, co pozwoliło na rozdzielnie logiki i warstwy biznesowej oraz zapewnienie wysokiej skalowalności i wydajności całej aplikacji
[2.] Dodanie lub usunięcie jakiejkolwiek funkcjonalności nie zakłóca działania aplikacji
[3.] Zastosowanie podejścia TDD (Test Driven Development)

4 ZREALIZOWANE FUNKCJONALNOŚCI WE FRONTENDZIE

[1.] Routing
[2.] Widoki:
a. Rejestracja
b. Logowanie
c. Dodawania kalendarza
d. Dodawania wydarzenia
e. Usuwanie kalendarza
f. Usuwanie wydarzenia
g. Edycja kalendarz
h. Edycja wydarzenia
i. Zmiana widoku kalendarza
j. Zmiana dnia
k. Zmiana tygodnia

l. Zmiana miesiąca
m. Działanie przycisku today
n. Zmiana hasła

[3.] Serwis działa bez przeładowywania

[4.]