

ZMP_Obserwowator cen produktów w sklepie internetowym STEAM

Piotr Tekieli,
Jan Kwiatkowski,
Mariusz Skuza,
Szymon Bacański

Kwiecień 2022

1 Opis funkcjonalny systemu

Serwis internetowy zostanie wykonany w oparciu o specyfikację projektową. Projektowany od początku serwis musi być elastyczny, tj. powinien umożliwić rozwój istniejących i dodawanie nowych funkcjonalności bez konieczności przebudowy znacznych części kodu lub architektury.

Celem projektu jest stworzenie aplikacji do obserwowania cen produktów w sklepie internetowym STEAM. Serwis będzie się składał z aplikacji WEB, Mobile, Desktop, które będą łączyć się do stworzonym API.

1.1 Opis funkcjonalny API

Api będzie pełniło funkcję głównego węzła komunikacyjnego dla systemu poprzez:

1. Wysyłanie i odbieranie informacji do innych aplikacji odnośnie użytkowników jak i przypisanych do konta informacji
2. Prowadzenie bazy danych z informacjami odnośnie kont i gier
3. Aktualizacja danych z sklepu STEAM

API ma w sposób dostępny i szybki umożliwić innym aplikacjom na dostęp do informacji wiązanych z treścią serwisu.

1.2 Opis funkcjonalny Web

Witryna internetowa wyposażona w narzędzia ułatwiające nawigację i orientację w zawartości serwisu. Główną jej funkcją będzie kontakt z użytkownikami czyli umożliwienie dostępu do informacji serwisu i czytelne przedstawienie ich. Według wstępnych założeń witryna zawierać będzie:

1. Strona logowania/rejestracji
2. Wyszukiwarke
3. Odnośnik do dodania gry do obserwowanych
4. Wgląd do historii ceny obserwowanego produktu

Dopuszcza się modyfikację tej struktury na etapie budowy innych elementów serwisu.

1.3 Opis funkcjonalny Desktop

Aplikacja na komputer pozwalająca korzystać z serwisu.
Według wstępnych założeń Aplikacja zawierać będzie:

1. Strona logowania/rejestracji
2. Wyszukiwarke
3. Dodanie gry do obserwowanych
4. Usuwanie z obserwacji
5. Wgląd do historii ceny obserwowanego produktu poprzez diagram liniowy
6. Automatyczne logowanie

Dopuszcza się modyfikację tej struktury na etapie budowy innych elementów serwisu.

1.4 Opis funkcjonalny Mobile

Aplikacja mobilna na androida pozwalająca korzystać z serwisu.
Według wstępnych założeń Aplikacja zawierać będzie:

1. Strona logowania/rejestracji
2. Wyszukiwarke
3. Dodanie gry do obserwowanych
4. Usuwanie z obserwacji
5. Wgląd do historii ceny obserwowanego produktu poprzez diagram liniowy
6. Automatyczne logowanie

Dopuszcza się modyfikację tej struktury na etapie budowy innych elementów serwisu.

2 Wykorzystywane technologie

2.1 API

- Node.js
Wieloplatformowe opensource środowisko do tworzenia aplikacji typu server-side napisanych w języku JavaScript.
- MongoDB
Otwarty, nierelacyjny system zarządzania bazą danych napisany w języku C++. Charakteryzuje się brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Zamiast tego dane składowane są jako dokumenty w stylu JSON.

2.2 Web

- React.js
Biblioteka języka programowania JavaScript, która wykorzystywana jest do tworzenia interfejsów graficznych aplikacji internetowych. Zainspirowana przez rozszerzenie języka PHP – XHP. Często wykorzystywana do tworzenia aplikacji typu Single Page Application

2.3 Desktop

- Electron
Otwarty Źródłowa platforma programistyczna pozwalająca tworzyć aplikacje GUI dla komputerów stacjonarnych za pomocą komponentów elementów front-endowych i back-endowych, opracowanych początkowo dla aplikacji sieciowych: Node.js (back-end) i Chromium (front-end). Electron jest główną strukturą GUI za kilkoma znaczącymi projektami open source, w tym edytorami kodu źródłowego Atom i Visual Studio Code oraz czatem Discord.
- Chromium
Otwarty projekt przeglądarki internetowej, z którego kod źródłowy czerpią między innymi Google Chrome, Opera czy Microsoft Edge.
- Node.js
Wieloplatformowe opensource środowisko do tworzenia aplikacji typu server-side napisanych w języku JavaScript.

2.4 Mobile

- Xamarin.Forms
To wieloplatformowa abstrakcyjna platforma narzędziowa z interfejsem użytkownika, która umożliwia programistom łatwe tworzenie interfejsów użytkownika, które można udostępniać na urządzeniach z Androidem, iOS, Windows i Windows Phone.

- Xamarin.Community
Zestaw narzędzi społeczności Xamarin to kolekcja animacji, zachowań, konwerterów i efektów do tworzenia aplikacji mobilnych przy użyciu platformy Xamarin.Forms. Upraszcza i demonstrowuje typowe zadania deweloperskie.

3 Wzorce projektowe

3.1 API

- Circuit Breaker
Wzorzec Circuit Breaker, zapewnia sposób na wykrycie popsutej zależności i zatrzymuje przepływ danych i pozwala uniknąć opóźnień i okropnego UX. Wykorzystujemy ten wzorzec do komunikacji z serwisami wewnętrznymi.

3.2 Web

- Singleton
Singleton jest wzorcem, który pozwala na stworzenie tylko jednej instancji obiektu z klasy bądź konstruktora funkcyjnego. W przypadku wielokrotnego wywoływania tej samej klasy zawsze będziemy otrzymywali tę samą instancję, która została stworzona podczas pierwszego wywołania.

3.3 Desktop

- Budowniczy
Budowniczy jest kreatywnym wzorcem projektowym, który daje możliwość tworzenia złożonych obiektów etapami, krok po kroku. Wzorzec ten pozwala produkować różne typy oraz reprezentacje obiektu używając tego samego kodu konstrukcyjnego.

3.4 Mobile

- MVVM
Wzorzec Model-View-ViewModel - opiera się na wydzieleniu odpowiednich warstw w systemie, w celu podziału zadań oraz zmniejszenia zależności pomiędzy klasami. Mamy więc klasy modelu danych, których zadaniem jest przechowywanie danych właśnie oraz ich ewentualną walidację.

4 Instrukcję lokalnego i zdalnego uruchomienia systemu

4.1 API

4.1.1 Postawienie aplikacji lokalnie

1. Clone the github repository
git clone https://github.com/Price-Tracker-ZMP/API/
2. Open the project and install NPM
npm install
3. Run the application with:
npm start

4.1.2 Postawienie aplikacji zdalnie

Na maszynie (serwerze) z zainstalowanymi pakietami:

1. nodejs
2. pm2
3. git
4. nginx

Z pomocą połączenia ssh.

1. Clone the github repository
git clone git@github.com:Price-Tracker-ZMP/API.git
2. Open the project and install NPM
npm install
3. Run the application with:
pm2 start api.js

4. Change port on file:
sudo nano /etc/nginx/sites-enabled/default
5. Reload file
sudo /etc/init.d/nginx reload

4.1.3 Instrukcja uruchomienia testów

1. Testy uruchamia w terminalu
npm run tests

4.2 Web

4.2.1 Postawienie aplikacji lokalnie

1. Clone the github repository
git clone https://github.com/Price-Tracker-ZMP/Web/
2. Open the project and install NPM
npm install
3. Run the application with:
npm start

4.2.2 Postawienie aplikacji zdalnie

1. Wchodzimy na strone vercel.com
2. Logujemy się z pomoca github
3. Importujemy dane repozytorium z aplikacją nazwyając ją
4. Ustawiamy framework na react app i klikamy deploy
5. Czekamy za zbudowanie
6. Strona dostępna w internecie dod domeną "nazwa-projektu".vercel.com

4.2.3 Instrukcja uruchomienia testów

1. Testy uruchamia w terminalu
npm run tests

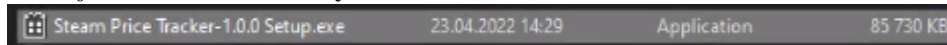
4.3 Desktop

4.3.1 Postawienie aplikacji deweloper

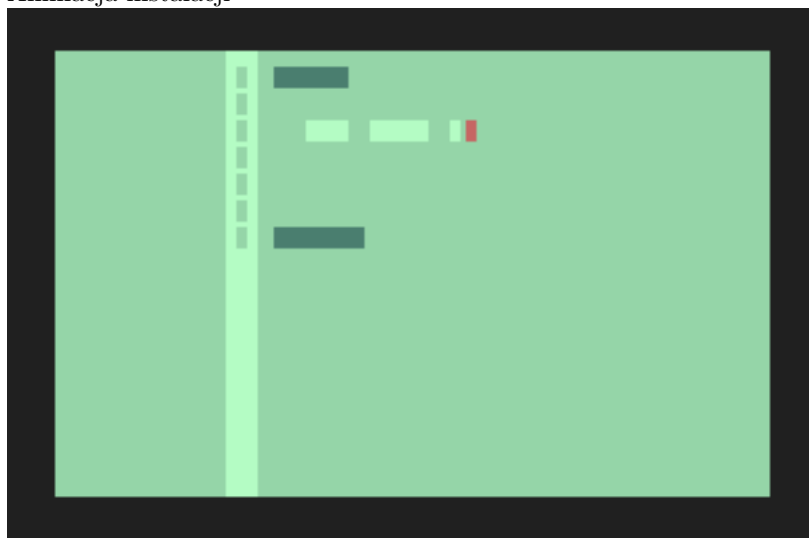
1. Clone the github repository
git clone https://github.com/Price-Tracker-ZMP/Desktop/
2. Open the project and install NPM
npm install
3. Run the application with:
npm start

4.3.2 Postawienie aplikacji użytkownik

1. Kliknij drukrotnie w instalkę



2. Animacja instalacji



3. Aplikacja gotowa do użytku

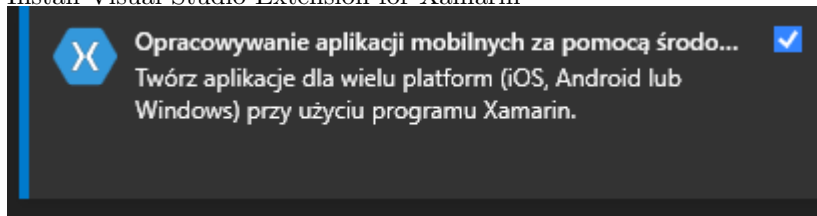
4.3.3 Instrukcja uruchomienia testów

1. Testy uruchamia w terminalu
npm run tests

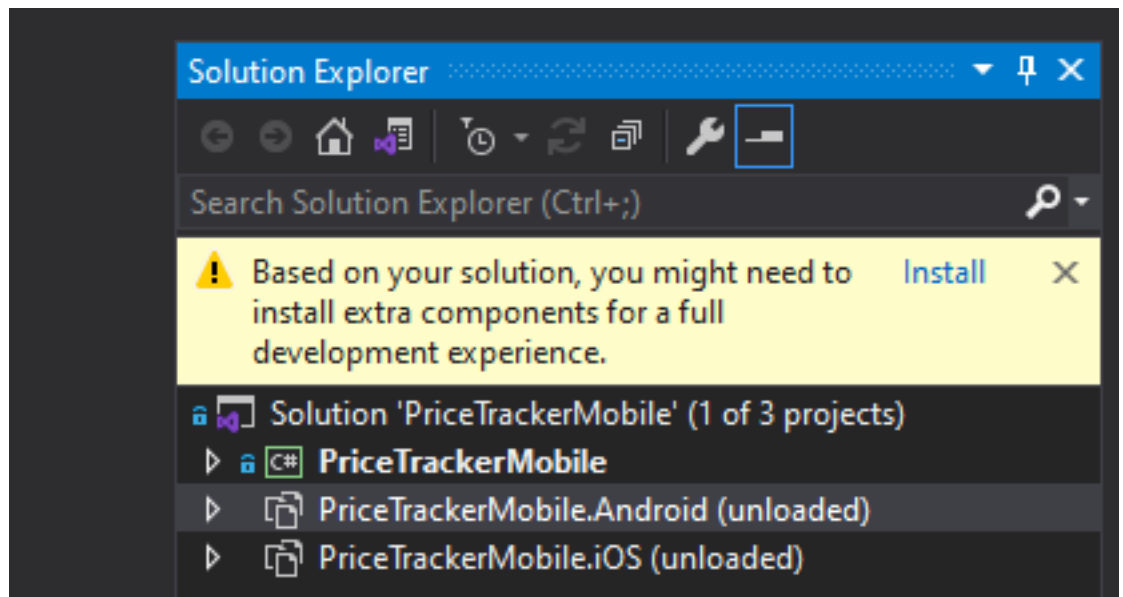
4.4 Mobile

4.4.1 Postawienie aplikacji deweloper

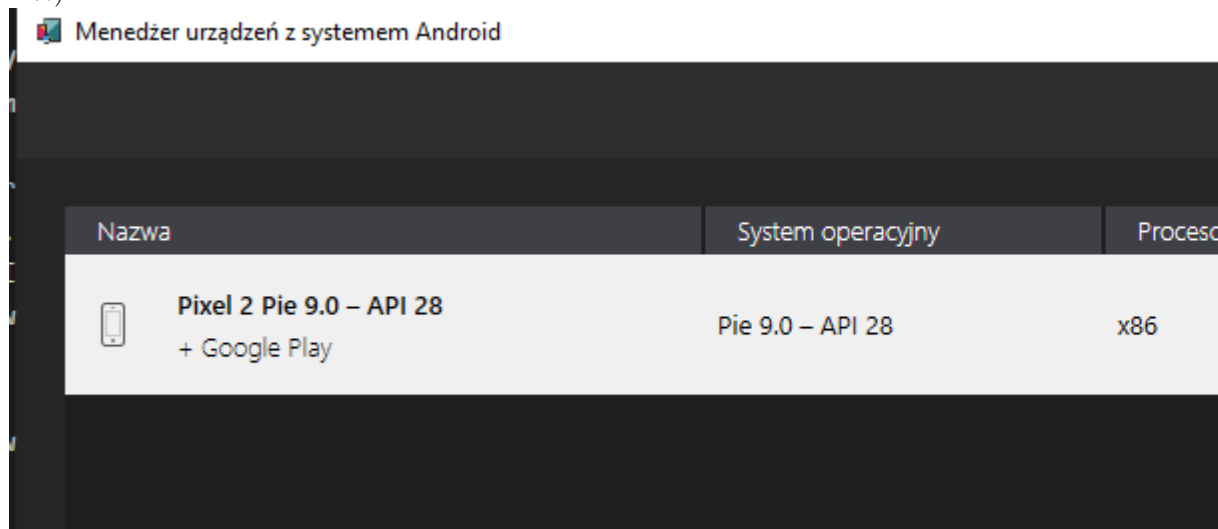
1. Install Visual Studio
2. Install Visual Studio Extension for Xamarin



3. Clone repository
[Git repository](#)
4. Open project (install additional if necessary)



5. Start project (during first start, install Android emulator and default device)



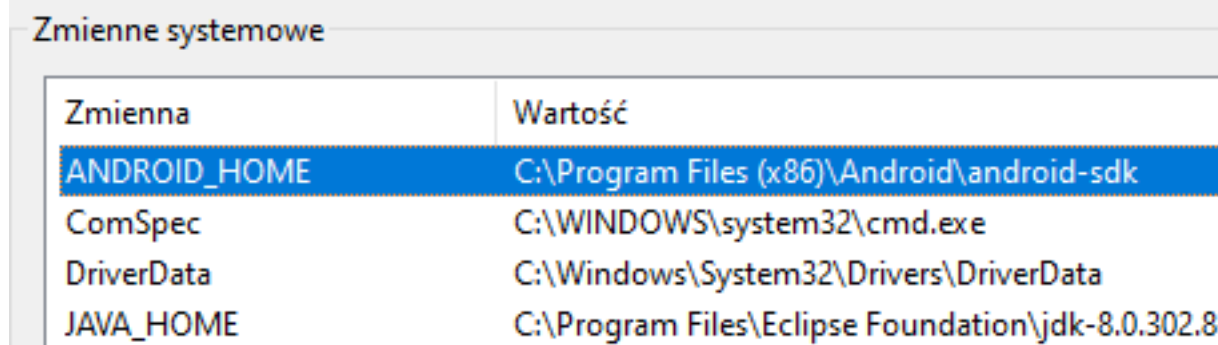
4.4.2 Postawienie aplikacji użytkownik

1. Wejdź na sklep play.

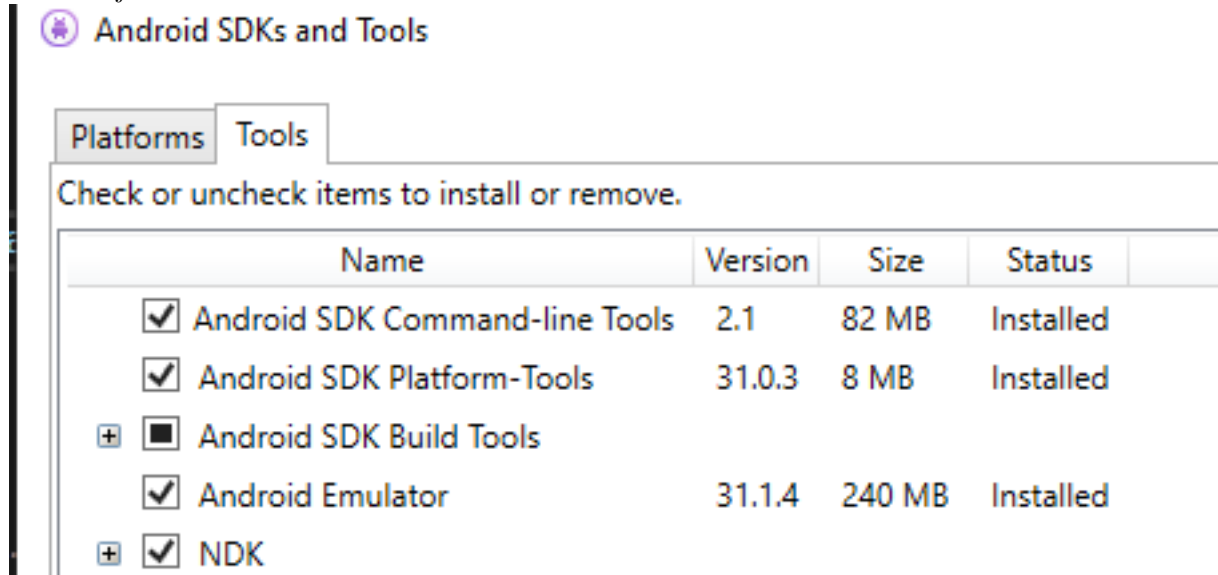
2. Znajdź aplikację pod nazwą PriceTracker lub wejdź w link <https://play.google.com/store/apps/details?id=com.merfeusz.pricetrackermobile>
3. Kliknij zainstaluj

4.4.3 Instrukcja uruchomienia testów

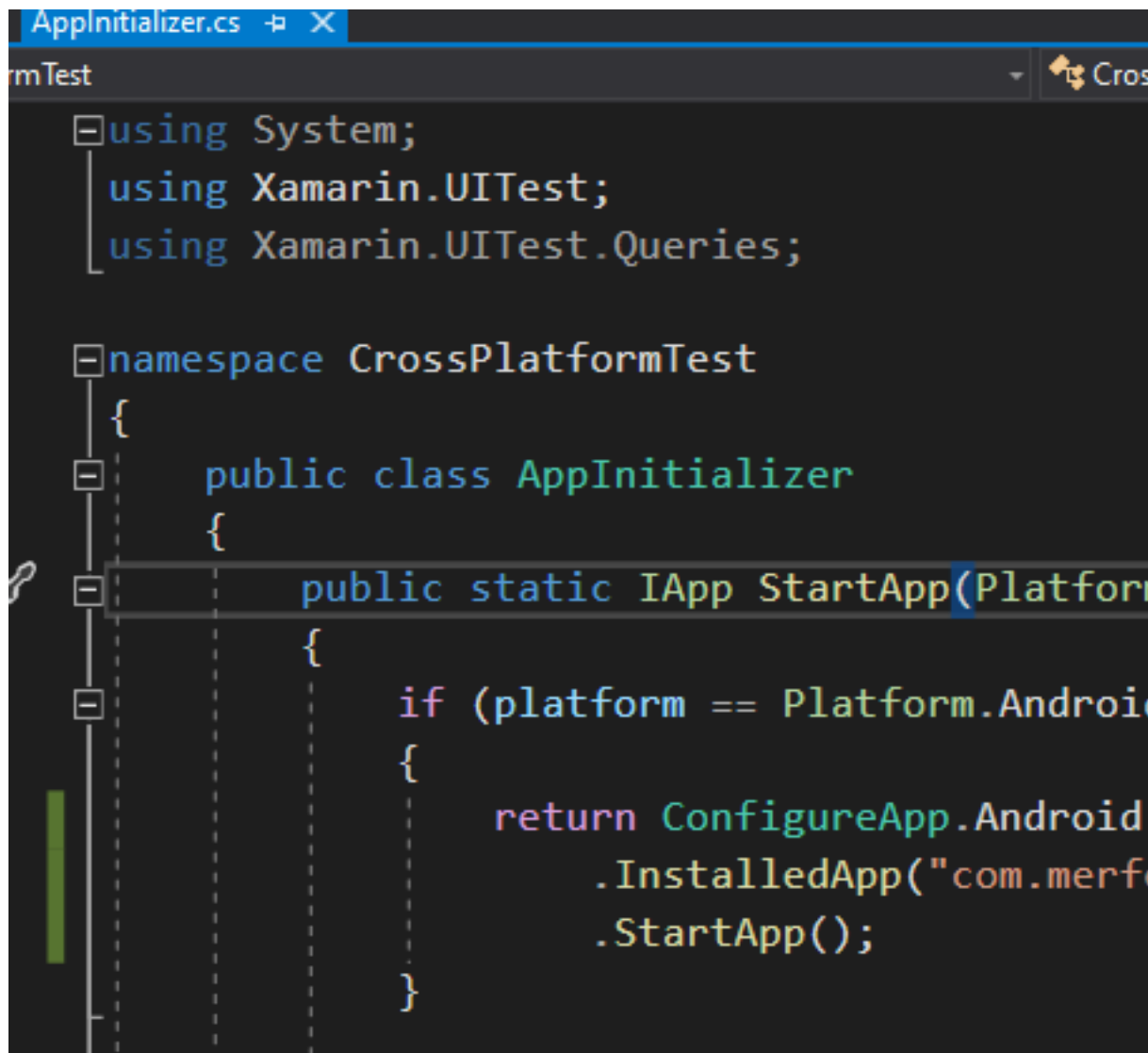
1. Dodaj zmienne środowiskowe (ANDROID_HOME i JAVA_HOME)” potrzebne do uruchomienia testów.



2. Zainstaluj NDK.



3. Skonfiguruj package name aplikacji dla inicjatora.



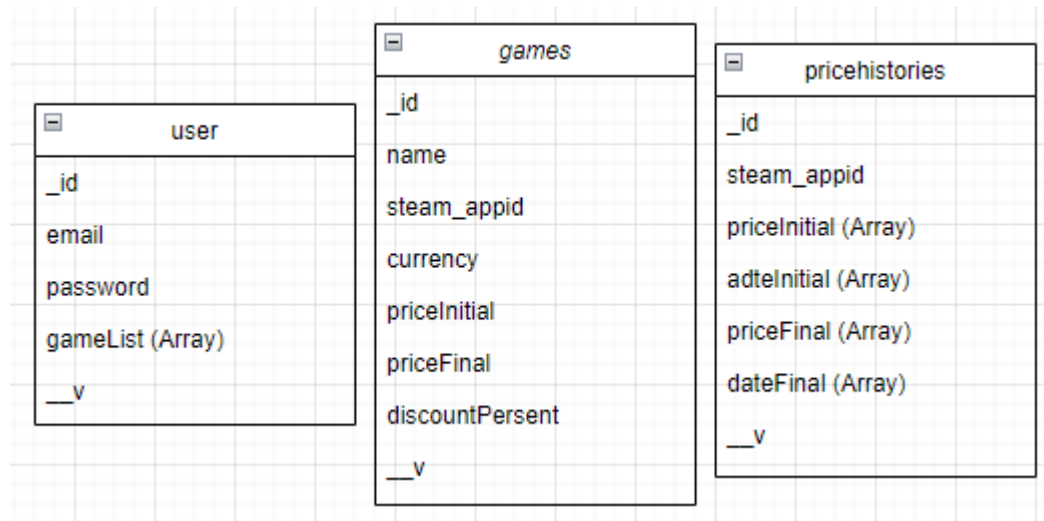
```
AppInitializer.cs
using System;
using Xamarin.UITest;
using Xamarin.UITest.Queries;

namespace CrossPlatformTest
{
    public class AppInitializer
    {
        public static IApp StartApp(Platform platform)
        {
            if (platform == Platform.Android)
            {
                return ConfigureApp.Android
                    .InstalledApp("com.merf")
                    .StartApp();
            }
        }
    }
}
```

4. Uruchom projekt na emulatorze, aby mieć pewność, że aplikacja jest zainstalowana.
5. Uruchom testy w zakładce Test Explorer.
Propane zdane testy

▲	✓	CrossPlatformTest (10)	2,7 min
▲	✓	CrossPlatformTest (10)	2,7 min
▲	✓	LoginPageTests(Android) (6)	1,5 min
	✓	NavigateToRegisterPage	14,5 sec
	✓	ShowErrorToastWhenLoginWithEm...	15,1 sec
	✓	ShowErrorToastWhenLoginWithWr...	15,5 sec
	✓	ShowErrorToastWhenLoginWithWr...	16,8 sec
	✓	ShowSuccessToastWhenLogin	18,9 sec
	✓	WelcomeTextOnLoginPage	11,3 sec
▲	✓	RegisterPageTests(Android) (4)	1,1 min
	✓	ShowErrorToastWhenRegisterWith...	17,7 sec
	✓	ShowErrorToastWhenRegisterWith...	20,4 sec
	✓	ShowErrorToastWhenRegisterWith...	15,5 sec
	✓	WelcomeTextOnRegisterPage	13,5 sec

5 Schematy i diagramy



6 Repozytorium

Link do naszej organizacji:

<https://github.com/Price-Tracker-ZMP>

7 Wnioski

Projekt ten nie był aż tak prostym przedsięwzięciem jak zakładaliśmy na starcie. Jednak ucząc się po drodze udało się nam go ukończyć. Nigdy wcześniej część naszego zespołu nie korzystała z jawa skryptu i innych wyczytanych technologii, co także utrudniło pracę. Więc musieliśmy pomagać sobie, a najbardziej osobom, które nie były zaznajomione z ów językiem, aby poprawnie utworzyć naszą aplikację.

Poza techniczną stroną projektu musieliśmy się też zmierzyć z trudność zarządzania czasem z powodu natłoku innych sytuacji losowych i pracy zawodowej. Ale miło to też dobrą stroną, ponieważ niektórzy członkowie naszego zespołu mieli już styczność z wykorzystywanymi technologiami w życiu zawodowym. Musiała to być praca wspólna, aby osiągnąć nasz cel końcowy.

Wynieśliśmy z tego następujące wnioski:

- Przygotowanie aplikacji wymaga wiedzy poza czystą wiedzą techniczną.

- Warto posiadać członków zespołu, którzy posiadają głęboką wiedzę w danym zakresie.
- Organizacja grupy może być wymagająca, gdy niektórzy członkowie są osobami pracującymi.
- Flow pracy może łatwo być przerywany w wypadku chorób i różnych prywatnych spraw w zespole.
- Nie należy być zbyt optymistycznym w przypadku planowania czasu pracy.