|  |  |
| --- | --- |
|  | **Wydział Finansów i Zarządzania Kierunek: Informatyka** |

Marcin Mielniczuk

(numer albumu: 70671)

Traqer

Inżynierska praca projektowa

Opiekun merytoryczny:

Mgr inż. Marcin Pieleszek

Wrocław 2024

Spis treści

[Wstęp 3](#_Toc164276322)

[Założenia projektu 3](#_Toc164276323)

[Podobne rozwiązania dostępne na rynku 3](#_Toc164276324)

[Strava 4](#_Toc164276325)

[Endomondo 4](#_Toc164276326)

[Asics Runkeeper 4](#_Toc164276327)

[FitoTrack 4](#_Toc164276328)

[Honorable mentions 4](#_Toc164276329)

[Projektowanie rozwiązania 4](#_Toc164276330)

[Środowisko 4](#_Toc164276331)

[Przegląd języków programowania na platformę Android 5](#_Toc164276332)

[Java 5](#_Toc164276333)

[Kotlin 5](#_Toc164276334)

[C++ 6](#_Toc164276335)

[C# 6](#_Toc164276336)

[JavaScript + React Native 6](#_Toc164276337)

[C# i Xamarin 7](#_Toc164276338)

[Dart i Flutter 7](#_Toc164276339)

[Wybór frameworku 7](#_Toc164276340)

[Język programowania Dart 7](#_Toc164276341)

[Środowisko developerskie 8](#_Toc164276342)

[Przegląd UI 8](#_Toc164276343)

[Ekran główny i analiza tras 8](#_Toc164276344)

[Ekran treningu i mapa 12](#_Toc164276345)

[Powiadomienia 13](#_Toc164276346)

[Omówienie implementacji 14](#_Toc164276347)

[Model 14](#_Toc164276348)

[View 14](#_Toc164276349)

[Controller 14](#_Toc164276350)

[Komponenty 15](#_Toc164276351)

[Widok 15](#_Toc164276352)

[Kontrolery 18](#_Toc164276353)

[Modele 20](#_Toc164276354)

[Dekoratory 22](#_Toc164276355)

[Testy aplikacji 24](#_Toc164276356)

[Dalsze możliwości rozwoju 24](#_Toc164276357)

[Podsumowanie 24](#_Toc164276358)

# 

# Wstęp

Inspiracją do stworzenia tego projektu jest moje zainteresowanie sportem i chęć własnoręcznego stworzenia aplikacji umożliwiającej śledzenie postępu treningów w telefonie z systemem android.

# Założenia projektu

Celem projektu inżynierskiego jest stworzenie aplikacji na urządzenia mobilne, która pozwoli na śledzenie treningów kolarskich – zarówno śledzenie „na bieżąco” parametrów takich jak czas, prędkość, dystans i trasa oraz archiwizacja i możliwość analizy zakończonych aktywności.

# Podobne rozwiązania dostępne na rynku

Na rynku znajduje się kilka rozwiązań posiadających funkcje, które przedstawiłem w założeniach projektowych, poniżej przedstawiłem kilka wybranych aplikacji

## Strava

Aplikacja służąca do śledzenia treningów różnego rodzaju od biegania, po narciarstwo górskie. Strava działa w modelu subskskrypcyjnym, aby mieć dostęp do wszystkich funkcji aplikacji należy uiścić opłatę co miesiąć lub rok – w zależności od wybranego planu opłat. Aby używać aplikacji należy założyć konto.

## Endomondo

Map my run – aplikacja stworzona przez korporację Under Armour. Do używania programu wymagane jest rejestracja i utworzenie konta w serwisie operatora.

## FitoTrack

Pierwszy w tym porównaniu program typu open-source. Program pozwala rejestrować trening biegowe/rowerowe, posiada też wiele dodatkowych opcji takich jak – śledzenie kaloryczności posiłków, śledzenie ćwiczeń siłowych (?), manualne wprowadzanie ćwiczeń, ściągniecię i używanie map w trybie offline. FitoTrack chwali się większą prywatnością w porównianiu do podobnych programów, nie wymaga utworzenia konta, czy przekazywania swoich danych twórcom aplikacji. Aplikacja jest na licencji GNU/GPL v3, co oznacza, że można dowolnie modyfikować jej kod źródłowy na własny użytek.

<https://alternativeto.net/software/fitotrack/about/>

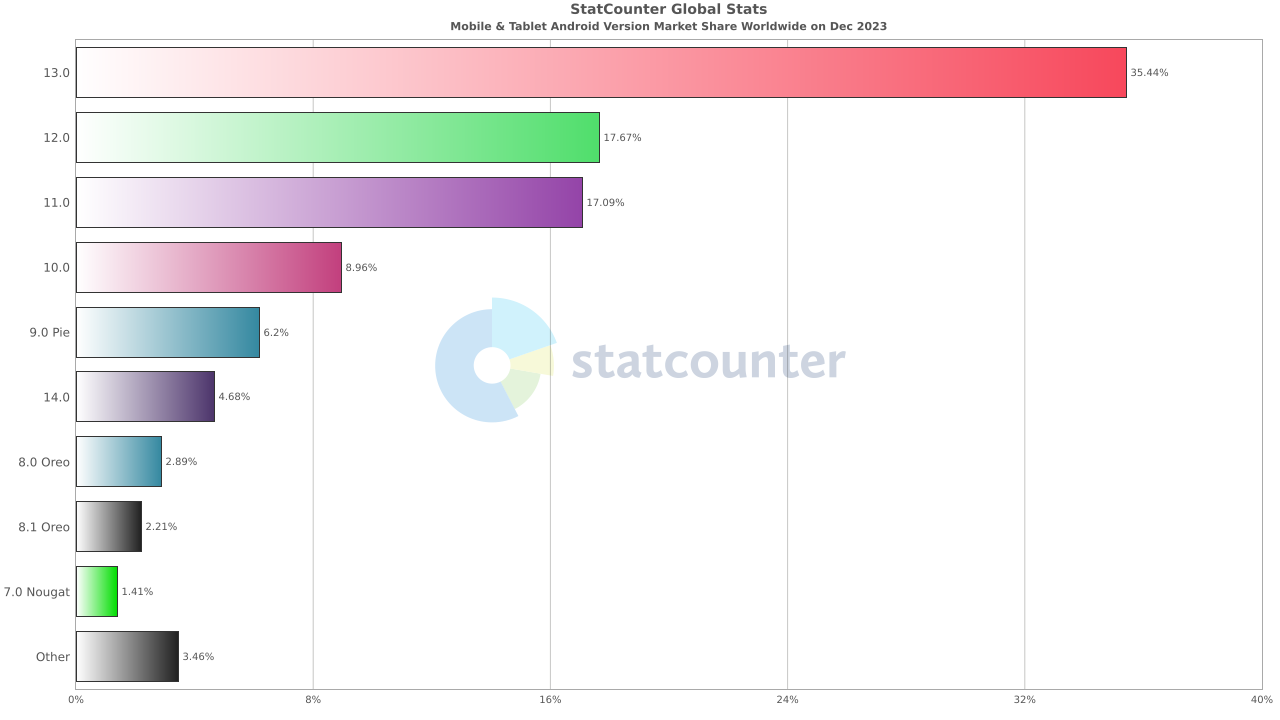
## Honorable mentions

Poza powyższymi aplikacjami wielu producentów sprzętu sportowego – zarówno elektroniki jak i odzieży posiada program do śledzenia treningów lansowany znakiem swojej marki np Adidas (adidas Running: Run Tracker), Nike (Nike Run Club – Running Coach), Garmin (Garmin Connect), Xiomi (Mi Fitness), Google (Google Fit: Activity Tracking), Fitbit (Fitbit).

# Projektowanie rozwiązania

## Środowisko

Platformą docelową aplikacji jest system Android. W czwartym kwartale 2023 roku, urządzenia z systemem Android stanowiły 70,11% całego rynku smartfonów (<https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/>).



Wykres 1

Z wykresu wynika, że 79% urządzeń z systemem Android to wersja 10 lub wyższa.

## Przegląd języków programowania na platformę Android

### ****Java****

Java jest bardzo popularnym językiem ogólnego przeznaczenia, którym posługuje się wg badania Stack Overflow z 2023 r. 30,55% deweloperów (<https://survey.stackoverflow.co/2023/#most-popular-technologies-language>).

Do zalet Javy należy możliwość przenoszenia aplikacji pomiędzy różnymi platformami, ponieważ do działania wymagają one jedynie wirtualnej maszyny Java (JVM). Składnia języka jest podobna do składni popularnych języków C++ i C#, co może niektórym programistom ułatwiać uczenie się. Ponadto obsługuje wielowątkowość i bardzo dobrze nadaje się do tworzenia aplikacji o złożonej architekturze.

Za wady Javy uznaje się konieczność używania przez developerów stosunkowo dużych ilości kodu, nawet do wykonywania prostych zadań.

### ****Kotlin****

Kotlin jest młodszą alternatywą dla Java, której popularność wciąż rośnie. Obecnie ma status oficjalnego języka platformy Android. Charakteryzuje się zwięzłą strukturą kodu, co daje mu przewagę nad poprzednikiem. Programowanie w nim jest szybsze i bardziej efektywne.

Kod stworzony przy użyciu Kotlin można skompilować do kodu JavaScript oraz uruchamiać go za pomocą wirtualnej maszyny Java (JVM). Język cechuje szybki proces kompilacji, ma też zwięzły i wygodny w użyciu zestaw bibliotek. Dla programistów istotne jest, że Kotlina łatwo jest się nauczyć.

Do wad języka można zaliczyć większą trudność przy debugowaniu (diagnozowaniu i usuwaniu błędów kodu). Podnoszona jest także czasem kwestia pewnych braków w dokumentacji oraz tego, że jest to język nowy, przez co nieodpowiedni do programowania aplikacji na starsze urządzenia. Wciąż ewoluuje i developerzy muszą być na bieżąco ze zmianami, które w nim zachodzą.

Zarówno używając Java, jak i Kotlin developerzy mogą korzystać z narzędzia Google do tworzenia aplikacji - Android Studio. Jest to ich dużą zaletą.

### ****C++****

C++ to język obiektowy, którego używa około 22,42% programistów. Mimo jego skomplikowania developerzy chętnie inwestują swój czas w naukę. Wynika to z faktu, że można go wykorzystać także do innych zastosowań niż pisanie aplikacji mobilnych - tworzenia gier, aplikacji desktopowych czy systemów uczenia maszynowego.

C++ jest uniwersalny, gdyż może być wykorzystywany zarówno do budowania aplikacji dla Androida, jak i iOS. Jego atut to również szybkość, ponieważ zużywa małą ilość pamięci RAM (doceniane jest to np. przez twórców gier 3D).

Wadą języka jest jednak mała elastyczność oraz większe ryzyko występowania błędów w kodzie.

### ****C#****

Język ten jest następcą C i C++, który rozwiązuje jeden z ich najważniejszych problemów - nadmierny stopień skomplikowania. Jest znacznie łatwiejszy do nauczenia, dzięki prostocie struktury i klarowności kodu.

C# został stworzony przez Microsoft, który udostępnia developerom użyteczne narzędzia. Należy do nich środowisko IDE Visual Studio będące częścią ekosystemu .NET, które zawiera wszystkie niezbędne funkcje pozwalające zbudować aplikację. Nie ma potrzeby pobierania dodatkowych narzędzi. Wadą C# może być jednak to, że jest zależny od platformy .NET. Zmniejsza to jego elastyczność, gdy nie planujemy oprzeć stosu technologicznego na tej technologii.

### JavaScript + React Native

JavaScript jest najpopularniejszym językiem programowania na świecie, który zna 65% developerów. Daje to pracodawcom dostęp do olbrzymiej ilości specjalistów, co przekłada się na łatwość i szybkość tworzenia zespołów.

Język ma wiele zastosowań - jest powszechnie używany przy tworzeniu stron internetowych, ale można go także wykorzystać do budowy oprogramowania po stronie serwera (dzięki rozwiązaniu Node.js). Przy jego użyciu powstaje także bardzo wiele aplikacji mobilnych - hybrydowych i cross-platforms.

Jednym z najpopularniejszych rozwiązań jest korzystanie z JavaScript i ReactNative. React Native jest frameworkiem stworzonym przez Facebooka, który umożliwia tworzenie wydajnych aplikacji mobilnych. Kod JavaScript można przeplatać w miarę potrzeby kodem natywnym, aby wykorzystywać funkcje urządzeń.

Do minusów React Native należy utrudnione debugowanie, ponieważ rozwiązanie nie zawsze precyzyjnie wskazuje źródła i lokalizacje błędów.

React Native to także framework, przy którym praca jest dosyć czasochłonna. Rozpoczęcie projektu trwa stosunkowo długo, a testowanie aplikacji wymaga korzystania z fizycznych urządzeń, gdyż emulatory i symulatory nie dają pewności co do jej działania aplikacji w wybranych systemach. Jeżeli dodamy do tego niedostateczną personalizację aplikacji pod kątem Androida czy iOS uzyskujemy rozwiązanie odpowiednie raczej dla mniej wymagających projektów.

### C# i Xamarin

Język C#, jak wspomnieliśmy, stosowany jest także do tworzenia aplikacji natywnych. Bazuje na nim jednak również platforma Xamarin, która umożliwia kompleksowe tworzenie aplikacji cross-platform. Rozwiązanie daje developerom dostęp do narzędzia Visual Studio oraz sklepu Xamarin Component Store, gdzie można znaleźć wiele użytecznych komponentów.

### Dart i Flutter

Flutter to platforma open source autorstwa Google. Opiera się na stworzonym w 2011 języku Dart. Składa ze zbioru narzędzi SDK (Software Development Kit) służących do rozwoju aplikacji, między innymi do kompilowania kodu oraz z frameworka. W ramach frameworka użytkownik ma do dyspozycji konfigurowalne widgety będące elementami wielokrotnego użytku (to np. przyciski, suwaki, pola do wprowadzania tekstu).

Platforma pozwala tworzyć wydajne i estetyczne aplikacje przy użyciu stosunkowo niewielkiej ilości kodu. Jest ceniona za dobrą dokumentację i łatwość użytkowania. Społeczność Fluttera dynamicznie się rozwija i wciąż pojawiają się nowe biblioteki, które są publikowane wraz z kodem źródłowym co daje możliwość ich modyfikacji oraz kontrybucji.

## Wybór frameworku

Do implementacji projektu wybrałem framework Flutter. Został on stworzony przez firmę Google. Jest to uniwersalny framework, który pozwala na tworzenie aplikacji na kilka platform. Ten sam kod może zostać zbudowany na kilka platform naraz

### Język programowania Dart

Jest to stosunkowo młody język, który powstał w 2011 roku i nie znajduje on użycia poza framework’iem Flutter. Jednak dzięki możliwości kompilacji na wiele platform posiada możliwość posiadania jednej bazy kodu aplikacji dla wszystkich wspieranych platform.

Dużą zaletą Dart’a jest duża ilość widgetów dostępnych w standardowych bibliotekach. Pozwala to na skupienie się na logice tworzonych aplikacji, poświęcając mniej czasu na UI/UX. Dart wyróżnia się także łatwością z jaką można modyfikować istniejące widgety, tworząc własne wariacje na podstawie tych dostępnych w standardowych bibliotekach.

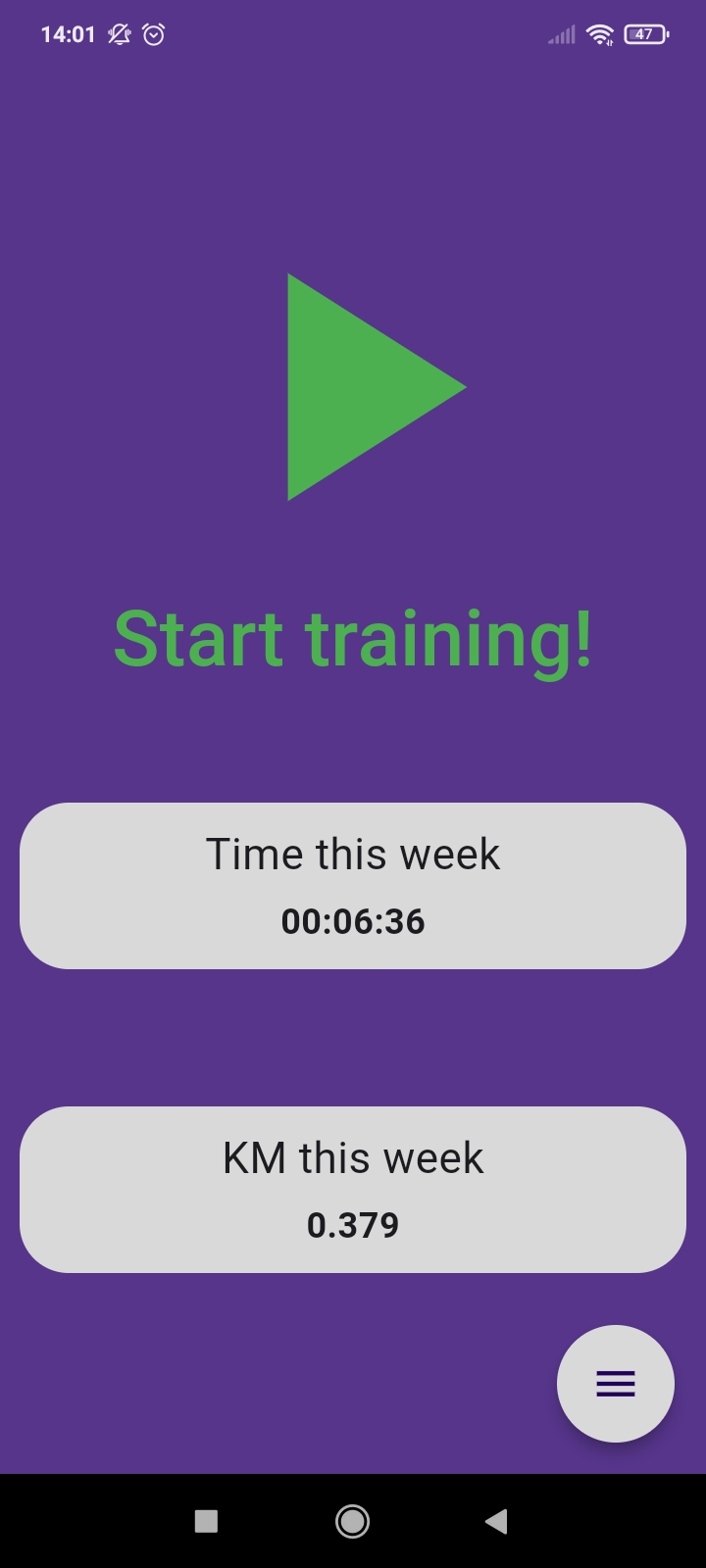
Poza standardowymi bibliotekami istnieje też dużo bibliotek na licencji open-source (najczęściej MIT lub BSD) tworzonych przez użytkowników. Dzięki menedżerowi bibliotek flutter pub łatwo jest je importować do własnych projektów i używać oszczędzając dzięki temu dużo czasu.

Dokumentacja bibliotek jest zazwyczaj szczegółowa oraz obfita w ilość przykładów. Portal pub.dev zbiera udostępnione przez użytkowników biblioteki w jednym miejscu wraz linkami do dokumentacji oraz changelog’ami i

## Środowisko developerskie

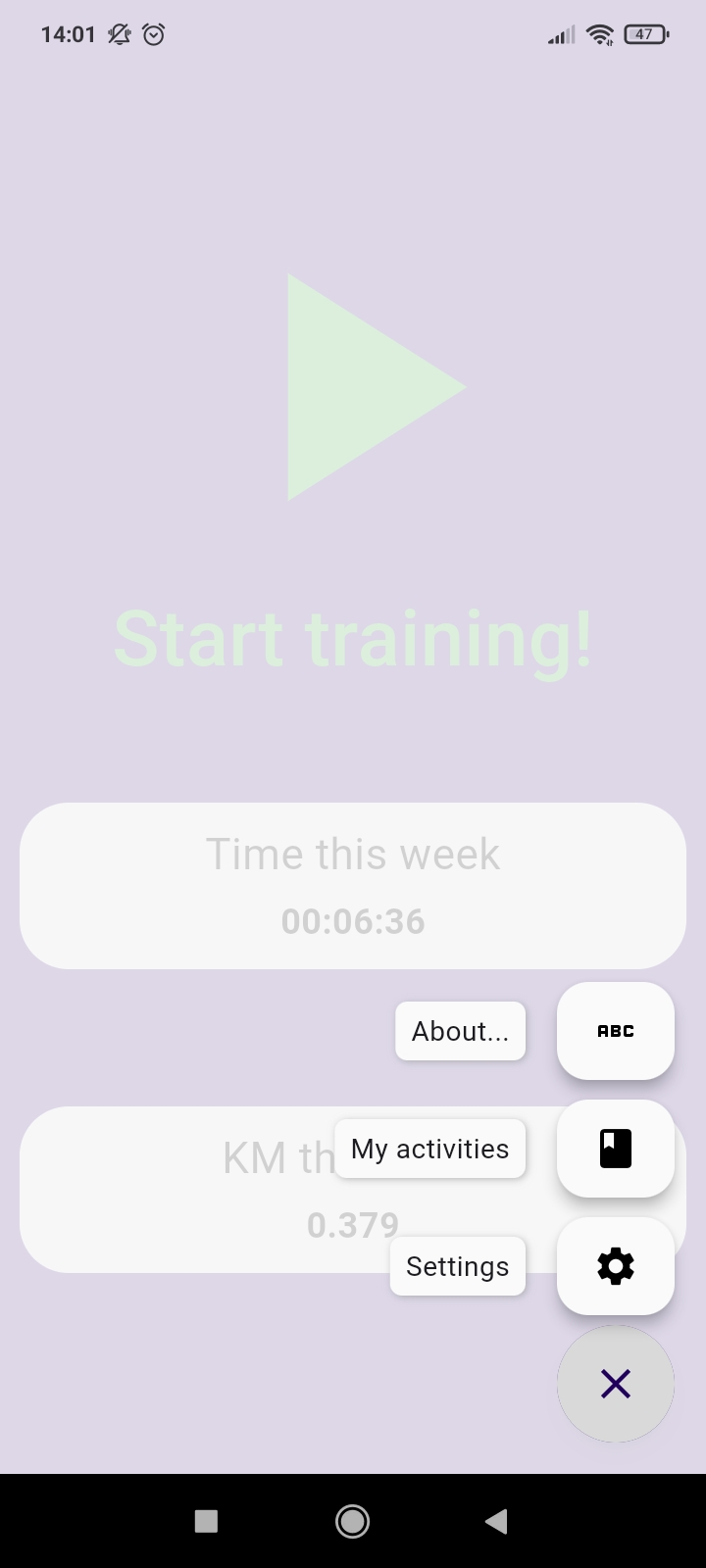
Do napisania programu użyłem programu Android Studio, wraz z emulatorem telefonu z systemem android. Zintegrowane środowisko oraz możliwość restartu programu „na gorąco” pozwalało na szybkie wprowadzanie zmian do programu i ich testowanie w przeciągu kilku sekund. Znacznie przyspieszyło to pracę nad układem graficznym aplikacji.

# Przegląd UI

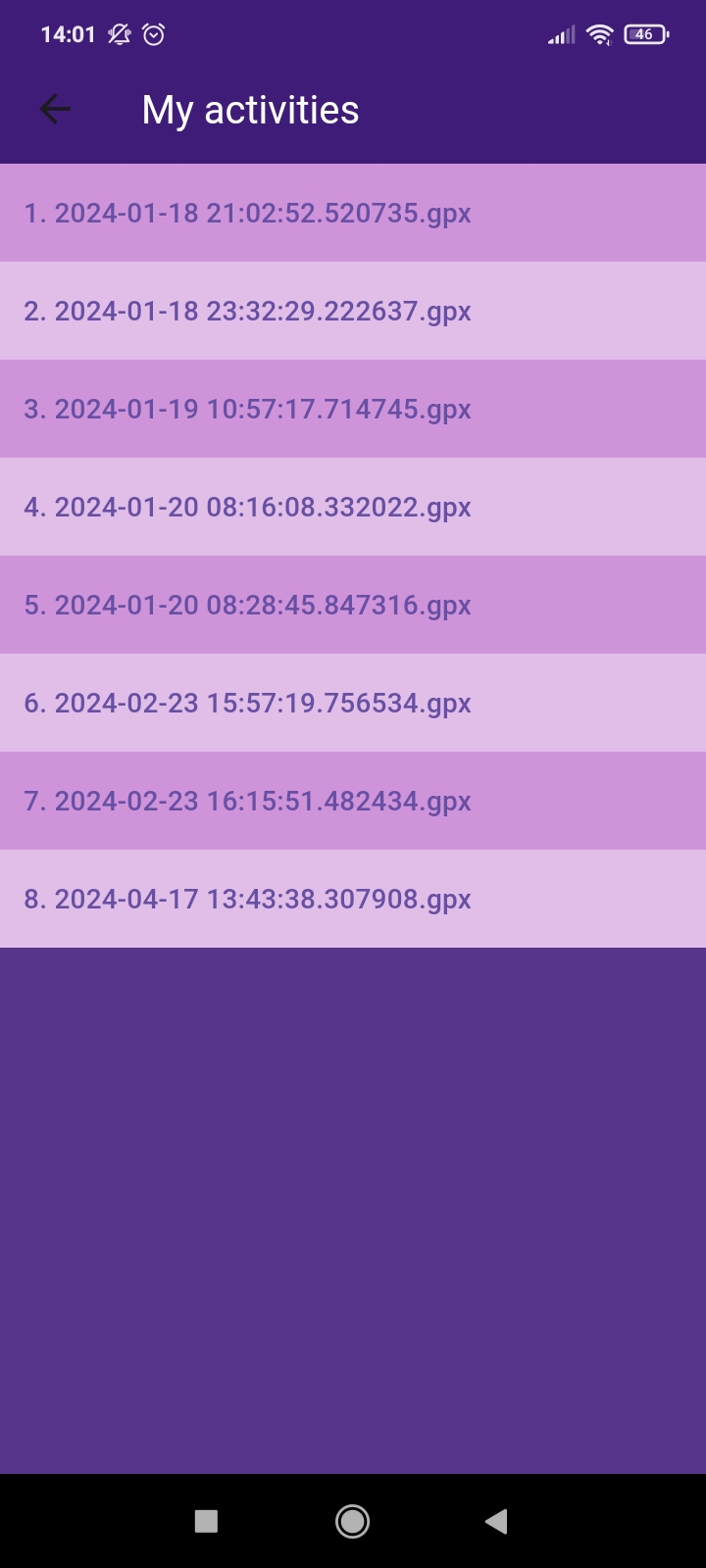


## Ekran główny i analiza tras

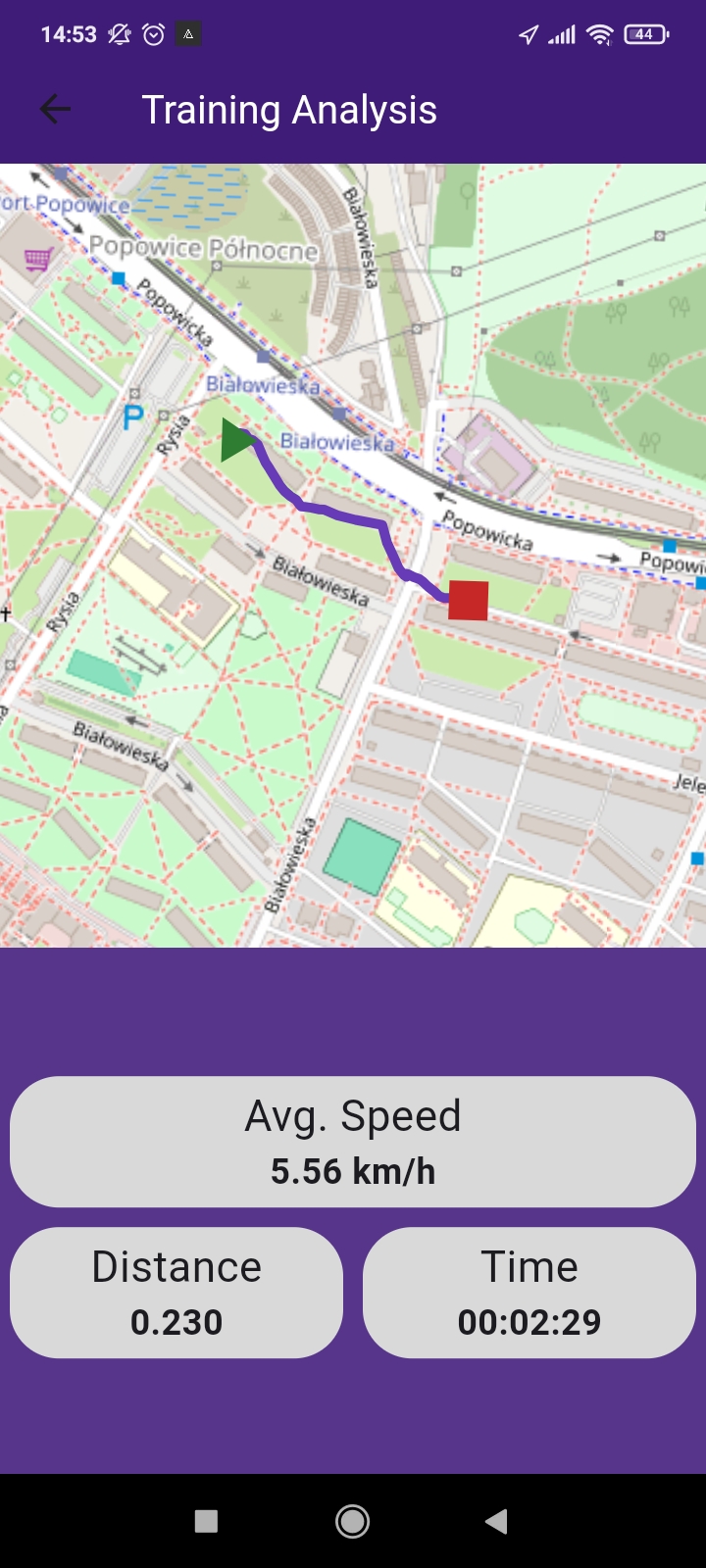
Na ekranie powitalnym użytkownik może zobaczyć podsumowanie treningów z ostatnich 7 dni. Znajduje się tutaj także możliwość przejścia do nowego treningu (zielony przycisk) oraz przycisk menu w prawym dolnym rogu ekranu.



Po otworzeniu menu użytkownik ma opcję wyświetlenia informacji o aplikacji, przeglądu minionych treningów oraz otwarcie ekranu opcji.

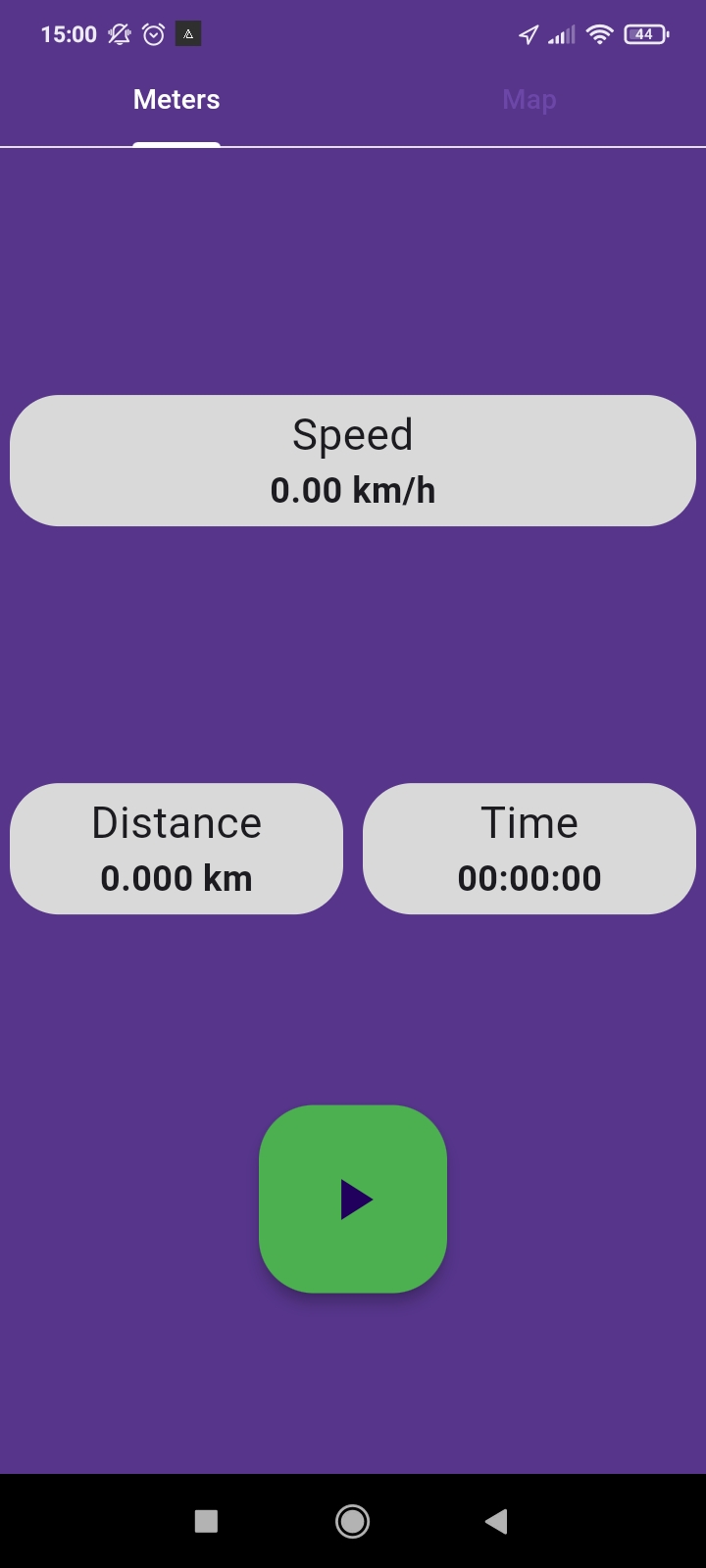


Po nasiśnięciu opcji „My activities” aplikacja wyświetli listę aktywności, a po naciśnięciu na dowolny przedmiot z listy otworzy widok szczegółowy danej aktywności.

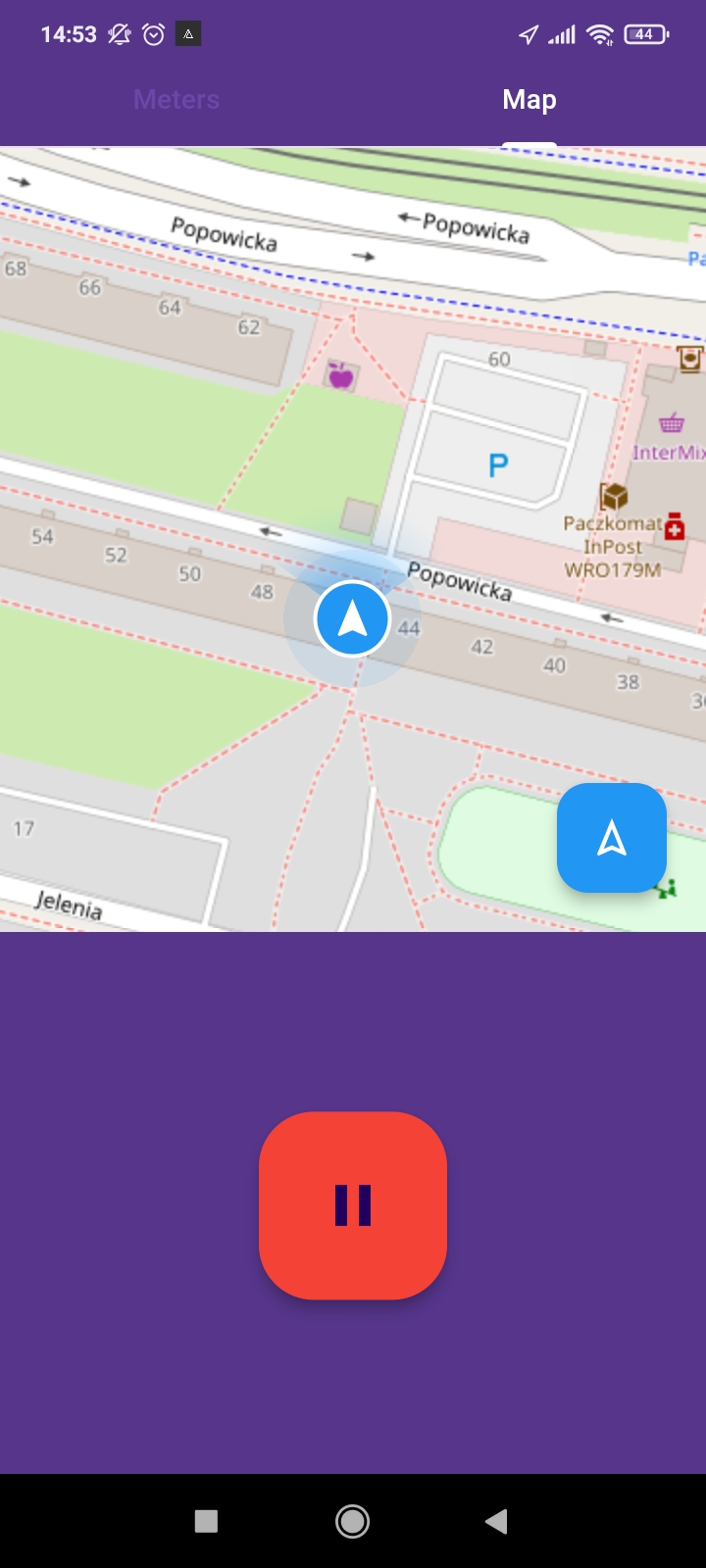


W szczegółowym widoku można znaleźć interaktywną mapę z zaznaczoną trasą, przebyty dystans, czas przemieszczania oraz średnią prędkość.

## Ekran treningu i mapa



Po kliknięciu przycisku „Start training!” znajdującego się na ekranie głównym aplikacja przeniesie użytkownika na ekran treningu, znajdują się na nim dwie zakładki. Pierwsza z nich zawiera informację o aktualnej prędkości urządzenia, dystans oraz czas przemieszczania. Pomiar odległości i czasu zaczyna się w momencie wciśnięcia zielonego przycisku, wtedy zmienia on kolor na czerwony, a trasa jest zapisywana.



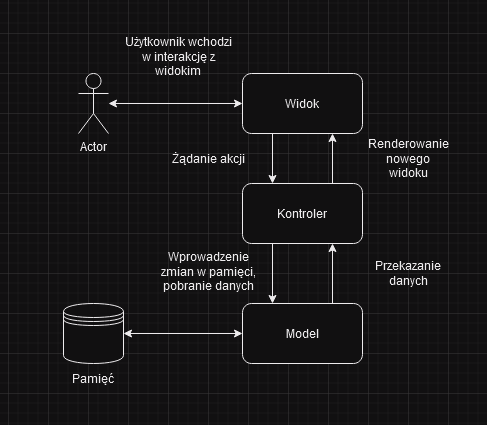
W drugiej zakładce ekranu treningu znajduje się interaktywna mapa z pozycją użytkownika, jest ona wyświetlana w czasie rzeczywstym. Mapa może posłużyć do nawigacji w terenie, posiada opcję przeglądania mapy lub może być przełączona w tryb nawigacji – obraz będzie podążać za pozycją użytkownika.

## Powiadomienia

Aplikacja podczas swojego działania wyświetla powiadomieni o tym, że śledzenie jest aktywne (ikona aplikacji widoczna na górnym pasku).

# Omówienie implementacji

Aplikacja została stworzona w architekturze MVC (Model-View-Controller), oznacza to, że jej komponenty zostały podzielone na klasy i każda klasa spełnia określony zestaw zadań. Schemat działania architektury MVC jest ukazany na poniższym Obraz 1.



Obraz 1

## Model

Jest to zbiór komponentów w które odpowiadają za dane znajdujące się w aplikacji (zapisy tras, ilość przebytych kilometrów itp.). Na zmianę danych znajdujących się w komponentach tej kategorii wpływają kontrolery.

## View

W tej kategorii znajdują się elementy, które są widoczne dla użytkownika i można z nimi wchodzić w interakcję na ekranie telefonu. Te interakcje są później przekazywane do kontrolerów, które są w stanie zmieniać dane znajdujące się w modelu

## Controller

Jak już wspomniałem wyżej kontroler odpowiada za odbieranie żądań od widoku, sprawdza, czy wprowadzone dane są poprawne i przekazuje żądanie do modelu, potem przekazuje informację zwrotną, czy zostało wykonane poprawnie lub zwraca dane do widoku.

## Komponenty

### Widok

Interfejs użytkownika jest zrealizowany za pomocą Widgetów, które są integralną częścią języka Dart, aplikacja zawiera w sobie zarówno widgety statyczne (class StatelessWidget) oraz dynamiczne (class StatefulWidget), czyli takie, które posiadają stany i po zmianie stanu renderują nowy widok. Widgetem mogą być poszczególne strony w aplikacji, ale także elementy znajdujące się na stronach, czyli widgety mogą zawierać w sobie inne widgety. Dużo elementów dostępnych w standardowych bibliotekach języka Dart dziedziczy od klasy StatefulWidget (guziki, checkboxy itp) albo StatelessWidget (kontenery, rozdzielacze, tekst). <https://api.flutter.dev/flutter/widgets/widgets-library.html>



W powyższym kodzie przedstawiona jest definicja dynamicznego widoku. Przyjmuje on jeden argument (path). Jednym z pól klasy jest stan widgetu. Logika dotycząca stanów jest zdefiniowana w klasie \_TrainingAnalysisState znajdują się tam instrukcje kiedy i w jaki sposób widget ma zrenderować się ponownie.

Klasa stanu widgeta posiada także własne pola, ale z tej klasy można także odwoływać się do pól klasy StatefullWidget. Widok jest renderowany kiedy wykonuje się metoda initState, czyli wtedy, gdy widok pojawia się na ekranie lub jedna z wartości zostaje zmieniona przy użyciu metody setState() wykonuje się wtedy metoda build, w niej opisana jest logika rozłożenia elementów oraz zmienne wyświetlane na ekranie.

Powyższy przykład funkcji build() pokazuje zdefiniowane elementy, które mają być wyświetlone na ekranie. Wyświetla on listę plików znajdujących się w lokalizacji gdzie zapisywane są ścieżki pliki gpx zarejestrowane przez program.

### Kontrolery

W programie znajduje się kontroler, który odpowiada za sterowanie przebiegiem zapisywania śladu, który można zainicjować z ekranu treningu. Odpowiada on za czytanie pozycji urządzenia i przekazywanie danych do modelu oraz widgetów. 

Powyższy kontroler udostępnia innym obiektom strumień danych, te odpowiednio przekazują je dalej, na ekran lub do modelu. Kontroler wywołuje także funkcję rozpoczęcia zapisu i zapis danych do pliku.

### Modele

Ścieżki utworzone podczas treningów są zapisywane w formacie gpx (opis formatu dotępny na stronie <https://www.topografix.com/gpx.asp>). Format gpx jest rozszerzeniem formatu XML, każdy ślad gpx składa się z par współrzędnych – długość i szerokość geograficzna. Dodatkowo znajduje się tam informacja o czasie, ale nie jest wymagana aby ślad był poprawny. Istnieje także możliwość zdefiniowania własnych rozszerzeń, które można dopisać do punktów np. oznaczenie ciekawego miejsca itp. Pary współrzędnych mogą być także zorganizowane w ścieżki (tracks) oraz segmenty (track segments), umożliwia to zapisywanie kilku śladów w jednym pliku formatu gpx, ale nie jest to konieczne do wszystkich zastosowań tego formatu plików.

Za operacje zapisu i odczytu plików odpowiadają klasy TrackWriter i TrackReader.

Klasa TrackReader przegląda folder aplikacji i zwraca listę plików gpx. Odpowiada także za wczytywanie tras do pamięci co pozwala na wyświetlanie tras i obliczenia szczegółów takich jak czas trasy, dystans i średnią prędkość. Klasa TrackWriter odpowiada za odwrotną operację, zapis tras z pamięci do pliku.

### Dekoratory

W klasie TrackDecorator znajdują się komponenty, które obliczają czas i dystans.

Czas całkowity obliczany jest na podstawie znacznika z pierwszego i ostatniego punktu na trasie.



Dystans obliczany przy użyciu funkcji haversine – jest to formuła pozwalająca na obliczenie odległości między dwoma punktami na sferze, gdy danymi jest kąt między tymi punktami. Taka operacja jest wykonywana dla wszystkich sąsiadujących ze sobą punktów ścieżki, suma daje dystans całkowity. (Heavenly mathematics ; the forgotten art of spherical trigonometry / Van Brummelen, Glen / 2013)

Promień Ziemi różni się między w zależności od tego, czy zostanie poprowadzony do bieguna (6356.752 km), czy do równika (6378.137 km), więc ta metoda daje dokładność ok 0,5%, która wystarczy do zastosowania w aplikacji, lecz przy budowaniu rozwiązań z małą tolerancją błędu należałoby wziąć tą niedokładność pod uwagę.



Zaimplementowane tutaj także zostały metody służące obliczaniu podsumować z ostatnich 7 dni, które są wyświetlane na ekraniu głównym po włączeniu aplikacji.

# Podsumowanie

Aplikacja po skompilowaniu zajmuje 22.9MB na urządzeniu docelowym. Szybko reaguje na akcje użytkownika. Pokazuje to, że wybór języka Dart był dobry. W trakcie sporządzania pracy udało mi się pogłębić swoją wiedzę na temat tworzenia aplikacji na urządzenia mobilne oraz stosunkowo nowego języka jakim jest Dart.