AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Aplikacja Survivalowa

Autor:
Bogusław Gruca
Karol Kowalik

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2022

Spis treści

1. Ogólne określenie wymagań

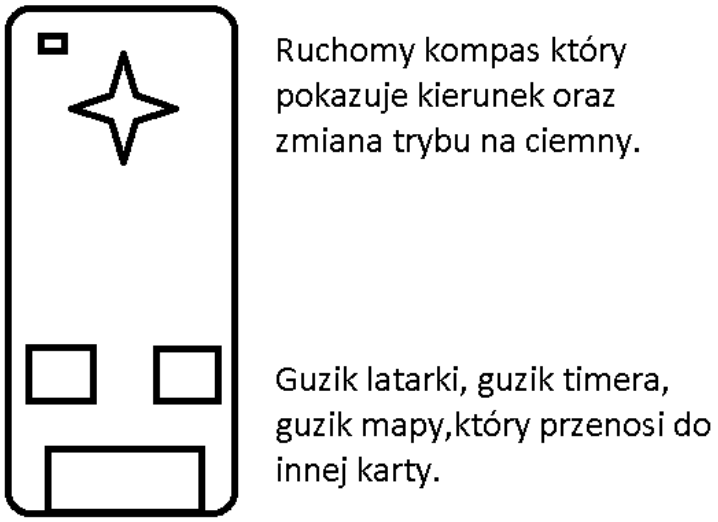
Celem programu jest utworzenie prostej w obsłudze oraz uniwersalnej w każdej sytuacji aplikacji, dzięki której będzie możliwe bezpieczne poruszanie się w terenie.Jeśli zamierzasz pojechać do lasu i chcesz zaoszczędzić jak najwięcej miejsca, korzystanie z aplikacji w telefonie to dobry pomysł, o ile masz przy sobie tak ze trzy powerbanki lub ewentualnie ładowarkę na korbkę. Mimo wszystko niektóre aplikacje mogą się przydać, nawet podczas zwykłego spaceru w lesie.

Aplikacja powinna zawierać wszystkie najważniejsze narzędzia do posługiwania się poza domem. Odbiorcą ma być każdy, niezależnie od wieku liczy się prostota w obsłudze, aby nikt nie miał problemu z użytkowaniem.

Aplikacja ma być obsługiwana przez system Android. Program musi posiadać dostęp do mapy, która będzie umożliwiała orientowanie się w terenie górzystym, zabudowanym, leśnym.Aplikacja mobilna może być świetnym narzędziem lojalnościowym, dzięki któremu zbierzemy bazę swoich klientów, bogatym źródłem informacji o nich, dostarczającym wartościowe dane o ich aktywnościach w kanale mobile. Ponieważ ze smartfonem w zasadzie nie rozstajemy, zaletą aplikacji mobilnej jest to, że dane te mogą być na bieżąco i stale aktualizowane. Są to informacje o rzeczywistych działaniach użytkowników, które podejmowane są każdego dnia, nie dane deklaratywne, stąd niezwykle cenne. Dane te można połączyć z innymi informacjami pozyskiwanymi z kanału mobile, a także z danymi zebranymi z desktopu czy offline.

Zleceniodawca życzy sobie oby kompas umiejscowiony był na górze ekranu na środku, a poniżej guziki funkcyjne. Kompas ma się poruszać w prawidłowe kierunki po zmianie położenia telefonu oraz wyświetlać stopnie gdzie połnoc to stan początkowy $90°C$

Przyciski, po których naciśnięciu będziemy przechodzili to osobnej karty mają być zaprojektowane według uznania projektantów.



Rys. 1.1

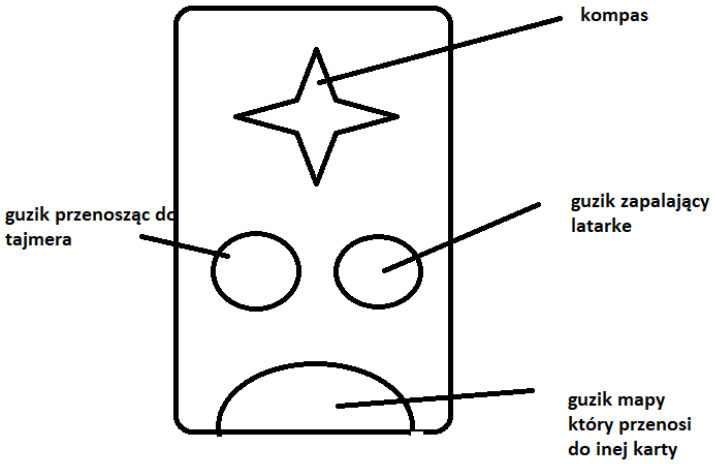
2. Określenie wymagań szczegółowych

Naszym celem w budowaniu aplikacji jest zaprogramowanie tak,aby posiadała następujące funkcje:
-mape
-latarke
-kompas
-stoper który będzie odliczał czas do przerwy
-lokalizacja
Aplikacja będzie miała prosty interfejs. Brak internetu będzie skutkował utrudnieniami w korzystaniu z aplikacji. Zaczniemy budowe programu od zaprojektowania graficznego aplikacji i ułożenia danych narzędzi,przycisków. Po ustaleniu grafiki i przycisków funkcyjnych (Rys.2.1), przejdziemy do rozpoczęnia kodowania aplikacji i podłączać mape z lokalizacją.Po zaprogramowaniu aplikacji będziemy sprawdzali jej prawidłowe działanie i poprawiali błedy. Program będziemy rozwijali w przyszłości o kolejne funkcje. Wygląd guzika latarki będzie się zmieniał w zależności od stanu włączenia lub wyłączenia latarki.Te stany są przedstawione na ryskach (Rys.2.4) oraz (Rys.2.5)

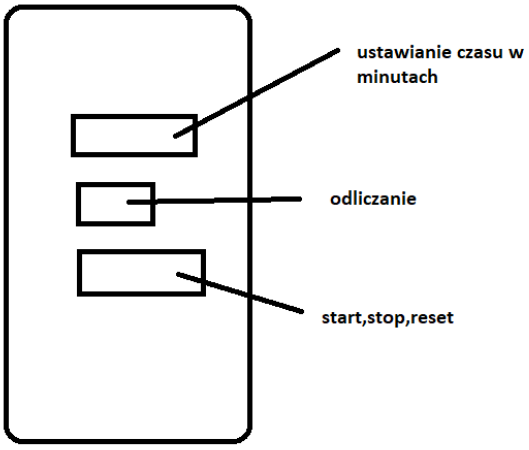
Przycisk latarki będącej w stanie off.(Rys.2.4)

Przycisk latarki będącej w stanie on.(Rys.2.5)

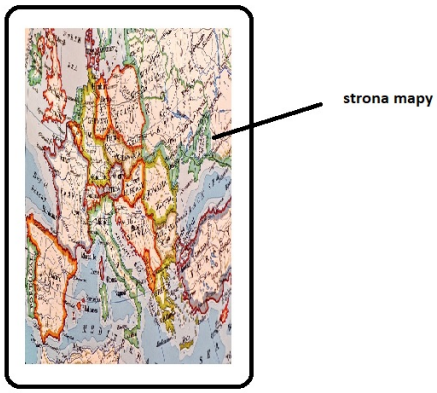
Wygląd kompasu.(Rys.2.6)



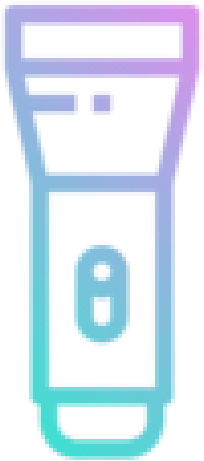
Rys. 2.1. Strona głowna.



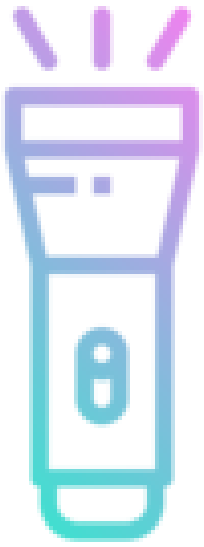
Rys. 2.2. Strona z minutnikiem.



Rys. 2.3. Strona z mapą.



Rys. 2.4. Przycisk latarki będącej w stanie off.



Rys. 2.5. Przycisk latarki będącej w stanie on.

Do kompasu który zaprojektujemy, aby się poruszał zgodnie z danymi pobranymi z żyroskopu i położenie geograficzne. Dodamy liczbowe stopnie nad kompasem.

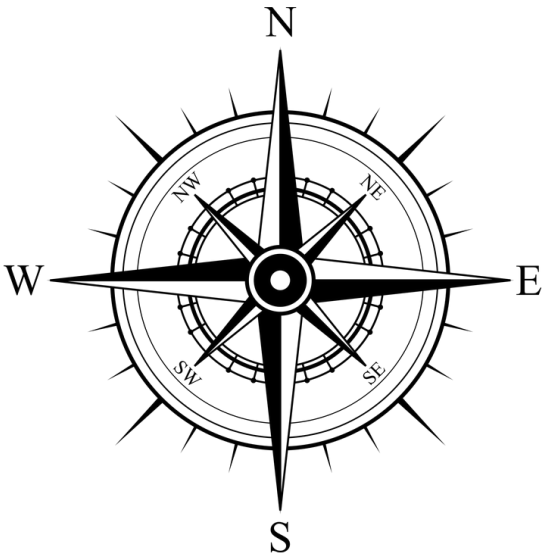
Kliknięcie ikonki timera (Rys.2.7) przenosi nas do osobnej karty, nad której będzie znajdował się timer. Na stronie timera będzie możliwość podania minut, po wprowadzeniu czasu będzie można go uruchomić, zastopować lub zresetować.

Przycisk mapy. (Rys.2.8)

Po kliknięciu przycisku mapy (Rys.2.5) przeniesie nas do nowej karty którą zaprojektujemy, gdzie będzie mapa google wraz z lokalizacją naszego aktualnego położenia.

Do aplikacji zostanie dodana funkcja trybu ciemnego, za pomocą przycisku w lewym górnym rogu ekranu głownego pojawi się możliwość włączenia lub wyłączenia trybu ciemnego.Tryb ciemny w telefonie włączamy głównie po to, by jasny ekran nie wypalał nam oczu. Innym też po prostu bardziej się taki motyw podoba. Jest jednak również taka grupa użytkowników, którzy aktywują go, aby zaoszczędzić energię i tym samym wydłużyć czas działania od ładowania do ładowania

Wygląd przycisku timera.(Rys.2.4)



Rys. 2.6. Wygląd kompasu.

Rys. 2.7. Przycisk timera.

Rys. 2.8. Przycisk mapy.

3. Projektowanie

Do projektu wykorzystaliśmy narzędzia takie jak GIT czyli rozproszony system kontroli wersji, oprogramowanie służące do śledzenia zmian głównie w kodzie źródłowym oraz pomocy programistom w łączeniu zmian dokonanych w plikach przez wiele osób w różnym czasie.

GitHub– hostingowy serwis internetowy przeznaczony do projektów programistycznych wykorzystujących system kontroli wersji Git. Stworzony został przy wykorzystaniu frameworka Ruby on Rails i języka Erlang. Serwis działa od kwietnia 2008 roku. GitHub udostępnia darmowy hosting programów open source i prywatnych repozytoriów (część funkcji w ramach prywatnych repozytoriów jest płatna)(Rys.3.1).

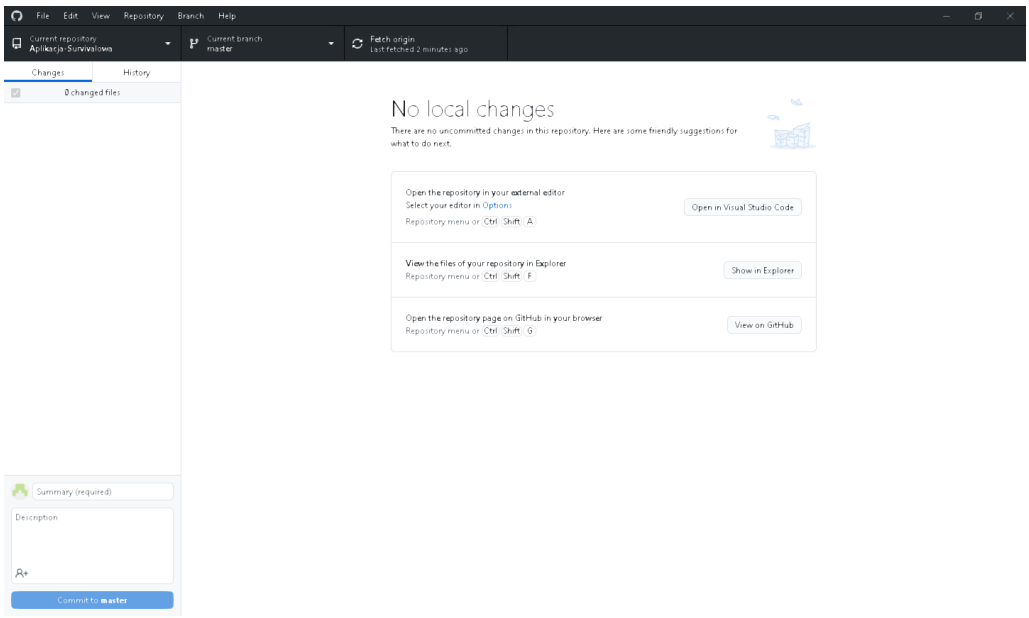
Aby efektowanie pracować w grupie, utworzyliśmy repozytorium Aplikacja-Survivalowa, gdzie umieściliśmy kod do naszej aplikacji wraz z dokumentacją, dzięki temu możemy w prosty sposób wprowadzać zmiany oraz ulepszać program.Do zapisywania oraz wysyłania zmian używamy programu GitHub Desktop (Rys.3.2).

Do wykonania dokumentacji oraz raportów wykorzystaliśmy TeXstudio to wieloplatformowy edytor LaTeX typu open source. Jego funkcje obejmują interaktywne sprawdzanie pisowni, składanie kodu i podświetlanie składni. Nie zapewnia samego LaTeX-a – użytkownik musi najpierw wybrać dystrybucję TeX-a i ją zainstalować.

Pierwotnie nazywany TexMakerX, TeXstudio został uruchomiony jako rozwidlenie Texmakera, który próbował rozszerzyć go o dodatkowe funkcje, zachowując jednocześnie jego wygląd i styl.

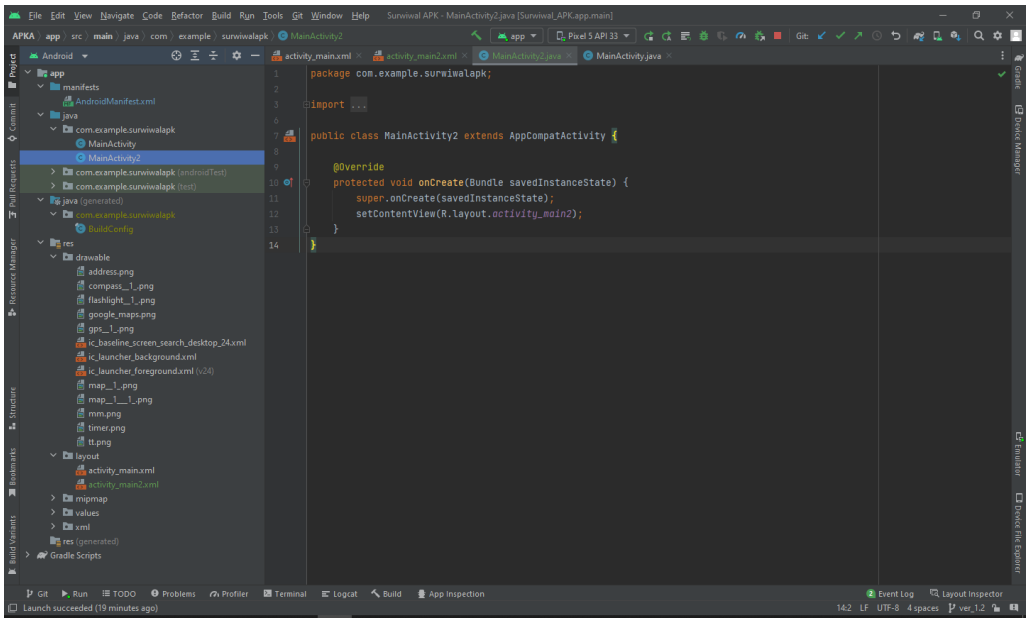
Język w jakim piszemy aplikację to Java (Rys.3.1) czyli wysokopoziomowy język programowania najczęściej wykorzystywany do tworzenia backendu aplikacji internetowych. Język ten jest łatwo przenośny, dzięki interpretowaniu przez wieloplatformową maszynę wirtualną Java Virtual Machine. Można stwierdzić, że Java jest językiem preferowanym przez korporacje i duże firmy. W Javie napisano m.in. takie aplikacje jak Gmail, OpenOffice czy Minecraft, ale także LinkedIn, Netflix czy Amazon.
Javę z założenia cechuje:
-obiektowość,
-dziedziczenie,
-niezależność od architektury,
-sieciowość,
-niezawodność,
-bezpieczeństwo.

Rys. 3.1. Java[**1**]



Rys. 3.2. GitHub Setup[**1**]

Środowiskiem programistycznym gdzie piszemy naszą aplikacje jest Android Studio (Rys.3.3) czyli środowisko programistyczne (IDE) stworzone przez Google na bazie IntelliJ, które kierowane jest do developerów aplikacji na Androida. Pozwala ono wygodnie projektować, tworzyć i debugować własne programy na najpopularniejszą obecnie platformę systemową dla urządzeń mobilnych.



Rys. 3.3. Android Studio[1]

4. Implementacja

Na listingu nr. 1 przedstawiony jest kod utworzenia przyciku latarki oraz pozycjonowanie na ekranie. Przykładowo w linijce 7 kodu marginBottom - oznacza zachować odległość od dołu widoku

```
1  <ImageButton
2    android:id="@+id/imageButton14"
3    android:layout_width="wrap_content"
4    android:layout_height="wrap_content"
5    android:layout_marginStart="203dp"
6    android:layout_marginEnd="16dp"
7    android:layout_marginBottom="260dp"
8    android:backgroundTint="#00FFFFFF"
9    app:layout_constraintBottom_toBottomOf="parent"
10   app:layout_constraintEnd_toEndOf="parent"
11   app:layout_constraintStart_toStartOf="@+id/imageButton12"
12   app:srcCompat="@drawable/flashlight__1_" />
13
```

Listing 1. Przykładowy kod Przycisku latarki

Na listingu nr. 2 przedstawiony jest kod przycisku wyłączonej latarki.

Linia 8 kodu - stosuje filtr kolorów do zasobu android:background, gdy jest używany razem z android:backgroundTintMode

Linia 10 kodu - służy do układania widoków w oparciu o niektóre formy relacji, które mają względem siebie.

```
<ImageButton
  android:id="@+id/imageButton"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_marginBottom="24dp"
  android:backgroundTint="#00FFFFFF"
  app:layout_constraintBottom_toTopOf="@+id/imageButton19"
  app:layout_constraintEnd_toEndOf="parent"
  app:layout_constraintHorizontal_bias="0.428"
  app:layout_constraintStart_toEndOf="@+id/imageButton12"
  app:srcCompat="@drawable/off" />
```

Listing 2. Przykładowy kod przycisku wyłączonej latarki

Na listingu nr. 3 przedstawiony jest kod definiowaniowania zmiennych.

```
private ImageButton toggleButton;
boolean hasCameraFlash = false;
boolean flashOn = false;
```

Listing 3. Przykładowy kod definiowania zmiennych

Na listingu nr. 4 przedstawiony jest kod pozwalający na wyłączenie latarki.

W liniach 1-11 przedstawiony mechanizm zmiany ikonki po klkiknięciu na latarke. Kod w linijce 14-19 powoduje wywołanie funkcji latarki w stan ON.

```
1    public void onClick(View view) {
2  if (hasCameraFlash){
3    if (flashOn){
4      flashOn = false;
5      toggleButton.setImageResource(R.drawable.off);
6      try {
7        flashLightOff();
8      } catch (CameraAccessException e) {
9        e.printStackTrace();
10     }
11   }
12
13
14   private void flashLightOn() throws CameraAccessException {
15     CameraManager cameraManager = (CameraManager)
   getSystemService(Context.CAMERA_SERVICE);
16   assert cameraManager != null;
17   String cameraId = cameraManager.getCameraIdList()[0];
18     cameraManager.setTorchMode(cameraId, true);
19     Toast.makeText(MainActivity.this, "FlashLight is ON",
   Toast.LENGTH_SHORT).show();
```

```
}
```

Listing 4. Przykładowy kod ukazujący mechanizm wyłączenia latarki.

Na listingu nr.5 przedstawiony jest kod prezentujący mechanizm przechodzenia do nowej strony poprzez kliknięcie przycisku na stronie głownej.

```
    toggleButton2 = findViewById(R.id.imageButton2);
  toggleButton2.setOnClickListener(new View.OnClickListener() {
  @Override//nowa strona tajmer
  public void onClick(View view) {
    openTajmer();
  }
});
```

Listing 5. Przechodzenie do nowej strony.

Na listingu nr.6 przedstawiony jest kod pozwalający na wyświetlenie przycisku restart.

```
<Button
  android:id="@+id/button_reset"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_below="@+id/text_view_countdown"
  android:layout_marginStart="36dp"
  android:layout_toEndOf="@+id/button_start_pause"
  android:text="reset"
  android:visibility="invisible"
  app:layout_constraintBaseline_toBaselineOf="@+id/
   button_start_pause"
  app:layout_constraintEnd_toEndOf="parent"
  app:layout_constraintHorizontal_bias="0.0"
  app:layout_constraintStart_toEndOf="@+id/button_start_pause"
  tools:visibility="visible" />
```

Listing 6. Przycisk resetujący czas..

Na listingu 7 przedstawiona jest funkcja SensorManager, która umożliwia dostęp do czujników urządzenia.

```
accelerometerSensor = sensorManager.getDefaultSensor(Sensor.
    TYPE_ACCELEROMETER);
magnetometerSensor = sensorManager.getDefaultSensor(Sensor.
    TYPE_MAGNETIC_FIELD);
sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

Listing 7. Czujniki potrzebne przy funkcjonalności kompasu.

Na listingu 8 przedstawiony wygląd layoutu mapy.

Linijka 5 kodu przedstawia identyfikator zasobu. Unikatowa nazwa zasobu dla elementu, której można użyć do uzyskania odwołania do widoku z aplikacji.

Linijka 6 kodu przedstawia komponent mapy w aplikacji. Ten fragment to najprostszy sposób na umieszczenie mapy w aplikacji.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <fragment xmlns:android="http://schemas.android.com/apk/res/android
3 xmlns:map="http://schemas.android.com/apk/res-auto"
4 xmlns:tools="http://schemas.android.com/tools"
5 android:id="@+id/map"
6 android:name="com.google.android.gms.maps.SupportMapFragment"
7 android:layout_width="match_parent"
8 android:layout_height="match_parent"
9 tools:context=".Map" />
```

Listing 8. Wygląd layoutu mapy.

Na listingu 9 zadeklarowano użyte sensory do prawidłowego działania kompasu.

```
1 textView = findViewById(R.id.textView);
2
3 imageView = findViewById(R.id.imageView);
4
5 sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
6 accelerometerSensor = sensorManager.getDefaultSensor(Sensor.
     TYPE_ACCELEROMETER);
7 magnetometerSensor = sensorManager.getDefaultSensor(Sensor.
     TYPE_MAGNETIC_FIELD);
```

Listing 9. Użyte sensory do prawidłowego działania kompasu..

5. Testowanie

6. Podręcznik użytkownika

Po zainstalowaniu aplikacji pojawi się ikonka na naszym telefonie w miejscu docelowym dla naszego urządzenia mobilnego.(Rys.6.1)

Uruchomienie aplikacji nastąpi wtedy,gdy klikniemy na ikonke aplikacji. Czas uruchomienia, zależy od naszego sprzętu lecz nie powinno to zająć długo. Po uruchomieniu wyświetli się głowny ekran aplikacji.(Rys.6.2)

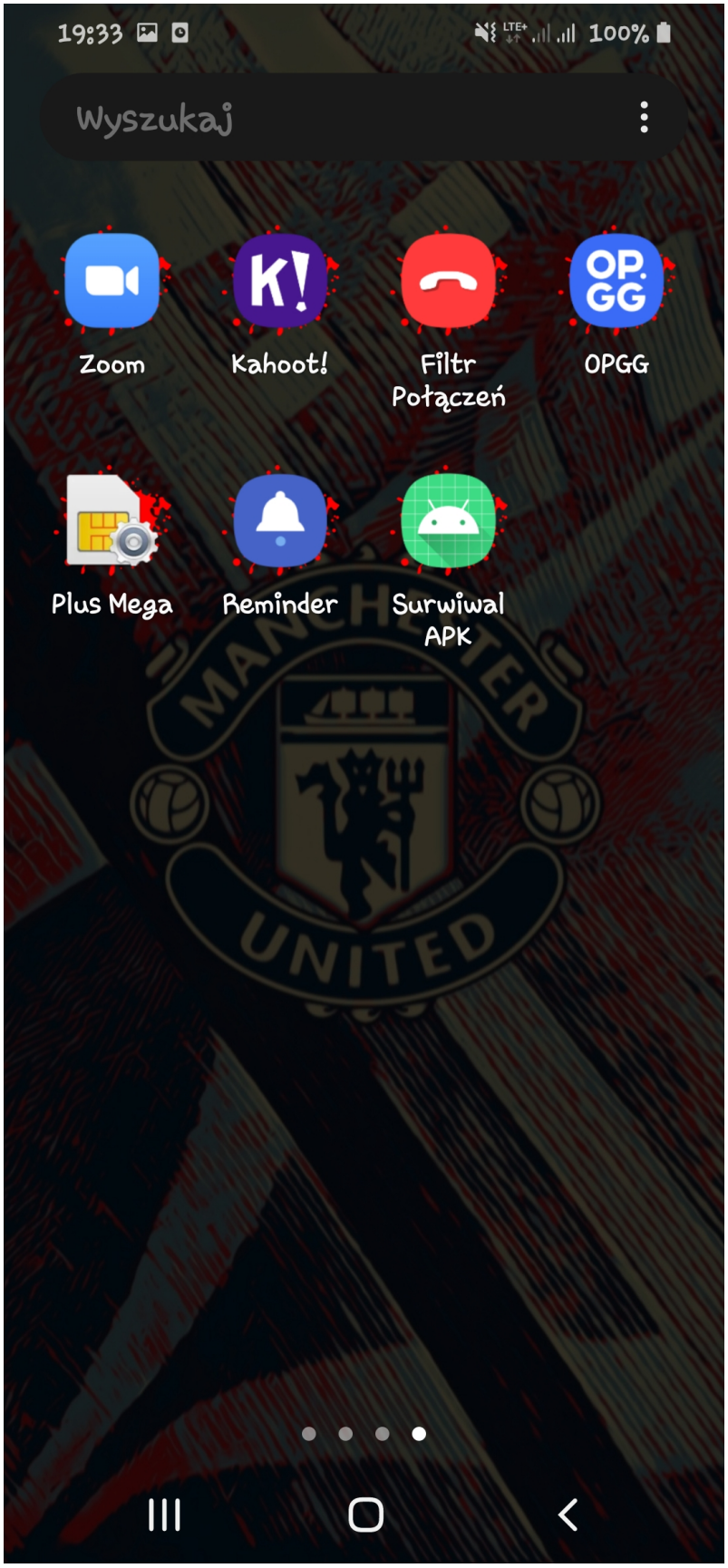
Kompas jest głowną funkcją aplikacji. Musimy trzymać nasze urządzenie równolegle do podłoża, następnie za pomocą ruchu telefonu lub naszej osoby możemy sprawdzić odpowiednie kierunki.

Przycisk znajdujący się w lewym górnym rogu odpowiada za zmianę trybu dziennego w nocny.(Rys.6.3)

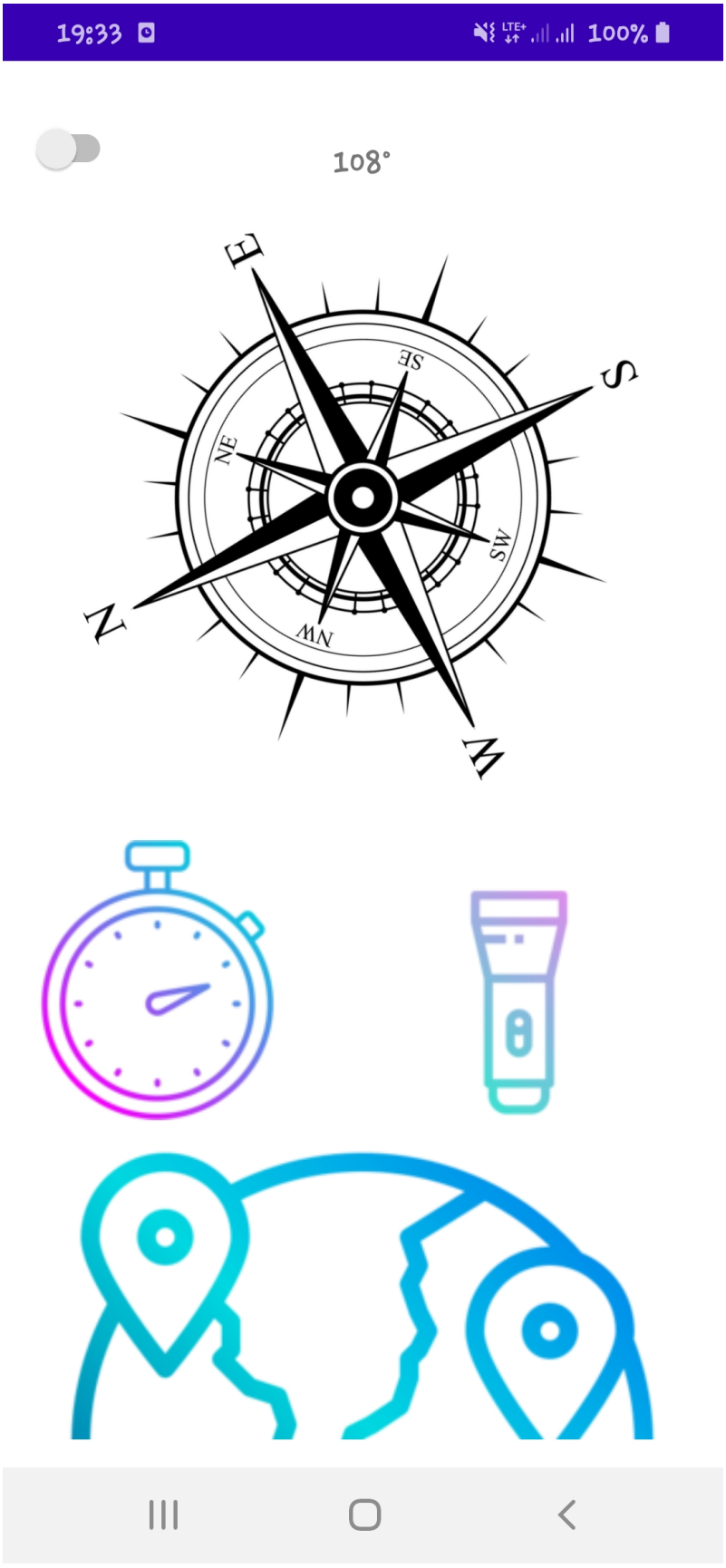
Na dole ekranu znajduje się przycisk mapy, po kliknieciu zostaniemy przeniesieni do osobnego okna z mapą(Rys.6.4)

Poruszamy się miedzy ekranami za pomocą pasku nawigacyjnego naszego urządzenia. Gdy wrócimy do ekranu głownego nad mapą mamy dwa przyciski.Ten po lewej służy do przejścia do ekranu z timerem, gdzie mamy do dyspozycji pole w którym wpisujemy liczbe minut do odliczenia. W każdej chwili możemy zatrzymać lub zresetować czas.(Rys.6.5)

Na prawo od przycisku timera mamy guzik latarki, który po kliknięciu zmienia ikonke.(Rys.6.6)

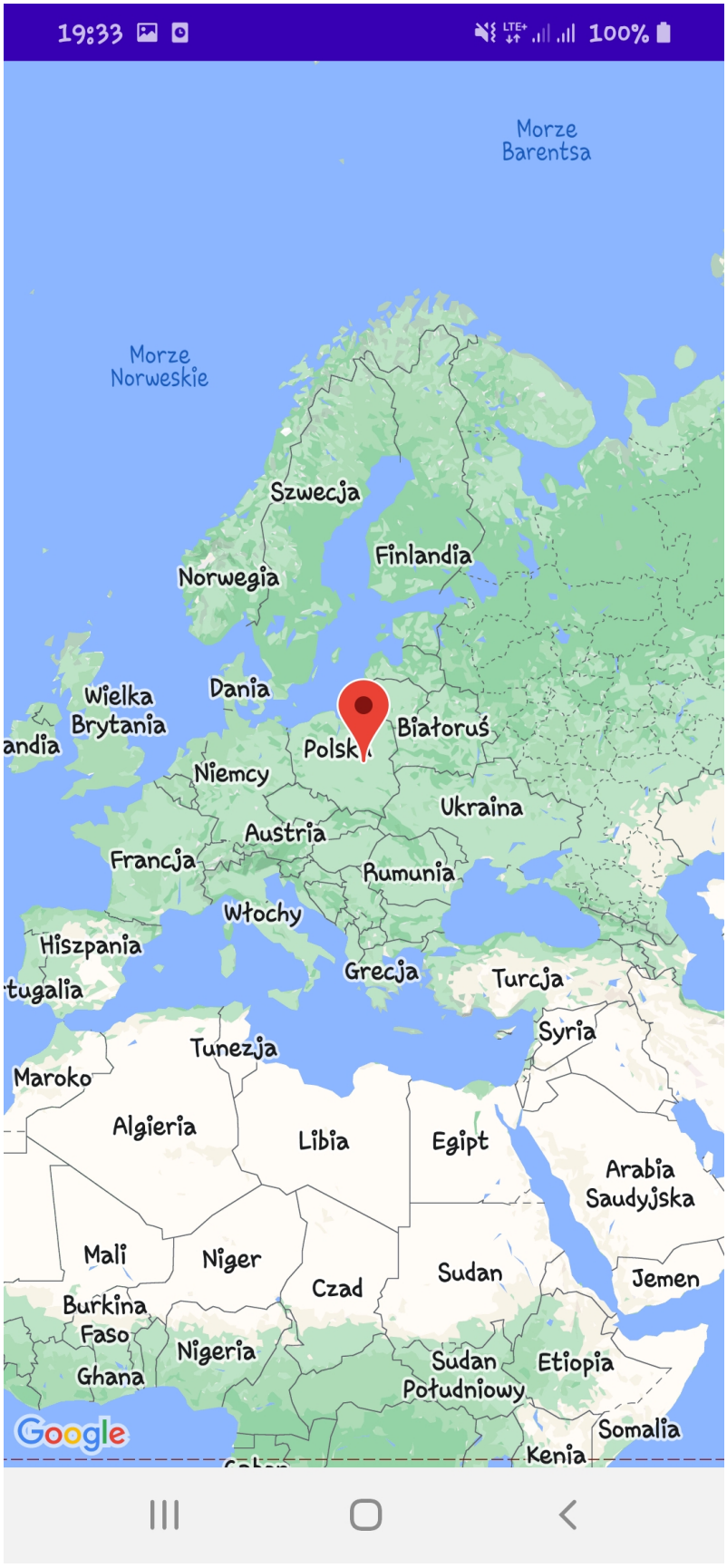


Rys. 6.1. Wygląd aplikacji.

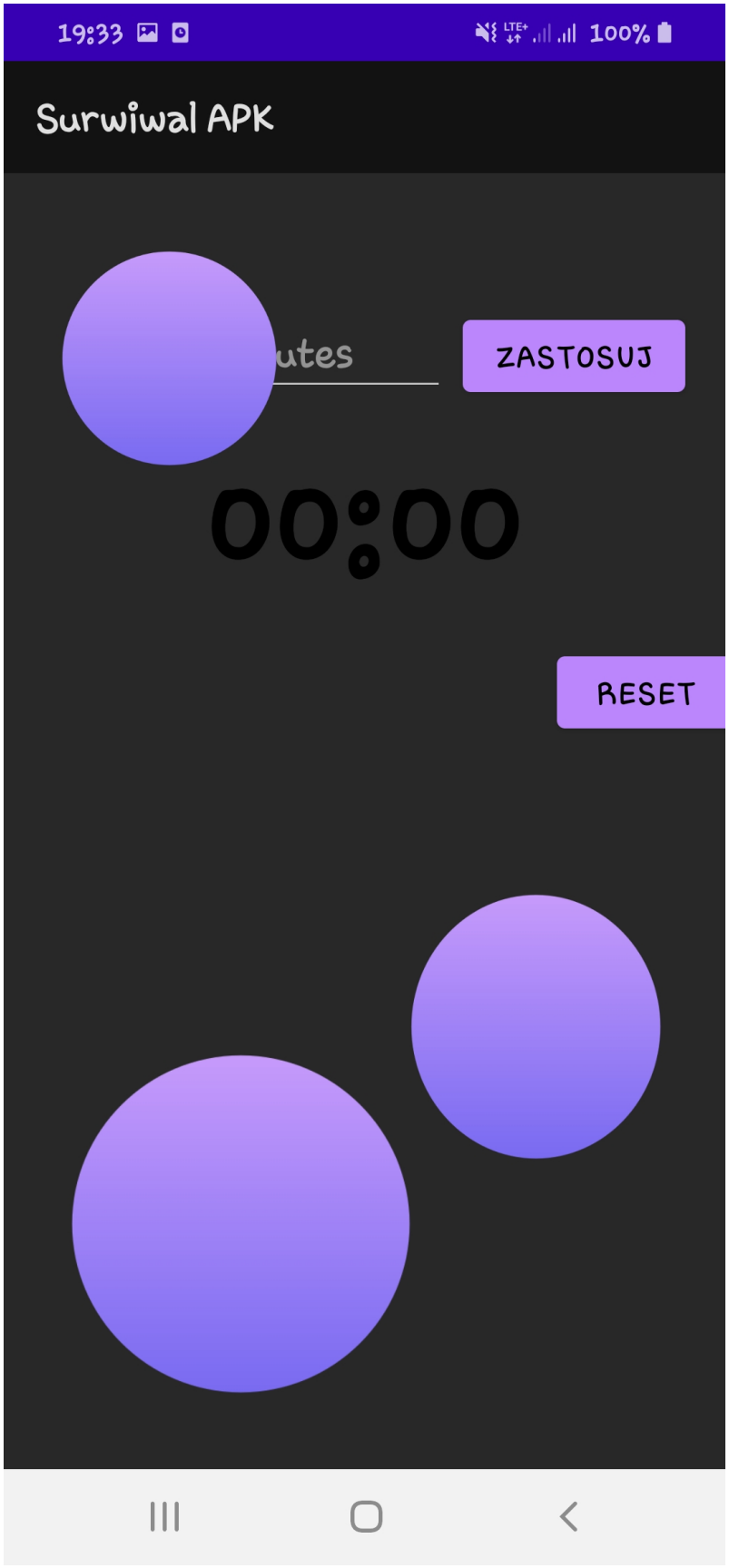


Rys. 6.2. Głowny ekran aplikacji.

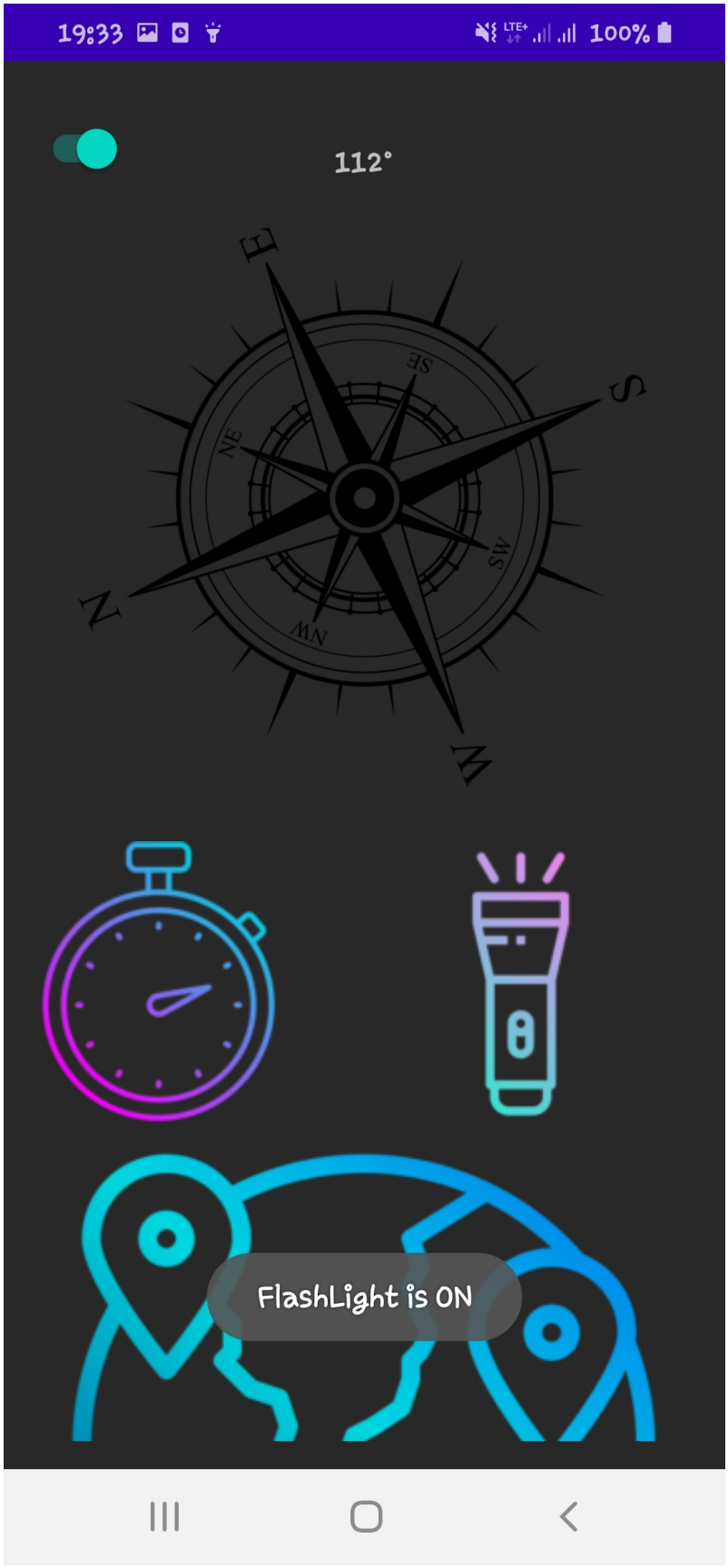
Rys. 6.3. Tryb nocny.



Rys. 6.4. Ekran z mapą.



Rys. 6.5



Rys. 6.6

Spis rysunków

Spis tabel

Spis listingów