AKADEMIA NAUK STOSOWANYCH
W NOWYM SĄCZU

Wydział Nauk Inżynieryjnych
Katedra Informatyki

DOKUMENTACJA PROJEKTOWA

PROGRAMOWANIE URZĄDZEŃ MOBILNYCH

Ttest

Autor:
Paulina Hudzik
Magdalena Krzyszowska

Prowadzący:
mgr inż. Dawid Kotlarski

Nowy Sącz 2022

Spis treści

1. Ogólne określenie wymagań

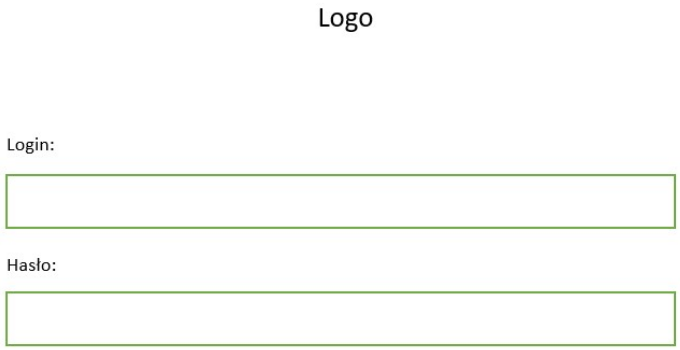
Nowoczesne smartfony są naszpikowane różnego rodzaju czujnikami. To one sprawiają, że codzienne używanie telefonów komórkowych jest łatwe i przyjemne. Dzięki nim urządzenia zawsze wiedzą gdzie jesteśmy, w którym kierunku się poruszamy oraz w jaki sposób je trzymamy. Wygaszanie ekranu, gdy zbliżymy słuchawkę do policzka, automatyczna zmiana orientacji wyświetlanego obrazu, kiedy obrócimy telefon czy w końcu sterowanie zaawansowanymi grami - to wszystko nie byłoby możliwe bez czujników. Ogólnym zamysłem naszego projektu jest stworzenie aplikacji która ma służyć do testowania czujników w telefonach, co ułatwiłoby i przyśpieszyło pracę w serwisie komórkowym. Zależy nam na możliwości sprawdzenia czujników takich jak:
- latarka,
- test dźwięku,
- bluetooth,
- gps,
- tryb ciemny
- czujnik zbliżeniowy.

Nasza firma powstała w roku 2011; ponad 10 lat prężnie pracujemy oraz staramy się dbać nie tylko o naszych klientów lecz i o naszych pracowników, stąd też pojawił się pomysł na wdrożenie nowej aplikacji, która to pozwoliła by na jeszcze większe usprawnienie pracy w naszym zespole. Co roku zatrudniamy kilkaset nowych osób i to właśnie do nich w dużej mierze miałaby trafić nowa aplikacja. Poprzednie aplikacje, które posiada nasza firma zdecydowanie nie nadają się już do użytku, ponieważ mają wiele mankamentów, takich jak na przykład: brak możliwości wykonania testu GPS'a, który to obecnie jest jednym z najpopularniejszych czujników w telefonach, z którego na co dzień korzysta setki tysięcy osób. Co więcej poprzednie aplikacje miały tendencje do ścinania się tuż po uruchomieniu, dlatego bardzo zależy nam na tym, aby aplikacja funkcjonowała w sposób szybki. Ponadto chcielibyśmy aby aplikacja była zachowana w sposób minimalistyczny i intuicyjny, przez co nawet starsze osoby, które będą z niej korzystać, w szybki sposób będą mogły bez pomocy innych, sprawdzić poszczególne czujniki. Przejrzysta oraz prosta budowa menu, usprawni szukanie odpowiednich komponentów. Mile widziane byłoby zastosowanie panelu logowania w którym to możliwe było by logowanie się do aplikacji poprzez podanie loginu bądź też e-maila oraz hasła. Dzięki temu każdy z naszych pracowników poprzez swój własne dane mógłby w każdej chwili i w każdym miejscu zalogować się do aplikacji.

Kolorem przewodnim naszej firmy jest niebieski, dlatego zależy nam na tym aby akcenty właśnie w tym kolorze pojawiły się w aplikacji; mogą to być na przykład przyciski czy chociażby kolor tła aplikacji. Nasza firma posiada również własne logo, które powinno zawierać się w projekcie.
Na rysunku 1.1 przedstawione jest logo firmy:

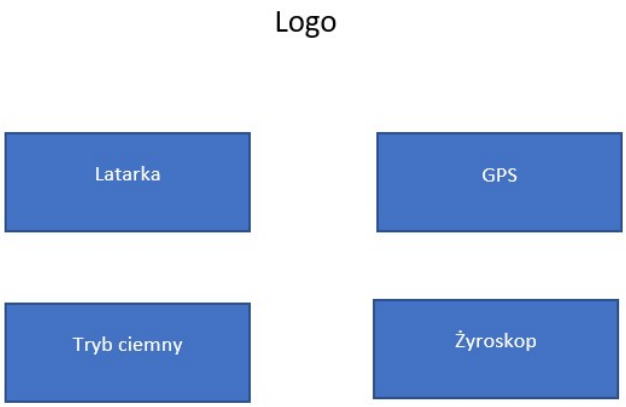
Rys. 1.1. Logo firmy TTest

Jak wspomnieliśmy powyżej chcielibyśmy również aby pojawił się panel logowania, dzięki któremu zarówno pracownicy jak i klienci będą mogli się zalogować do aplikacji i korzystać z niej bez obaw, że ktoś inny będzie mógł na przykład namierzyć ich lokalizację. Rysunek 1.2 ukazuję przykładowy panel logowania.



Rys. 1.2. Przykładowy panel logowania

Poniżej na rysunku 1.3 przedstawiamy wstępny layout aplikacji na jakim nam zależy, jeżeli chodzi o przyciski to zależałoby nam na tym aby zamiast wpisanych nazwy tak jak zostało to zaprojektowane na wstępnym layoucie pojawiły się ikonki przedstawiające daną funkcje; na przykład zamiast napisu "Latarka", pojawi się przycisk z wizerunkiem przedstawiającym latarkę; sprawi to że aplikacja będzie wyglądać jeszcze bardziej przejrzyście.



Rys. 1.3. Przykładowy layout

2. Określenie wymagań szczegółowych

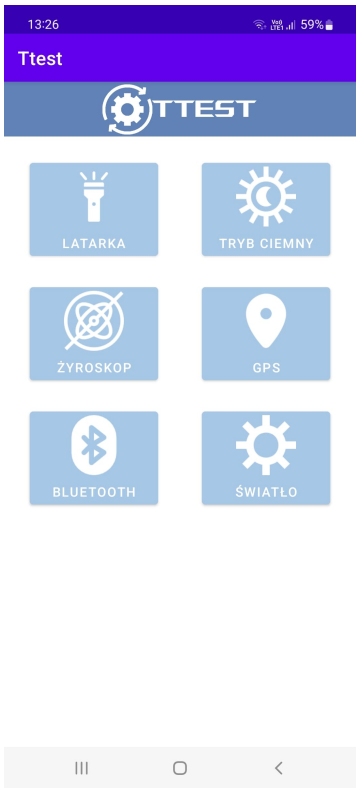
Firma nie związana z technologia pragnie aby stworzyć aplikację mobilną do testowania czujników, która byłaby zachowana w sposób minimalistyczny i intuicyjny. Przejrzysta i prosta budowa menu, która miałaby na celu usprawnić szukanie odpowiednich komponentów. Firma posiada również logo, które powinno zawierać się w projekcie, jak i niebieskie akcenty (np. przyciski) - który jest przewodnim kolorem firmy.

Jednym z kluczowych zadań jest stworzenie aplikacji w sposób jak najbardziej przejrzysty, do tego celu użyjemy ikonek, które w prosty i szybki sposób nakierują nas na odpowiedni komponent. Przykładem może być latarka; chcąc przetestować jej działanie wystarczy wybrać ikonkę która będzie przedstawiała latarkę. Po jej naciśnięciu zostaniemy przekierowani do strony z testem, który będzie posiadał dwa przyciski jeden to zielony włącznik oraz czerwony wyłącznik. Ponadto będziemy starać się aby wyniki testów były stopniowo zapisywane w pliku wyjściowym. Jako wykonawcy aplikacji chcielibyśmy wdrożyć do projektu przyciski które znajdowałyby się przy każdym teście i informowałyby o tym ,że: "test przebiegł pomyślnie"lub "test nie powiódł się". W zależnie od tego który przycisk naciśniemy to w taki sposób uzupełni się plik wyjściowy z podsumowaniem testu. Zapewniamy również zawarcie logotypu i motywu niebieskiego aplikacji, i sprawimy by layout w całości był jak najbardziej intuicyjny i zachowany w minimalistyczny sposób (brak jaskrawych kolorów, duże przejrzyste przyciski, okienka pop-up z wiadomościami o przebiegu testu). Postaramy się aby komponenty testujące takie jak: latarka, test dźwięku, czujnik światła, tryb nocny i tym podobne; będą poprawnie spełniać swoje zadanie. Aplikacja zostanie napisana w programie Android Studio, natomiast język którym będziemy się posługiwać to: Java.

Tak jak już wspomnieliśmy powyżej postaramy się by każdy z czujników działał w należyty sposób; to znaczy by latarka poprzez kliknięcie przycisków włączała się i wyłączała. Test dźwięku pozwoli nam poprzez kliknięcie przycisku usłyszeć z głośników wydobywającą się melodię; tryb nocny, który jest bardzo przydatną funkcją w telefonie szczególnie wieczorami gdy korzystamy z urządzeń mobilnych, zmieni kolorystykę z jasnej na ciemną. Przez tą funkcję jaka jest tryb ciemny nasze oczy nie są narażane na tak mocne światło, co sprawia, że czujemy większy komfort w użytkowaniu telefonów komórkowych. Niektóre osoby preferują korzystanie z trybu nocnego nawet podczas dnia a nie tylko nocą. Kolejny czujnik jakim będzie GPS

pozwoli nam w szybki sposób określić położenie w którym się znajdujemy. Test Wifi umożliwi spawdzenie połączenia z siecią, natomiast test mikrofonu sprawdzi się w sytuacji gdy na przykład pojawi się problem podczas rozmów telefonicznych. Test aparatu pozwoli wykonać zdjęcie oraz przetestować jakość wykonanego zdjęcia.
Pragnieniem firmy, która zleciła nam wykonanie aplikacji, jest stworzenie panelu logowania, dzięki któremu pracownicy wraz z klientami będą mogli logować się do aplikacji. Naszym zdaniem nie ma konieczności do tworzenia odrębnego panelu do logowania, ponieważ aplikacja wykonuje tylko testy czujników i nie przechowuję poufnych danych. Co więcej w sytuacji gdy kilkaset osób postanowi wejść w aplikację i zalogować się na nią, może dojść do przeciążenia i zawieszenia się strony, co wiąże się z tym, że nabywcy aplikacji będą musieli poczekać dłuższą chwilę aby logowanie się powiodło. Uważamy, że w dużych firmach mogłoby to spowolnić pracę pracowników, a czas pracy odgrywa bardzo ważną rolę.

Rozpoczęliśmy pracę przy tworzeniu wstępnego layout strony głównej (rysunek 2.1) i na ten moment prezentuję się ona następująco:



Rys. 2.1. Wstępne menu

Na tą chwilę udało nam się stworzyć menu z sześcioma przyciskami, które prezentują poszczególne czujniki. Aktualnie przyciski nie przekierowują jeszcze na stronę testów lecz tym zajmiemy się w kolejnym etapie naszej pracy. Co więcej na górze ekranu dokładniej w jego centrum umieszczone zostało logo firmy, które dostaliśmy w zleceniu. Kolorystyka jest utrzymana w błękicie oraz bieli.

3. Projektowanie

3.1. Kilka słów o środowisku Android studio oraz języku programowania Java.

Android Studio to 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. Oprogramowanie oferuje podobne możliwości co środowisko Eclipse z zainstalowaną wtyczką ADT, jednak jest ono znacznie prostsze i bardziej intuicyjne w szczególności dla początkujących programistów. Przejęło ono wszystkie najlepsze rozwiązania znane użytkownikom IntelliJ, oferując narzędzie zoptymalizowane w dużym stopniu do wygodnej pracy z kodem źródłowym. Decydując się na korzystanie z Android Studio użytkownik otrzymuje środowisko programistyczne z przejrzystym i konfigurowalnym interfejsem graficznym, nie wspominając o funkcji kolorowania składni czy mechanizmie zakładek, pozwalającym na pracę z wieloma plikami jednocześnie.

Java to 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.

3.2. Środowisko programisty/Składanie dokumentów - Latex

Latex służy do wytwarzania przejrzyście wyglądających dokumentów tekstowych takich jak książki, artykuły, czy nawet prezentacje. Docelowym formatem jest wydruk, czy też pliki w różnych formatach, takich jak PDF, Postscript, czy też HTML. Szczególnie wygodne jest tworzenie dokumentów technicznych, matematycznych, ale z powodzeniem może też być stosowany do pisania dokumentacji programów albo zbioru opowiadań.

Latex, podobnie jak języki programowania, ma swój własny język, w którym pisze się treść dokumentu oraz posiada narzędzia (można by powiedzieć "kompilatory"), które przetwarzają pliki źródłowe i generują pliki docelowe. W językach

programowania zazwyczaj jedną z istotnych rzeczy jest zbiór bibliotek z gotowymi implementacjami różnych typowych czynności. Również w Latexu jest dużo gotowych pakietów pozwalających w szybki sposób tworzyć najróżniejsze elementy i rodzaje dokumentów.

Filozofia Latexa jest taka, aby skupiać się na tym co merytorycznie ma zawierać dany dokument, a jak najmniej poświęcać uwagi na to, jak ma to wyglądać. Innymi słowy wprowadzamy tylko strukturę i zawartość dokumentu, a latex za nas robi resztę roboty, aby wyjściowy dokument wyglądał jak należy. Oczywiście mamy dużą możliwość ingerencji w wygląd, ale zazwyczaj jest to tylko dobieranie jakiegoś szablonu lub potrzeba uzyskania niestandardowego efektu. Jest to zupełnie inna filozofia, niż w wielu innych edytorach tekstowych, szczególnie w różnych aplikacjach biurowych, gdzie prawie na każdym kroku musimy od decydować, jaki ma być wygląd, wielkość liter, czcionka, odstępy, sposób wyświetlania tytułów itp.
Podstawą możliwości cieszenia się twórczością w Latexu jest posiadanie wszystkich narzędzi, pakietów, czcionek, itp. Gotowe zbiory są dostępne w różnych dystrybucjach. Oprócz tych narzędzi, początkujący użytkownicy mogą skorzystać z gotowych środowisk do obrabiania dokumentów Latexu.
Podstawową dystrybucją jest TeX Live. Jest ona dostępna pod wiele różnych platform. Jest łatwą w instalacji kompletną paczką narzędzi, programów, czcionek.

3.3. Czym jest Git oraz do czego służy?

Co to jest Git i dlaczego cieszy się tak dużą popularnością? Ten system kontroli wersji znacznie usprawnia, a jednocześnie zabezpiecza codzienną pracę przy kodzie. Dzięki swojej prostocie i elastyczności może być wykorzystywany zarówno przy drobnych, jak i ogromnych projektach. Dlatego też jest używany przez programistów oraz grafików na całym świecie. Odpowiadając w skrócie na pytanie, co to jest Git, należy powiedzieć, że to system kontroli wersji. Służy on więc do zarządzania historią kodu źródłowego. Jego funkcjonalność ma kilka podłoży. Między innymi sprawdza się tak dobrze, ponieważ
- pozwala na jednoczesną pracę na tym samym kodzie przez kilka osób,
- umożliwia transferowanie i łączenie zmian z różnych branchy w jednym projekcie
- pozwala na pracę offline we własnym repozytorium
- jest szybki i wydajny.

Cechy te sprawiły, że Git szybko został doceniony w całej branży. Przechowywanie wersji, a także możliwość rozgałęziania kodu to niewątpliwie jego ogromne zalety.

GitHub z kolei to firma, która hostuje repozytoria Git i dostarcza oprogramowanie do korzystania z niego. Jednym z przykładów jest tytułowy GitHub Desktop na systemy Windows 10 i 11. GitHub jest obecnie najpopularniejszym hostem projektów open source pod względem zarówno liczby projektów , jak i użytkowników. Choć GitHub koncentruje się głównie na kodzie źródłowym, to inne projekty coraz częściej wykorzystują systemy kontrolowania wersji do zarządzania przepływem pracy związanym z publikowaniem czasopism, artykułów, podręczników itp.

Test rozpoczynamy od uruchomienia naszej aplikacji a następnie pojawia nam się główne menu (rysunek 3.1), które składa się z dużych przycisków, zachowanych w kolorystyce firmy. Na każdym z przycisków widnieje ikonka przedstawiającą daną funkcję na przykład: chcemy przetestować latarkę; klikamy przycisk, który ją przedstawia.

Rys. 3.1. Przykładowy przycisk

Po kliknięciu ikonki, aplikacja przekierowuje nas do testu (rysunek 3.2), gdzie pojawiają się dwie latarki w centralnej części ekranu: po lewej stronie w kolorze zielonym a po prawej stronię w czerwonym. Kolor zielony ma na celu informować nas o tym że latarka jest włączona i działa, natomiast kolor czerwony sygnalizuje, że latarka jest wyłączona.

Rys. 3.2. Test latarki

4. Implementacja

Listing 1 (s. 13) przedstawia implementacje jednego z przycisków którego zadaniem jest otworzenie nowej strony oznaczonej w 5 linijce kodu po kliknięciu.

```
1  button_1 = (Button) findViewById(R.id.button_1);
2  button_1.setOnClickListener(new View.OnClickListener() {
3    @Override
4    public void onClick(View view) {
5      Intent intent = new Intent(MainActivity.this, latarka.class);
6      startActivity(intent);
7    }
8  });
```

Listing 1. Menu - Działanie Przycisków

Listing 2 (s. 13) przedstawia włączenie latarki. Jeśli telefon ma latarkę i mamy do niej dostęp, latarka zostanie uruchomiona i wyświetli się nam komunikat o włączeniu. Jeżeli nie, to dostaniemy odpowiednie powiadomienie o niepowodzeniu operacji. W podobny sposób wygląda metoda z wyłączeniem latarki. W 6 linijce trzeba zamienić wartość na false oraz w 7 wierszu nadpisać wyświetlany komunikat na odpowiedni.

```
1  private void flashLightOn(){
2    CameraManager cameraManager = (CameraManager) getSystemService(
   Context.CAMERA_SERVICE);
3    try{
4      assert cameraManager != null;
5      String cameraId = cameraManager.getCameraIdList()[0];
6      cameraManager.setTorchMode(cameraId, true);
7      Toast.makeText(latarka.this, "Latarka wlaczona", Toast.
   LENGTH_SHORT).show();
8    }
9    catch(CameraAccessException e){
10     Log.e("Camera Problem", "Nie mozna uruchomic latarki");
11   }
12 }
```

Listing 2. Latarka - Włączenie/wyłączenie latarki

Aby zaimplementować tryb ciemny potrzebujemy dostępu do modyfikowania interfejsu. Listing 3 (s. 14) pokazuje jak go uzyskać.

```
1  SharedPreferences sharedPreferences = getSharedPreferences("
    sharedPrefs", MODE_PRIVATE);
2  final SharedPreferences.Editor editor = sharedPreferences.edit();
3  final boolean isDarkModeOn = sharedPreferences.getBoolean("
    isDarkModeOn", false);
```

Listing 3. Tryb Ciemny - Modyfikowanie interfejsu

Listing 4 (s. 14) przedstawia metodę odpowiedzialną za działanie trybu ciemnego. Domyślnie aplikacja korzysta z trybu jansego, co pokazuje kod od 5 do 10 linijki. Jednak po naciśnięciu na przycisku, zmieniamy kolorystyke aplikacji na ciemną (13-15 linijak), dodatkowo od 16 linijki zmieniamy przycisk na czerwony, aktualne logo na ciemne oraz wyświetlamy stosowny komunikat.

```
1  toggle_tryb_ciemny.setOnClickListener(new View.OnClickListener()
    {
2     @Override
3     public void onClick(View view) {
4       if(isDarkModeOn){
5         AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.
    MODE_NIGHT_NO);
6         editor.putBoolean("isDarkModeOn", false);
7         editor.apply();
8         toggle_tryb_ciemny.setImageResource(R.drawable.
    tryb_ciemny_off);
9         logo.setImageResource(R.drawable.logo_white);
10        Toast.makeText(tryb_ciemny.this, "Tryb ciemny wylaczony",
    Toast.LENGTH_SHORT).show();
11      }
12      else {
13        AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.
    MODE_NIGHT_YES);
14        editor.putBoolean("isDarkModeOn", true);
15        editor.apply();
16        toggle_tryb_ciemny.setImageResource(R.drawable.
    tryb_ciemny_on);
17        logo.setImageResource(R.drawable.logo_black);
18        Toast.makeText(tryb_ciemny.this, "Tryb ciemny wlaczony",
    Toast.LENGTH_SHORT).show();
19      }
20    }
21 });
```

Listing 4. Tryb Ciemny - Włączenie/wyłączenie trybu ciemnego

Listing 5 (s. 15) prezentuje uzyskanie dostępu do sensora, za pomocą którego sprawdzimy działanie czujnika zbliżeniowego.

```
SensorManager sensorManager;
sensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
if(sensorManager!=null) {
  Sensor proximitySensor = sensorManager.getDefaultSensor(Sensor.
 TYPE_PROXIMITY);
  if(proximitySensor!=null) {
    sensorManager.registerListener(this, proximitySensor,
 sensorManager.SENSOR_DELAY_NORMAL);
  }
}
```

Listing 5. Czujnik Zbliżeniowy - Dostęp do czujnika

Metoda onSensorChanged ukazana na listingu 6 (s. 15) sprawdza czy przy czujniku jest obiekt, zależnie od wyniku zostanie wyświetlony odpowiedni komunikat.

```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
  if(sensorEvent.sensor.getType()==Sensor.TYPE_PROXIMITY) {
    if(sensorEvent.values[0]==0) {
      ((TextView)findViewById(R.id.sensor)).setText("Przy
 czujniku jest obiekt");
    } else {
      ((TextView)findViewById(R.id.sensor)).setText("Przyloz
 obiekt do czujnika");
    }
  }
}
```

Listing 6. Czujnik Zbliżeniowy - Działanie

Aby wdrożyć test aplikacji trzeba sprawdzić pozwolenie, w tym wypadku interesuje nas uprawnienie dostępu do lokalizacji, co obrazuje listing 7 (s. 15).

```
if(ContextCompat.checkSelfPermission(gps.this, Manifest.
 permission.ACCESS_FINE_LOCATION) != PackageManager.
 PERMISSION_GRANTED) {
  ActivityCompat.requestPermissions(gps.this, new String[]{
    Manifest.permission.ACCESS_FINE_LOCATION
  }, 1000);
}
```

Listing 7. GPS - Dostęp do lokazlizacji

Metoda onLocationChanged przedstawiona jako listing 8 (s. 16) odpowiada za wyświetlenie powiadomienia push-up wyświetlającego długość i szerokość geograficzną obecnej lokalizacji oraz pobiera adres tej lokalizacji na podstawie długości i szerokości geograficznej, który wyświetla jako tekst.

```
1  @Override
2  public void onLocationChanged(Location location) {
3    Toast.makeText(this, ""+location.getLatitude()+", "+location.
     getLongitude(), Toast.LENGTH_SHORT).show();
4    try {
5      Geocoder geocoder = new Geocoder(gps.this, Locale.getDefault
     ());
6      List<Address> addresses = geocoder.getFromLocation(location.
     getLatitude(), location.getLongitude(), 1);
7      String address = addresses.get(0).getAddressLine(0);
8      text_location.setText(address);
9    } catch (Exception e) {
10     e.printStackTrace();
11   }
12 }
```

Listing 8. GPS - Wyświetlanie lokalizacji

Listing 9 (s. 16) przedstawia implementacje MediaPlayer (linijka 1) do którego podłączamy dźwięk. Teraz w prosty sposób możemy wywołać dźwięk poprzez medote OnClick,

```
1  final MediaPlayer sound = MediaPlayer.create(this, R.raw.
     dzwiek_audio);
2  Button btn_dzwiek = findViewById(R.id.btn_dzwiek);
3  btn_dzwiek.setOnClickListener(new View.OnClickListener() {
4    @Override
5    public void onClick(View view) {
6      sound.start();
7    }
8  });
```

Listing 9. Dźwięk - Działanie z wykorzystaniem MediaPlayer

Aby test mikrofonu spełniał swoje zadanie trzeba sprawdzić autoryzację aplikacji. W tym wypadku uzyskujemy pozwolenie na nagrywanie audio, co przedstawia listing 10 (s. 17).

```
1  private void getMicrophonePermission() {
2    if(ContextCompat.checkSelfPermission(this, Manifest.permission.
   RECORD_AUDIO) == PackageManager.PERMISSION_DENIED) {
3      ActivityCompat.requestPermissions(this, new String[] {
   Manifest.permission.RECORD_AUDIO}, MICROPHONE_PERMISSION_CODE);
4    }
5  }
```

Listing 10. Mikrofon - Dostęp do nagrywania audio

Listing 11 (s. 17) przedstawia metodę z wykorzystaniem MediaRecorder który służy do nagrywania dźwięku lub obrazu. W tym wypadku MediaRecorder wykorzystany zostanie do nagrania dźwięku przez mikrofon w telefonie. Od 4 do 7 linijki określamy źródło dźwięku, format wyjściowy, zaznaczenie deskryptora pliku, definiowanie kodowania dźwięku.

```
1  public void btnRecordPressed(View v) {
2    try {
3      mediaRecorder = new MediaRecorder();
4      mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
5      mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.
   THREE_GPP);
6      mediaRecorder.setOutputFile(getRecordingFilePath());
7      mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.
   AMR_NB);
8      mediaRecorder.prepare();
9      mediaRecorder.start();
10
11     Toast.makeText(this, "Nagrywanie rozpoczete", Toast.
   LENGTH_LONG).show();
12   }
13   catch (Exception e) {
14     e.printStackTrace();
15 }
```

Listing 11. Mikrofon - Włączenie nagrywania

Po naciśnięciu przycisku "Stop"nagranie zostaje zakończone, co obrazuje listing 12 (s. 18). MediaRecorder kończy nagrywanie, a opcja realse użyta w 3 linijce nie pozwoli na ponowne uruchomienie pliku dźwiękowego w celu dogrania do niego dźwięku.

```
1  public void btnStopPressed(View v) {
2    mediaRecorder.stop();
3    mediaRecorder.release();
4    mediaRecorder = null;
5
6    Toast.makeText(this, "Nagrywanie zakonczone", Toast.LENGTH_LONG
     ).show();
7  }
```

Listing 12. Mikrofon - Przerwanie nagrywania

Obiekt MediaPlayer wykorzystany w listingu 13 (s. 18) służy do obsługi odtwarzania plików audio i wideo. W 4 linijce kodu określamy skąd ma zostać pobrany plik, a następnie plik z dźwiękiem zostanie włączony.

```
1  public void btnPlayPressed(View v) {
2    try {
3      mediaPlayer = new MediaPlayer();
4      mediaPlayer.setDataSource(getRecordingFilePath());
5      mediaPlayer.prepare();
6      mediaPlayer.start();
7
8      Toast.makeText(this, "Odtwarzanie nagrania", Toast.
     LENGTH_LONG).show();
9    }
10   catch(Exception e){
11     e.printStackTrace();
12   }
13 }
```

Listing 13. Mikrofon - Odtworzenie nagrania

Metoda przedstawiona na listingu 14 (s. 19), odpowiada za uruchomienie aparatu w telefonie po naciśnięciu przycisku o id btnCam. Od 6 do 8 linijki znajduję się kod opowiedzialny za uruchomienie aplikacji aparatu.

```
1  btnCam = findViewById(R.id.btnCam);
2  btnCam.setOnClickListener(new View.OnClickListener() {
3    @Override
4    public void onClick(View view) {
5      try {
6        Intent intent = new Intent();
7        intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
8        startActivity(intent);
9      } catch (Exception e) {
10       e.printStackTrace();
11     }
12   }
13 });
```

Listing 14. Aparat - Włączenie aparatu

Pętla switch (rozpoczynająca się w linijce 6) przedstawiona w listingu 15 (s. 19) odpowiada za informowanie użytkowanika o aktualnym statusie wifi. W linijce 7 sprawdzamy czy Wifi jest włączone, jeśli jest włączone - wyświetlamy odpowiedni tekst, w 10 linijce sprawdzamy czy wifi jest wyłączone na danym urządzeniu, jeśli tak zostaje wyświetlony adekwatny tekst.

```
1  private BroadcastReceiver wifiStateReceiver = new
    BroadcastReceiver() {
2    @Override
3    public void onReceive(Context context, Intent intent) {
4      int wifiStateExtra = intent.getIntExtra(WifiManager.
   EXTRA_WIFI_STATE,
5      WifiManager.WIFI_STATE_UNKNOWN);
6      switch (wifiStateExtra) {
7        case WifiManager.WIFI_STATE_ENABLED:
8        wifiSwitch.setText("WiFi jest wlaczone, mozna pobrac
   informacje");
9        break;
10       case WifiManager.WIFI_STATE_DISABLED:
11       wifiSwitch.setText("WiFi jest wylaczone, wlacz aby pobrac
   informacje");
12       break;
13     }
14   }
15 };
```

Listing 15. Wifi - Sprawdzenie statusu Wifi

Metoda przedstawiona na listingu 16 (s. 20) odpowiada za pobranie informacji o sieci wifi do której urządzenie jest obecnie podłączone oraz wyświetlenia tych informacji. Od 5 do 9 linijki mamy kolejno pobranie informacji o adresie IP telefonu, sformatowanie pobranego IP ma tekst, pobranie adresu MAC routera, SSID oraz wskaźnika mocy naszego połączenia. Pobrane informacje implementujmey jako string (od 11 do 15 linijki) i wyświetlamy jako tekst (linijka 17).

```
1     public void getWifiInformation(View view) {
2   WifiManager wifiManager = (WifiManager) getApplicationContext().
     getSystemService(WIFI_SERVICE);
3   WifiInfo wifiInfo = wifiManager.getConnectionInfo();
4
5   int ip = wifiInfo.getIpAddress();
6   String ipAddress = Formatter.formatIpAddress(ip);
7   String bssid = wifiInfo.getBSSID();
8   String ssid = wifiInfo.getSSID();
9   int rssi = wifiInfo.getRssi();
10
11  String info =
12  "\n Adres IP: " + ipAddress +
13  "\n Adres MAC Routera: " + bssid +
14  "\n SSID: " + ssid +
15  "\n Wskaznik mocy: " + rssi;
16  txtWifiInfo.setText(info);
17 }
```

Listing 16. Wifi - Pobieranie informacji o Wifi

Kod ukazany w listingu 17 (s. 20) jest odpowiedzialny za pobranie marki i modeu urządzenia z którego obecnie korzystamy i wyświetlenie tej informacji jako tekst.

```
1   model = (TextView) findViewById(R.id.model);
2   String stringBuildModel = "Marka i model: " + Build.MANUFACTURER
     + " " + Build.MODEL;
3   model.setText(stringBuildModel);
```

Listing 17. Wyniki - Pobranie marki i modelu urządzenia

5. Testowanie

5.1. Testowanie latarki.

Tab. 5.1. Testowanie latarki

lp	Zadania do przetestowania	Tak	Nie
1	Latarka po naciśnięciu na guzik włączyła się	X	
2	Po włączeniu latarki, guzik zmienia kolor na zielony	X	
3	Latarka po naciśnięciu na guzik wyłączyła się	X	
4	Po wyłączeniu latarki, guzik zmienia kolor na czerwony	X	

Obrazek 5.1 przedstawia zrzuty ekranu potwierdzające pomyślny przebieg testu.

Rys. 5.1. Przebieg testowania latarki

6. Podręcznik użytkownika

Spis rysunków

Spis tabel

Spis listingów