AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Runly: Running App

Autor:

Łukasz Skraba
Bartosz Tobiasz

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2022

Spis treści

1. Ogólne określenie wymagań

1.1. Opis działania

Aplikacja na urządzenia mobilne umożliwiająca monitoring dokonań sportowych w dziedzinie biegania. Program ma umożliwić monitorowanie naszej aktywności biegowej. Aplikacja ma zapisywać przede wszystkim czas treningu, dystans, trasę uzyskaną dzięki modułowi GPS oraz intensywność treningu (np. wyliczając średnie tempo, średnią i maksymalną prędkość oraz skalone kalorie). Kożystając z aplikacji mamy mieć możliwość szczegółowej weryfikacji danych treningu, zarówno w trakcie jego trwania jak i po jego zakończeniu. Dodatkowo w podsumowaniu dzięki współpracy programu z GPS-em, można także spawdzić informacje o najniższym i najwyższym punkcie trasy. Szczegółowe statystyki mają pozwolić na analizę postępów i wyciągnięcie wniosków na przyszłość.

Treningi mają być zapisywane w pamięci. Użytkownik ma mieć możliwość zobaczenia statystyk wybranego treningu.

Aplikacja ma za zadanie także motywować nas do ćwiczeń, np. wysyłając nam powiadomienia, w ustalonym przez użytkownika momencie, o tym, że nie odbyliśmy jeszcze treningu.

Poza pomiarami w trakcie treningu, aplikacja ma także liczyć kroki, kiedy działa w tle.

1.2. Opis wyglądu

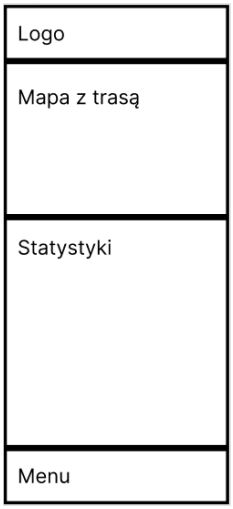
Na głównej stronie treningu, którą widzi użytkownik po otwarciu aplikacji, Powinny znajdować się takie informacje jak:

- czas trwania aktywności,
- prędkość w danym momencie,
- średnia prędkość,
- dystans,
- spalone kalorie,

Oprócz tego na stronie treningu Rys. 1.1 (s. 4) powinna znajdować się mapa, na której będzie pokazana aktualna pozycja uzytkownika, oraz przebyta trasa.

Po zakończonym treningu aplikacja ma pokazać całą przebytą trasę na mapie oraz dać dostęp do szczegółowych statystyk treningu Rys. 1.2 (s. 4). Użytkownik ma mieć podgląd na wszystkie możliwe dane.

Rys. 1.1. Ekran treningu



Rys. 1.2. Ekran podsumowania

Ekran krokomierza Rys. 1.3 (s. 4) ma zawierać tylko liczbę zrobionych w biezacym dniu kroków.

Rys. 1.3. Ekran krokomierza

Na dole aplikacji ma się znajdować menu za pomocą którego użytkownik może się przełączać pomiędzy ekranem krokomierza, treningu, odbytymi treningami i ustawieniami.

Ekran zawierający historię odbytych treningów powinien przedstawiać je w postaci list. Ustawienia też powinny być przedstawione w postaci listy.

2. Określenie wymagań szczegółowych

2.1. Środowisko programistyczne

Aplikacja zostanie napisana korzystając z platformy Xamarin.Forms. Jest to środowisko umożliwiające tworzenie aplikacji za pomocą języka XAML oraz kodu w języku C#. Xamarin to platforma typu open source do tworzenia nowoczesnych i wydajnych aplikacji dla systemów iOS, Android i Windows za pomocą platformy .NET. Xamarin to warstwa abstrakcji, która zarządza komunikacją udostępnionego kodu z bazowym kodem platformy. Xamarin.Forms umożliwia deweloperom tworzenie aplikacji Xamarin.iOS, Xamarin.Android i Windows z pojedynczej udostępnionej bazy kodu. Xamarin.Forms Umożliwia deweloperom tworzenie interfejsów użytkownika w języku XAML z użyciem kodu w języku C#. Te interfejsy użytkownika są renderowane jako wydajne kontrolki natywne na każdej platformie.
Oto kilka przykładów funkcji udostępnianych przez Xamarin.Forms :

- Język interfejsu użytkownika XAML
- Powiązanie danych
- Gestów
- Efekty
- Style

Dodatkowo, w celu koordynacji wykonywania projektu, organizacji kodu oraz ogólnego wspomagania pracy wykorzystany zostanie system kontroli wersji oprogramowania Git. Zapewnia on:

- Dobre wsparcie dla rozgałęzionego procesu tworzenia oprogramowania: jest dostępnych kilka algorytmów łączenia zmian z dwóch gałęzi, a także możliwość dodawania własnych algorytmów,
- Każdy programista posiada własną kopię repozytorium, do której może zapisywać zmiany bez połączenia z siecią; następnie zmiany mogą być wymieniane między lokalnymi repozytoriami. Programista może także dodawać oraz usuwać gałęzie,
- Efektywną pracę z dużymi projektami, jest o rzędy wielkości szybszy niż niektóre konkurencyjne rozwiązania

2.2. Ogólny wygląd interfejsu

Aplikacja będzie podzielona na podstrony. Na dole będzie znajdować się menu z opcjami. Wybór poszczególnej opcji w menu wyświetli daną podstronę w ekranie aplikacji. Kolejne podstrony to: Kroki, Trening, Historia, Ustawienia.

2.3. Krokomierz

W zakładce Kroki aplikacja będzie wyświetlała, korzystając z odpowiedniego sensora (krokomierza), liczbę kroków jaką użytkownik wykonał od rozpoczęcia treningu. Moduł krokomierza jest wymagany by zwiększyć wiarygodność aplikacji. Zastosowanie krokomierza umożliwia wykluczenie wpisów podejrzanych, które mogłyby wystąpić, gdyby użytkownik jechał na przykład rowerem lub samochodem. Takie rozwiązanie jest normalne w tego typu aplikacjach. Oprogramowanie Android w wersjach 4.4 i wyższych posiada wsparcie dla sensorów takich jak detektor kroków oraz licznik kroków. Kroki podczas biegu są łatwiejsze do odróżnienia od tych podczas zwykłego spaceru ze względu na bardziej wyraźne oddziaływanie na sensory. Tak więc jest duże prawdopodobieństwo, że krokomierz będzie bardzo dokładnie mierzył kroki, a system będzie działał niezawodnie w wyjątkowych sytuacjach.

2.4. Moduł GPS

W zakładce Trening aplikacja będzie wyświetlała pomiar biegu użytkownika, a dokładniej mówiąc pomiar przebiegniętego dystansu, prędkości biegu w danej chwili oraz czasu . Odbywa się to na podstawie informacji o jego pozycji. Można to zrealizować na kilka sposobów. Te sposoby to: wykorzystanie globalnego systemu pozycjonowania (GPS), technologia lokalizacji wieży komórkowej lub lokalizacja za pomocą WiFi Powinno się wybrać sposób najbardziej odpowiedni dla naszej aplikacji biorąc pod uwagę przede wszystkim środowisko w jakim będzie ona używana. W naszym przypadku jest to system GPS, gdyż zapewnia on najdokładniejsze dane lokalizycyjne, wykorzystuje najwięcej mocy i działa najlepiej na zewnątrz, co w naszym przypadku jest kluczowe dla odpowiedniego działania aplikacji. Zakładając, że użytkownik przemieszcza się, można zdefiniować jego pozycję z dokładnością od około 6 do 100 metrów.

Aplikacja z obsługą lokalizacji wymaga dostępu do czujników sprzętowych urządzenia w celu odbierania danych GPS. Dostęp jest kontrolowany za pomocą odpowiednich uprawnień w manifeście Androida aplikacji (plik AndroidManifest.xml). Dostępne są dwa uprawnienia:

ACCESS_COARSE_LOCATION - zapewnia aplikacji dostęp do dostawcy GPS oraz *ACCESS_FINE_LOCATION* - umożliwia aplikacji dostęp do sieci komórkowej i WiFi lokalizacji. Wymagane dla dostawcy sieci gdy ACCESS_COARSE_LOCATION jest nieustawiony.

2.5. Mapy Google

W zakładce Trening będzie wyświetlana również, korzystając z modułu GPS, aktualna pozycja użytkownika na Mapach Google. Dostęp do Map Google jest możliwy dzięki API (Maps SDK for Android) udostępnianego przez Google. Dostęp do tego API jest uzyskiwany z kluczem API, który został wygenerowany w panelu Google Cloud API. Możliwość wyświetlania pobranej mapy w aplikacji jest możliwa korzystając z pakietu NuGet Xamarin.Forms.Maps. Na mapie będzie także rysowana linią (klasa Polyline z wyżej wymienionego pakietu) trasa przebyta w trakcie treningu.

2.6. Baza danych SQLite

SQLite to mała, szybka i przystosowana do osadzania baza danych SQL oparta na systemie plików typu open source. Nie posiada oddzielnego komponentu serwerowego, jak tradycyjne bazy danych.

3. Projektowanie

3.1. Przygotowanie narzędzi (Git, Visual Studio)

Pierwszym i oczywistym krokiem, który musimy wykonać jest przygotowanie właściwych technologii i środowisk, które posłużą do tworzenia naszego projektu. Jako środowisko programistyczne wybrano Visual Studio 2022 i dostępną na nim wspomnianą wcześniej platformę Xamarin.Forms. W celu instalacji tego narzędzia, pobieramy plik instalacyjny ze strony Microsoftu. Podczas instalacji, w menedżerze dodatkowych pakietów, wybieramy pakiet *Mobile developement with .NET*, zaznaczając też opcjonalne dodatki w zależności od wymagań (np. Xamarin, emulator Androida itd.). Środowisko Visual Studio posiada zintegrowany system kontroli wersji oprogramowania Git, który będzie wspomagał naszą pracę.

Na stronie Github stworzone zostało zdalne repozytorium, do którego wysyłane będą kolejne wersje naszego projektu, z możliwością kontrybucji ze strony członków zespołu programistów.

3.2. Wybrane kody i pseudokody z objaśnieniami działania

3.2.1. Zakładka Historia

Inicjalizacja połączenia z bazą danych:

```
public History()
{
InitializeComponent();

_database = new SQLiteAsyncConnection(Path.Combine(Environment.
GetFolderPath(Environment.SpecialFolder.LocalApplication
Data), "trainingHistory.db3"));
}
```

Wypisanie danych na ekran:

```
protected override async void OnAppearing()
{
base.OnAppearing();

//Wypisanie danych na ekran
```

```
collectionView.ItemsSource = await GetTrainingData();
}
```

Pobranie danych z bazy danych:

```
public async Task<List<TrainingData>> GetTrainingData()
{
var query = await _database.Table<TrainingData>().ToListAsync();
results = Enumerable.Reverse(query).ToList();
return results;
}
```

Funkcja otwierająca widok statystyk, podsumowania:

```
private async void OpenStatistics(object sender, EventArgs e)
{
var btn = (Button)sender;
await Navigation.PushAsync(new StatisticsView(btn.ClassId));
}
```

3.2.2. Zakładka Trening

Połączenie z bazą danych:

```
public Training()
{
InitializeComponent();

//Połączenie z bazą danych
_database = new SQLiteAsyncConnection(Path.Combine(Environment.
GetFolderPath(Environment.SpecialFolder.
LocalApplicationData), "trainingHistory.db3"));
_database.CreateTableAsync<TrainingData>();
GetLocation();
}
```

Funkcja pobierająca lokalizację:

```
private async void GetLocation()
{
```

```
//Pobranie lokalizacji
var request = new GeolocationRequest(GeolocationAccuracy.Best);
var location = await Geolocation.GetLocationAsync(request);

if (location != null)
{
position = new Position(location.Latitude, location.Longitude);

//Przesunięcie widoku na lokalizację przeciwnika
map.MoveToRegion(MapSpan.FromCenterAndRadius(position, Distance.
FromKilometers(0.5)));

if (isTraining)
{
//Zapis danych lokalizacji
positionsList.Add(new PositionList { Location = location, TimeLasted =
(hours * 3600 + mins * 60 + secs) });
await SaveCurrentData(new CurrentData
{
Latitude = location.Latitude,
Longitude = location.Longitude,
TimeLasted = (hours * 3600 + mins * 60 + secs)
});
UpdateInfo();
}

}

//Pobranie lokalizacji wywoływane co 2 sekundy
await Task.Delay(2000);
GetLocation();

}
```

Wyświetlenie dystansu w określonym formacie zależnym od dystansu:

```
if (way < 1)
{
```

```
amountDistance.Text = (way * 1000).ToString();
distanceSize.Text = " m";
}
else if (way < 10)
{
amountDistance.Text = string.Format("{0:0.00}", way);
distanceSize.Text = " km";
}
else if (way < 100)
{
amountDistance.Text = string.Format("{0:00.0}", way);
distanceSize.Text = " km";
}
else
{
amountDistance.Text = string.Format("{0:000}", way);
distanceSize.Text = " km";
}
```

Zapisanie danych do bazy danych:

```
public Task<int> SaveTrainingData(TrainingData trainingData)
{
return _database.InsertAsync(trainingData);
}

public Task<int> CountTrainings()
{
return _database.Table<TrainingData>().CountAsync();
}

public Task<int> SaveCurrentData(CurrentData currentPosition)
{
return _databaseTraining.InsertAsync(currentPosition);
}
```

4. Implementacja

5. Testowanie

6. Podręcznik użytkownika

Spis rysunków

Spis tabel

Spis listingów