# Instrukcja obsługi klas z przyrostu SPI-4

Autor: Mikołaj Krzymiński

Data: 13.04.2021

Wersja: 1.0

1. Struktura projektu
   1. Drzewo struktury projektu

lib  
|---managers

| |------------db\_manager.dart

| |------------run\_manager.dart

|---models

| |---------geolocation\_model.dart

| |---------run\_model.dart

|---pages

| |-------training\_page.dart

|---main.dart

* 1. Tworzenie kodu źródłowego
     1. Kod źródłowy tworzymy w katalogu lib, zgodnie ze strukturą Fluttera i konwencją Dart: <https://dart.dev/guides/language/effective-dart/style>.

1. Odpowiedzialności stworzonych klas
   1. Katalog Managers
      1. db\_manager
         1. Jest klasą utworzoną przy pomocy wzorca singleton. Aby korzystać ze wszystkich możliwość tej klasy wystarczy utworzyć obiekt typu DbManager. Konstruktor klasy woła skrypt łączący się z lokalną, sqlite-ową bazą danych. Dodatkowo, na potrzeby testów konstruktor przy pomocy prywatnej metody \_delete usuwa (jeśli istnieje) poprzednio utworzoną bazę danych.
         2. Przykład utworzenia obietku DbManager w pliku training\_page.dart w metodzie initState.
         3. Klasa udostępnia na zewnątrz metody addGeolocation i addRun, za ich pomocą dodajemy do bazy kolejne wpisy.
         4. Metody \_inserGeolocation i \_insertRun wołane w metodach addGeolocation i addRun, umieszczają wpisy w bazie danych.
         5. Metoda \_getGeolocations, na razie służy do testów i pozwala odczytać wszystkie zapisane geolocatory.
      2. run\_manager
         1. Zawiera dwie klasy
         2. RunManager, zbudowany na podstawie wzorca singleton. Jest odpowiedzialny za rejestrację wszystkiego co związane z obecnym biegiem użytkownika
            1. Pole bool-owe \_isRunning zawiera informacje o tym czy użytkownik w danej chwili jest w trakcie biegu.
            2. Publiczna metoda changeState, wołana w przypadku zdarzenia rozpoczęcia i końca rejestrowania geolokalizacji (przycisk STTART/STOP z training\_page). Na podstawie stanu z prywatnego atrybutu \_isRunning, wywołuje metodę \_startRun, albo \_endRun.
            3. \_startRun dodaje nowy bieg do bazy danych i uruchamia metodę \_startStream
            4. \_endRun zamyka stream i wypisuje wszystkie geolocatory z wszystkich biegów z bazy danych na konsolę (cele testowe).
            5. Metoda \_startStream zawiera w sobie informacje na temat ustawień pobierania informacji z GPS i otwiera strumień pobierający dane z GPS-a użytkownika. Każdy nowy geolocator trafia od razu do bazy danych z odpowiednim runId. Obserwator jest informowany o każdym nowym geolocatorze pobranym ze streama.
         3. RunObserver odpowiedzialny za obserwowanie zmian (nowych geolocatorów) w obiekcie RunManager.
            1. Stworzony po to aby oddzielić logikę biznesową, rejestrację treningu od widoku i rysowania mapy.
            2. Implementowany jest przez trainingPage.
   2. Models
      1. geolocation\_model
         1. Posiada statyczne atrybuty z nazwami kolumn w bazie danych. Chodzi o to, żeby nazwy kolumn z tabeli były zebrane w jednym miejscu, jako stałe.
         2. Posiada nazwę tabeli, do której ma być wpisany.
         3. Posiada skrypt tworzący tabelę, do której jest wpisywany.
         4. Jego atrybuty, są równoważne ze wszystkimi atrybutami, które pozyskujemy z obiektu Position (geolocatora) pozyskanego każdorazowo z GPS-a za pomocą streama i dodatkowy atrybut \_id, czyli id biegu.
         5. Model zawiera metody fromMap, który z bazodanowej krotki tworzy obiekt GeolocationModel.
         6. Metoda toMap zwraca Mapę, którą można zapisać w bazie danych.
         7. Metoda toString tworzy tekst, który jest reprezentacją obiektu.
      2. run\_model
         1. Podobnie jak w geolocation\_model, poza metodą toString
   3. Pages (tu będziemy trzymać wszystkie widoki stron, np. settings\_page, my\_account\_page, strength\_page itp..)
      1. training\_page
         1. Implementuje RunObserver, by wiedzieć o zmianach w obiekcie RunManager.
         2. Jest Flutterowym StatefulWidgetem.
         3. Tworzy obiekt RunManager, któremu przekazuje w parametrze siebie jako obserwatora.
         4. Jej klasa stanu za pomocą zintegrowanego API FlutterMap potrafi narysować mapę i trasę z podawanych przez obiekt RunManager pozycji użytkownika.
         5. Ma przycisk START/STOP, którym rozpoczyna i kończy bieg.