Wykonawca: Maciej Szefliński - 145294	**Wydział:** Informatyki i telekomunikacji	**Semestr:** 6	**Grupa:** L5
Przedmiot: Aplikacje Mobilne	**Data oddania:** 16.09.2022	**Temat:** Sprawozdanie z projektu - laboratorium 8 - 12	

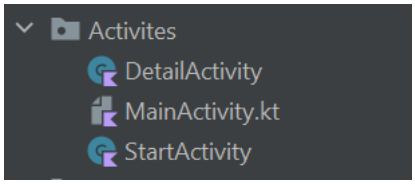
Opis projektu

W ramach projektu została wykonana aplikacja mobilna dla rowerzystów w języku Kotlin. W aplikacji można wybrać poziom trudności trasy rowerowej (łatwa oraz trudna) oraz zobaczyć jej szczegóły . W aktywności szczegółów, można skorzystać z wbudowanego stopera do mierzenia czasu oraz do późniejszych statystyk. Aplikacja jest w pełni responsywna.

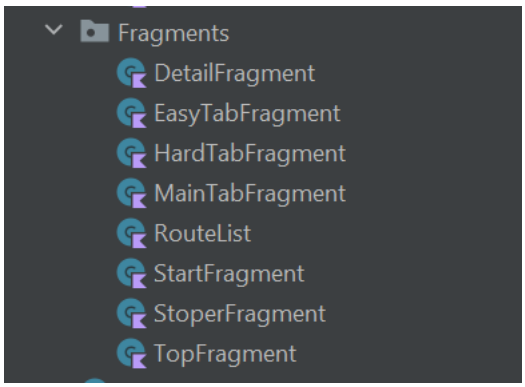
Przedstawienie wykonanych zadań

Kod został napisany w Kotlinie.

Aplikacja dzieli się na trzy aktywności



Każda z aktywności korzysta z fragmentów

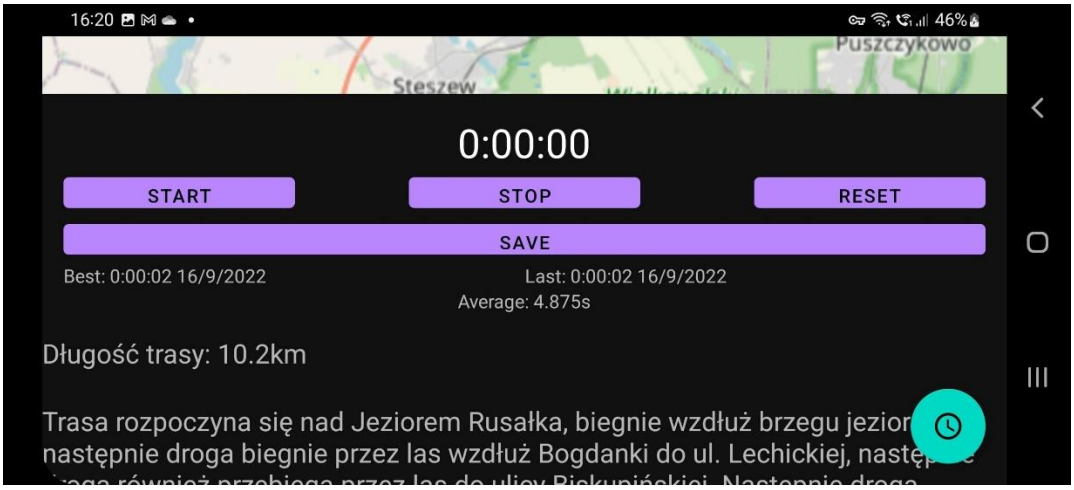


Przykładowa implementacja fragmentu w pliku .xml

```xml
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="com.example.lab8_12.Fragments.RouteList"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="0.9"/>
```

Responsywność aplikacji

Aplikacja działa poprawnie po zmianie orientacji urządzenia, jest w pełni responsywna. W layoutach nie ustawiam komponentów pikselami.



Źródłem danych jest lokalna baza danych SQLite

Tabela „route” przechowuje informacje o trasach.

Tabela „timer” przechowuje zapisane wyniki pokonanych tras.

```kotlin
class Database(context: Context) : SQLiteOpenHelper(context, DATABASE_NAME, factory: null,
    DATABASE_VERSION) {

    companion object{
        private const val DATABASE_NAME = "route.db"
        private const val DATABASE_VERSION = 5

        private const val TABLE_ROUTE = "route"
        private const val COLUMN_ID = "id"
        private const val COLUMN_ROUTE_TITLE = "title"
        private const val COLUMN_ROUTE_DESCRIPTION = "description"
        private const val COLUMN_ROUTE_LENGTH = "length"

        private const val TABLE_TIMER = "timer"
        private const val COLUMN_TIMER_ROUTE_ID = "routeId"
        private const val COLUMN_TIMER_SECONDS = "seconds"
        private const val COLUMN_TIMER_DATE = "date"
```

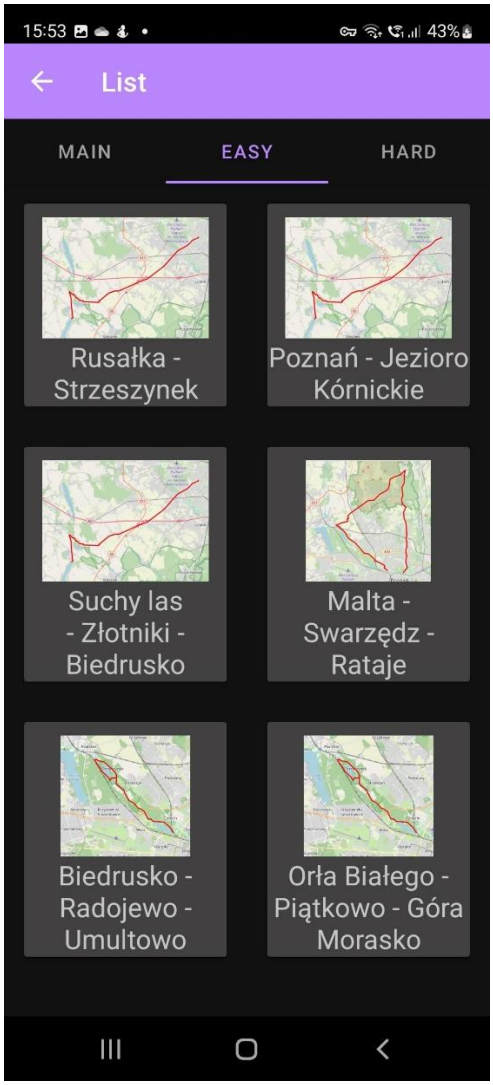
Stoper

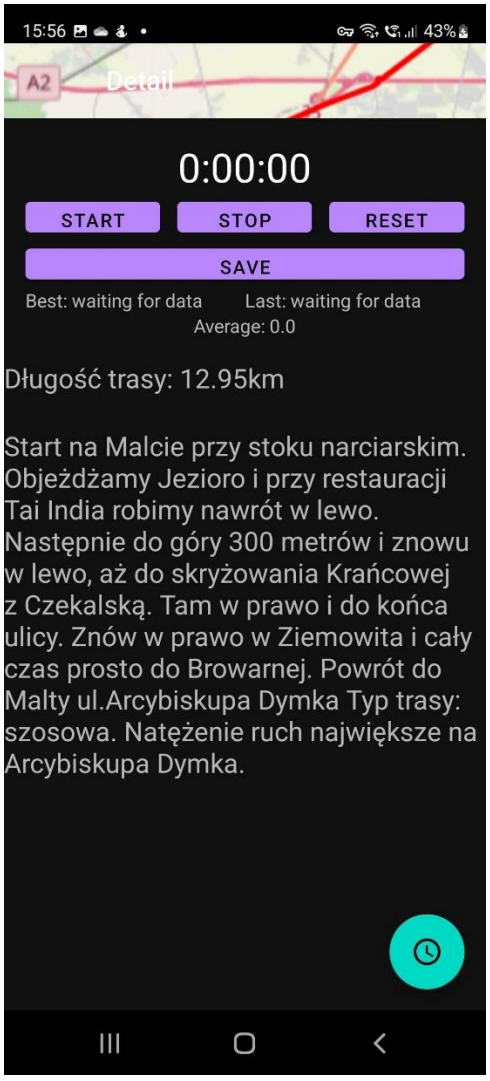
Stoper wyświetla czas z dokładnością do sekundy. Możliwe jest wystartowanie stopera, zatrzymanie, wznowienie, zresetowanie oraz zapisanie rekordu. Pod stoperem można odczytać najlepszy wynik, ostatni wynik oraz średnia pomiarów podana w sekundach.

Karty kategorii

Karty kategorii używają widoku RecyclerView z układem grid, w którym poszczególne pozycje są prezentowane w postaci obrazka i nazwy, kliknięcie wybranej trasy powoduje wyświetlenie większego obrazka, stopera wraz ze statystykami, opisu oraz długości trasy.

```
class RouteListRecyclerAdapter(private val dataSet: MutableList<Route>) : RecyclerView.Adapter<RouteListRe

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        val v = LayoutInflater.from(parent.context).inflate(com.example.lab8_12.R.layout.route_list_layout
        return ViewHolder(v)
    }


    override fun getItemCount() = dataSet.size

    inner class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView){
        var routeListText: TextView
        var touch: LinearLayout
        var image: ImageView

        init {
            routeListText = itemView.findViewById(R.id.routeListText)
            image = itemView.findViewById(R.id.image)
```

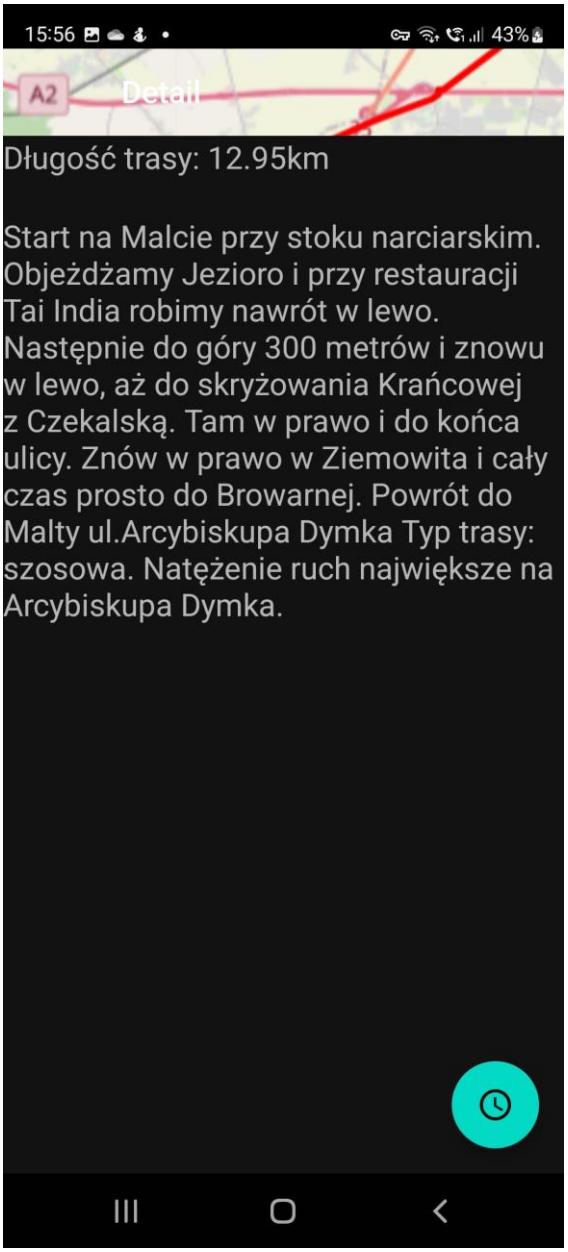


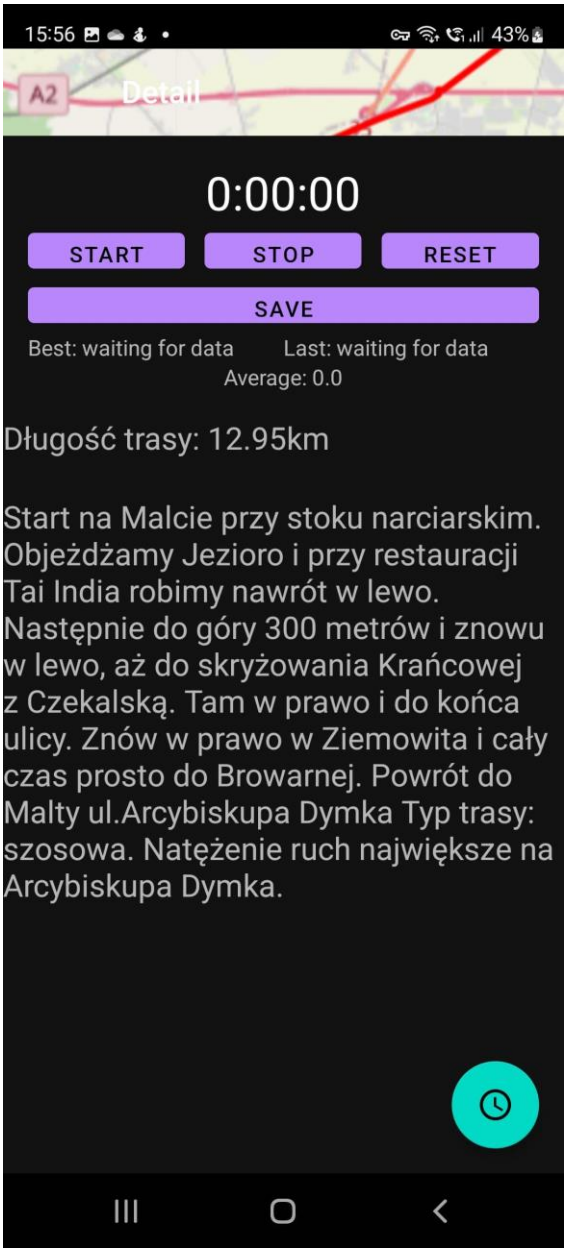



```xml
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Fragments.EasyTabFragment">

    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/easy" />

    <ListView
        android:id="@android:id/list"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:drawSelectorOnTop="false"/>

</FrameLayout>
```

Przycisk FAB

Na ekranie szczegółów znajduje się przycisk FAB, wykorzystywany do chowania oraz pokazywania stopera razem ze statystykami.





Motywy

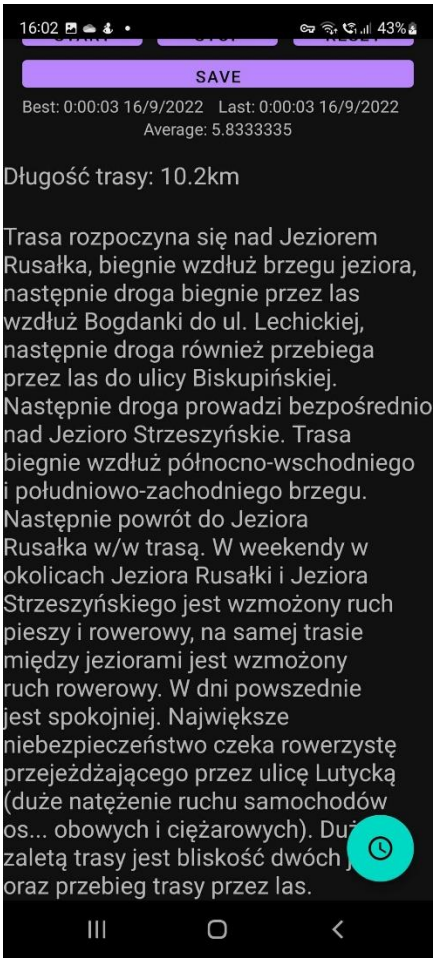
W aplikacji zastosowano motywy korzystając z biblioteki wzornictwa.

```
<activity
    android:name=".Activites.StartActivity"
    android:exported="true"
    android:theme="@style/Theme.Lab812.NoActionBar"
    android:label="Cycling">
```

```
<style name="Theme.Lab812.NoActionBar" parent="Theme.MaterialComponents.DayNight.NoActionBar">
    <item name = "android:windowActionBar">false</item>
    <item name = "android:windowNoTitle">true</item>
</style>
```

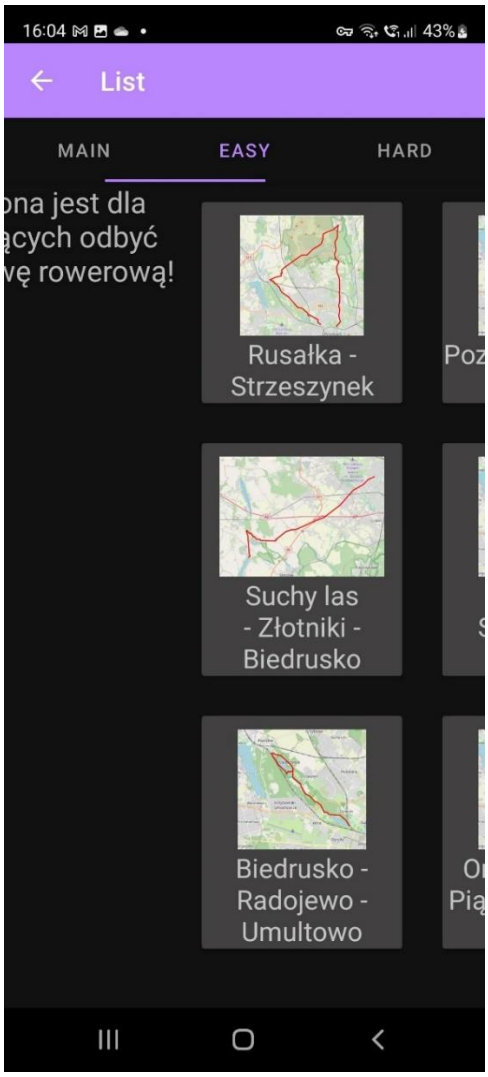
Przewijanie ekranu szczegółów

Ekran szczegółów jest przewijany w pionie razem z paskiem aplikacji. Obrazek pojawia się na pasku aplikacji oraz zwija się razem z nim.



Gest przeciągnięcia

Przechodzenie pomiędzy kartami odbywa się także za pomocą gestu przeciągnięcia.

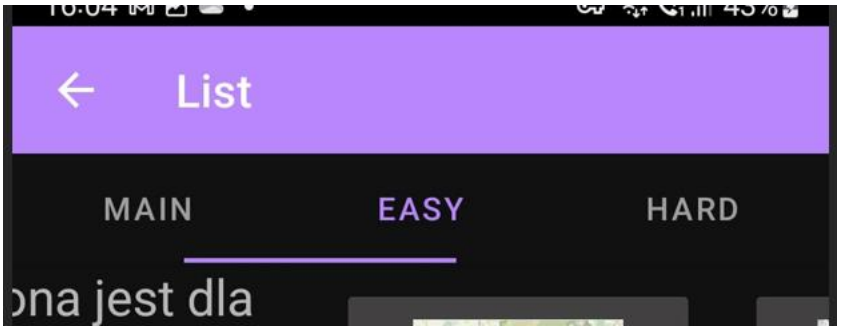



```
val pagerAdapter = SectionsPagerAdapter(supportFragmentManager)
val pager = findViewById<View>(R.id.pager) as ViewPager
pager.adapter = pagerAdapter
val tabLayout = findViewById<View>(R.id.tabs) as TabLayout
tabLayout.setupWithViewPager(pager)
```

```
<ListView
    android:id="@android:id/list"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:drawSelectorOnTop="false"/>
```

Pasek narzędzi

Dodanie do paska aplikacji akcji, a konkretnie akcji powrotu do poprzedniej aktywności. Nie można wrócić do aktywności startowej, ponieważ ta jest niszczona po zakończonej animacji.



Wersja dla tabletów

Poczynione kroki:

Sprawdzanie i przypisywanie do zmiennej informacji, czy urządzenie jest tabletem.

```
StaticVariables.isTablet = resources.getBoolean(R.bool.isTablet)
```

Ustawianie odpowiedniego układu w zależności od urządzenia.

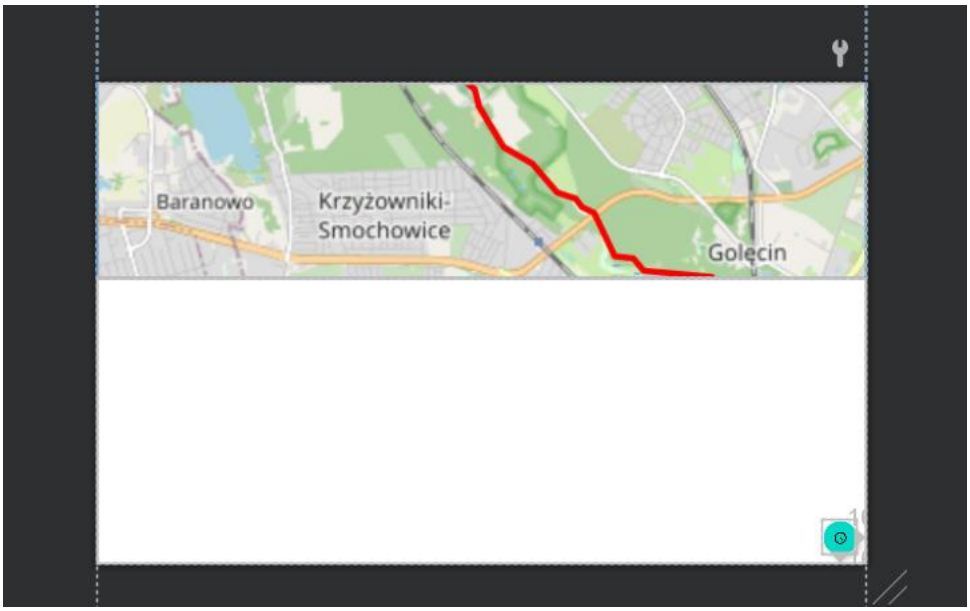
```
if (StaticVariables.isTablet) {
    setContentView(R.layout.activity_main_tab)
} else {
    setContentView(R.layout.activity_main)
}
```

Dla tabletów wyświetlane są 4 kolumny w aktywności listy.

```
if(StaticVariables.isTablet){
    easy.layoutManager = GridLayoutManager(activity, spanCount: 4)
}
else{
    easy.layoutManager = GridLayoutManager(activity, spanCount: 2)
}
```

```
if(StaticVariables.isTablet){
    hard.layoutManager = GridLayoutManager(activity, spanCount: 4)
}
else{
    hard.layoutManager = GridLayoutManager(activity, spanCount: 2)
}
```

Sprawdzenie, czy aplikacja jest responsywna dla tabletu. Oczywiście jest.



Animacje

Animacje zostały wykonane zgodnie z wykładem. Zmieniony został sposób ruchu słońca (bardziej naturalny) oraz kolory tła.

Animacja wykonuje się przez 6,5 sekundy. Kod oczywiście oparty o fragmenty.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/androi
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Activites.StartActivity"
    android:id="@+id/startActivity"
    android:orientation="vertical">
    <include
        layout="@layout/toolbar"
        android:id="@+id/toolbar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <fragment
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:name="com.example.lab8_12.Fragments.StartFragment"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="0.9" />
</LinearLayout>
```