Aplikacja mobilna dla turystów do zwiedzania Warszawy

Mateusz Bąkała
Paweł Duszak
Mateusz Jabłoński

14 czerwca 2016

Streszczenie

W niniejszym dokumencie przedstawiona zostanie krótka analiza funkcjonalna aplikacji, podział pracy oraz motywacja i potencjał komercyjny końcowego produktu. Zawarte będą również: wykorzystane środowiska i biblioteki, a także opis architektury projektu.

1 Motywacja

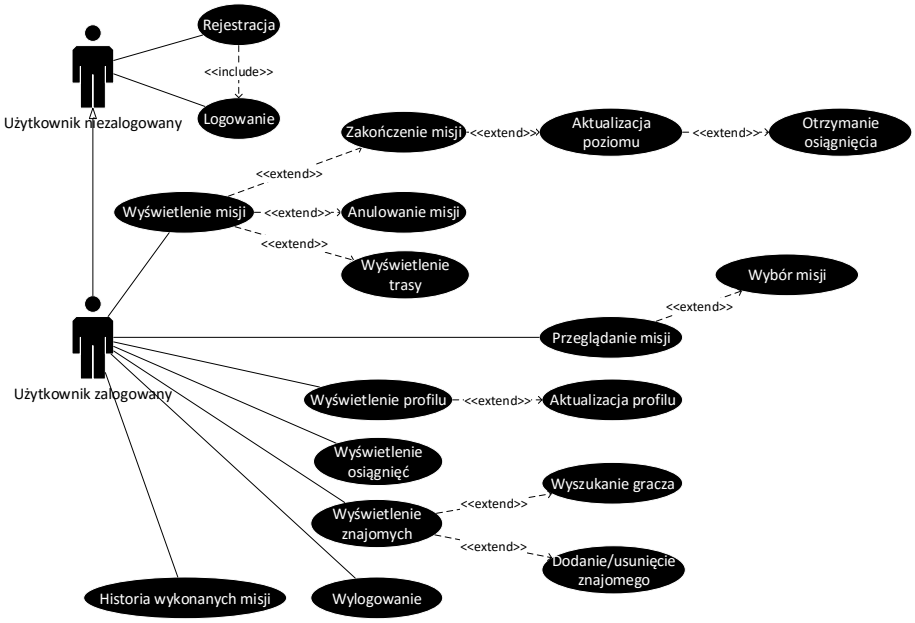
Celem projektu było stworzenie produktu, który będzie wyróżniał się na rynku aplikacji mobilnych. Niemal wszystkie aplikacje dla podróżników dostępne w sklepach oferują szeroką gamę miejsc do zwiedzenia, odpowiednio skategoryzowaną. Interakcja użytkownika z tego typu aplikacjami ogranicza się do wyboru miejsca, zobaczenia szczegółowych informacji na jego temat i nawigacji do punktu docelowego. Użytkownicy w żaden sposób nie są ze sobą powiązani.
Nasz produkt wyróżnia się tym, że użytkownicy wchodzą ze sobą w jakieś interakcje. Aplikacja docelowo miała być utrzymana w klimacie przygodowej gry akcji, w której użytkownicy rywalizują ze sobą. Poza możliwością wyboru atrakcji turystycznej i nawigacji do niej, użytkownik może przeglądać swoją sieć znajomości oraz obserwować swoją pozycję w rankingu. Za wykonywanie misji - odwiedzenie każdego miejsca, które związane jest z konkretną misją - użytkownik zostaje nagrodzony punktami doświadczenia, dzięki czemu jego poziom może wzrosnąć oraz może otrzymać kolejne odznaki osiągnięć.

2 Potencjał komercyjny

Na końcowy sukces aplikacji wpływa kilka kwestii. Pierwszą z nich jest zdobycie bazy użytkowników, gdyż bez tego istnienie aplikacji nie ma sensu. Najłatwiej jest to rozwiązać promując aplikację najpierw w sieci lokalnej i powoli rozszerzać zakres jej użycia na szerszą grupę użytkowników. Kolejnym problemem jest dostarczanie użytkownikom misji i ogólna praca nad tym by użytkownicy nie byli znudzeni aplikacją. KOnieczni są wówczas ludxie, którzy zadbają o dalszy rozwój aplikacji poprzez dodawanie nowych misji itp. Istnieją również mniejsze problemy jak np. dbanie o poprawne dane w systemie.

Reasumując, wdrożenie takiego produktu na rynek jest dość ryzykownym rozwiązaniem i wymagałoby dużego nakładu pracy w kwestii utrzymania aplikacji.

3 Funkcjonalność aplikacji



Na diagramie przypadków użycia można wyróżnić dwóch aktorów: użytkownika zalogowanego i jego uogólnienie - użytkownika niezalogowanego. Użytkownik nieposiadający konta w aplikacji lub niezalogowny ma dostęp do panelu rejestracji i logowania.
Po zalogowaniu staje się aktorem zalogowanym, który ma dostęp do innych aktywności; może uzupełnić swój profil o zdjęcie, krótki opis, zmienić nazwę pod którą jest widoczny przez innych graczy, może zmienić również hasło i adres email. W widoku dostępnych misji może wybrać jedną z sugerowanych dla niego misji do przejścia. W widoku aktywnej misji wyświetlone są szczegóły dotyczące aktualnie wykonywanej misji, w tym miejscu można zakończyć lub anulować misję, lub też przejść do widoku mapy w którym wyświetlona jest proponowana trasa do punktu docelowego. Po wykonaniu misji użytkownik otrzymuje punkty doświadczenia, dzięki którym może zwiększyć swój poziom w grze oraz otrzymać dodatkowe odznaki osiągnięć. W pozostałych widokach dostępna jest historia wykonanych misji, zdobyte osiągnięcia oraz ranking znajomych. Użytkownik zalogowany może się również wylogować lub wspomóc twórców aplikacji za pośrednictwem serwisu Paypall. Dostępny jest również krótki wykaz zespołu programistów tworzących aplikację.

4 Wykorzystane środowiska

Ważną częścią projektu jest również serwer napisany w architekturze REST udostępniający API dla klientów aplikacji. Stworzony został w technologii ASP.NET Web API. Solucja ma wielowarstwową architekturę, w której niższe warstwy udostępniają serwisy umożliwiające dostęp do danych, natomiast wyższe warstwy odpowiadają za logikę biznesową. Istnieje również kilka warstw pomocnicznych takich jak np. modele encji bazy danych co umożliwiło skorzystanie z mechanizmu migracji bazy danych. Do mapowania obiektowo-relacyjnego wykorzystano bibliotekę Entity Framework. W celu zarządzania kontami użytkowników oraz zapewnienia ich autoryzacji z serwerem użyto bibliotek ASP.NET Identity oraz zintegrowanej z nimi biblioteki OWIN. Aby ograniczyć ilość przesyłanych danych przez sieć serwer skonfigurowany został tak by zwracać obiekty DTO w postaci plików w notacji JSON.
Solucja wykorzystuje kilka wzorców architektonicznych i projektowych, m.in.:

- Dependency Injection (wstrzykiwanie zależności, biblioteka Autofac)
- Generic repository pattern
- Unit of work pattern
- w implementacji wykorzystano wzorce projektowe fasady, kompozycji, dekoratora i singletonu

Wykorzystano wiele dodatkowych bibliotek, np. AutoMapper do mapowania obiektów, Moq do mokowania klas na potrzeby testowania aplikacji, xUnit do pisania testów jendostkowych i inne. Serwer został wdrożony na platformę Microsoft Azure, licencję uzyskano dzięki programowi dla rozwijających się startupów - Microsoft Bizspark. W ramach solucji serwera skonfigurowaliśmy również mechanizm Continuous Deployment który ułatwił pracę i testowanie aplikacji po stronie użytkownika. Wykorzystana w projekcie baza danych to relacyjna baza danych MS SQL (T-SQL), wzniesiona na serwerze MS SQL Server.

Aplikacja po stronie klienta również tworzy architekturę wielowarstwową. Niższe warstwy odpowiadają za dostęp do danych, natomiast wyższe odpowiadają za logikę aplikacji i warstwę prezentacji. Wykorzystane biblioteki to m.in.:

- Dagger (realizacja wzorca Dependency Injection)
- Retrofit (klient HTTP)
- Butter Knife (dynamiczne wiązanie pól i metod)
- gson (mapowanie obiektów JSON do klas)
- javax (atrybuty wspomagające mapowanie obiektów)
- android-image-cropper (przycinanie obrazków)
- CircleImageView (okrągłe grafiki)
- CardView, RecyclerView

Aplikacja jest dostępna na telefony z systemem android z minimalną wersją SDK 16.

5 Podział pracy

Podczas projektu zespół pracował w oparciu o metodykę programowania ekstremalnego. W początkowych iteracjach zajmowano się głównie tworzeniem architektury i projektów, w kolejnych etapach prace przybrały bardziej pionowy model, tj. każdy z developerów tworząc daną funkcjonalność realizował ją zarówno po stronie serwera i bazy danych, jak i po stronie klienta (czasem z wyjątkiem tzw. front-endu).
Projekt dostępny jest na licencji MIT pod adresem:
https://github.com/Koteczeg/WarsawCityGame
Historia repozytorium najlepiej odzwierciedla zakres wykonywanych zadań przez poszczególnych programistów.

Ogólny zakres prac każdej osoby:

Mateusz Bąkała:

- tworzenie struktury bazy danych
- tworzenie funkcjonalności: profil, ranking użytkowników, historia misji, mapa (backend)
- mockowanie na potrzeby testów po stronie serwera
- seedowanie bazy danych, tworzenie misji

Paweł Duszak:

- budowa architektury po stronie serwera i klienta
- mockowanie na potrzeby testów po stronie serwera
- testy jednostkowe po stronie serwera
- wdrażanie serwera i bazy danych na platformę Microsoft Azure, program Microsoft BizSpark
- tworzenie funkcjonalności: profil, osiągnięcia, znajomi, historia misji (frontend & backend)
- tworzenie funkcjonalności: ranking, wybór misji, bieżąca misja, historia misji (frontend)
- konfiguracja połączenia klient-serwer, autoryzacja z serwerem
- tworzenie menu głównego, grafik, animacji

Mateusz Jabłoński:

- tworzenie funkcjonalności: wybór misji (backend & frontend)
- tworzenie funkcjonalności: bieżąca misja (backend)
- tworzenie funkcjonalności: lokalizacja & mapa (backend)
- testy jednostkowe po stronie serwera
- wsparcie merytoryczne

6 Podsumowanie

Głównym celem projektu było poznanie środowiska Android od strony programistycznej i stworzenie skalowalnej aplikacji. Końcowy produkt spełnia założenia postawione na początku oraz jest przygotowany do dalszego rozwoju. Przygotowanie projektu pozwoliło na zapoznanie się z zagadnieniem programowania aplikacji mobilnych oraz ich integracji z rozproszonymi systemami.