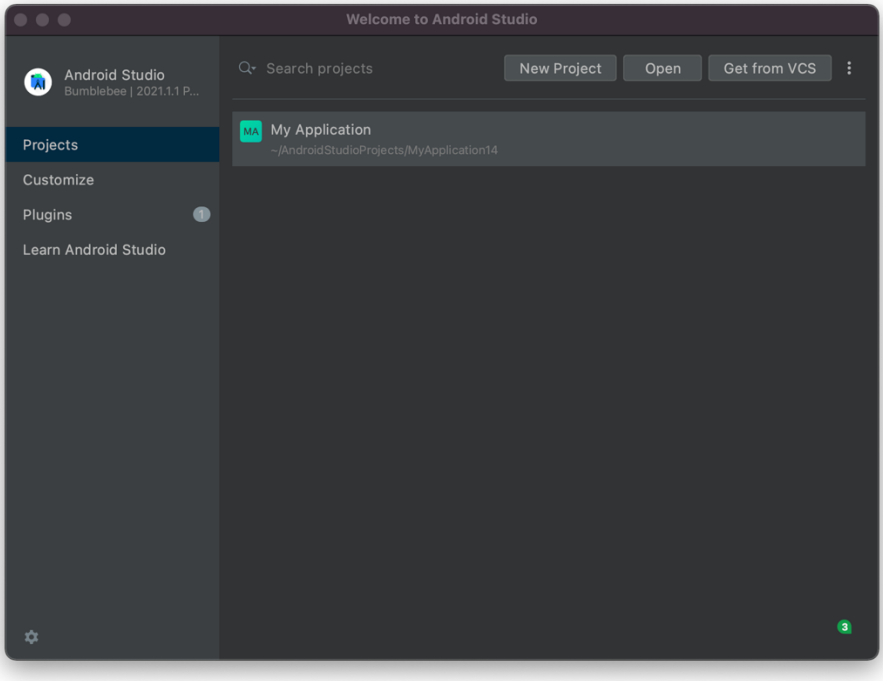
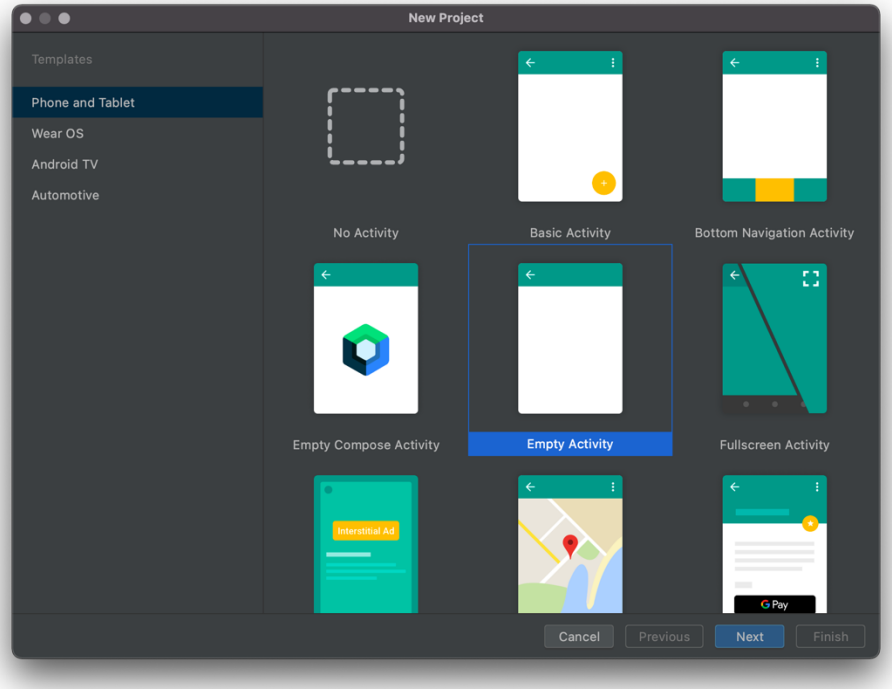
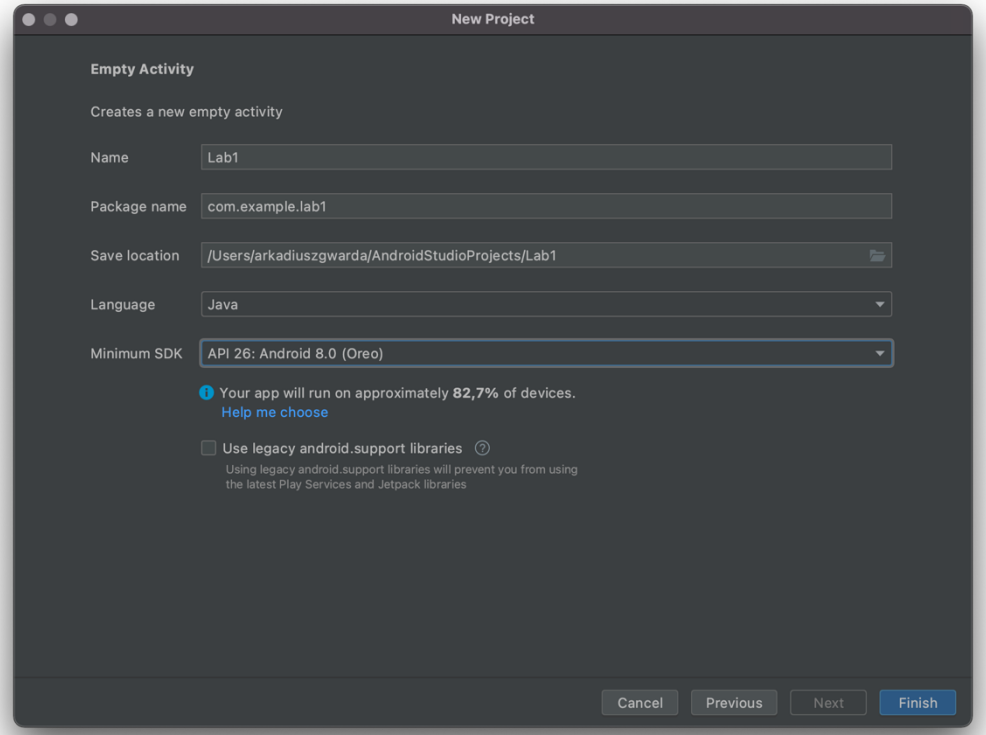
Zacznijmy od zaprojektowania prostej aplikacji wyświetlającej tekst na ekranie głównej aktywności aplikacji. Po otworzeniu Android Studio otrzymamy okno:



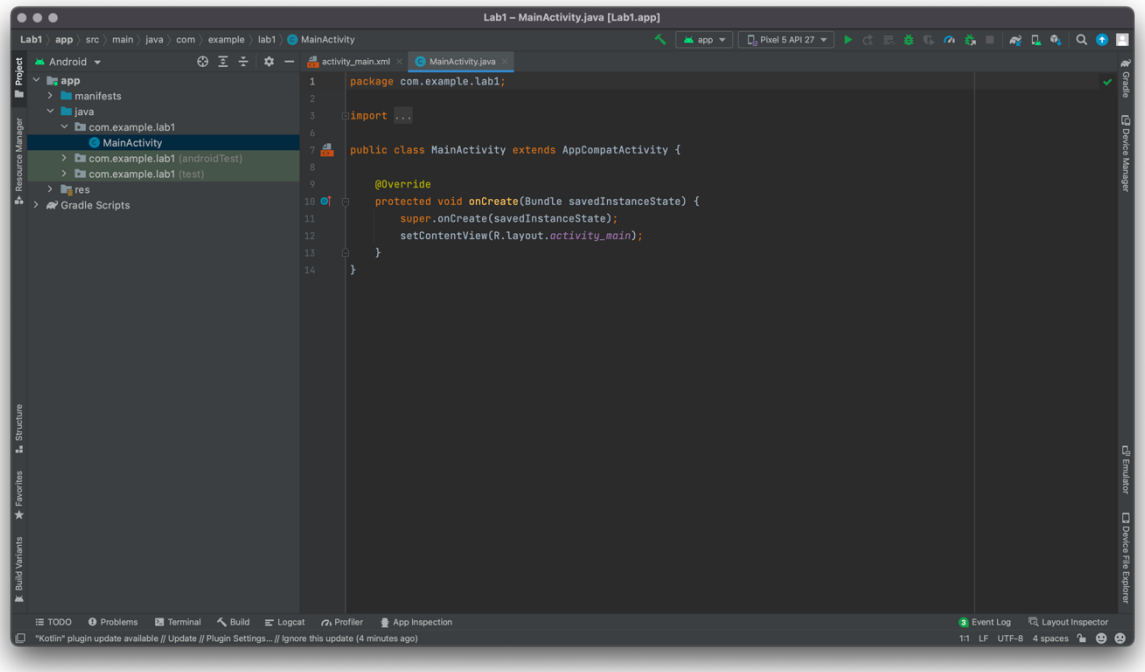
Wybieramy New Project, a na kolejnym oknie wybieramy pustą aktywność (Empty Avtivity)



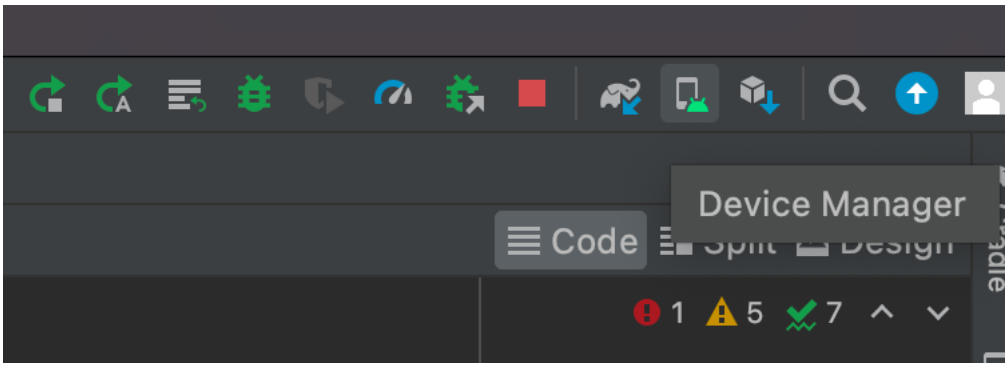
Następnie nadajemy naszej aplikacji nazwę, wybieramy wersję systemu Android, dla którego nasza aplikacja będzie z pewnością działać prawidłowo, oraz koniecznie wybieramy język programowania, czyli Java:



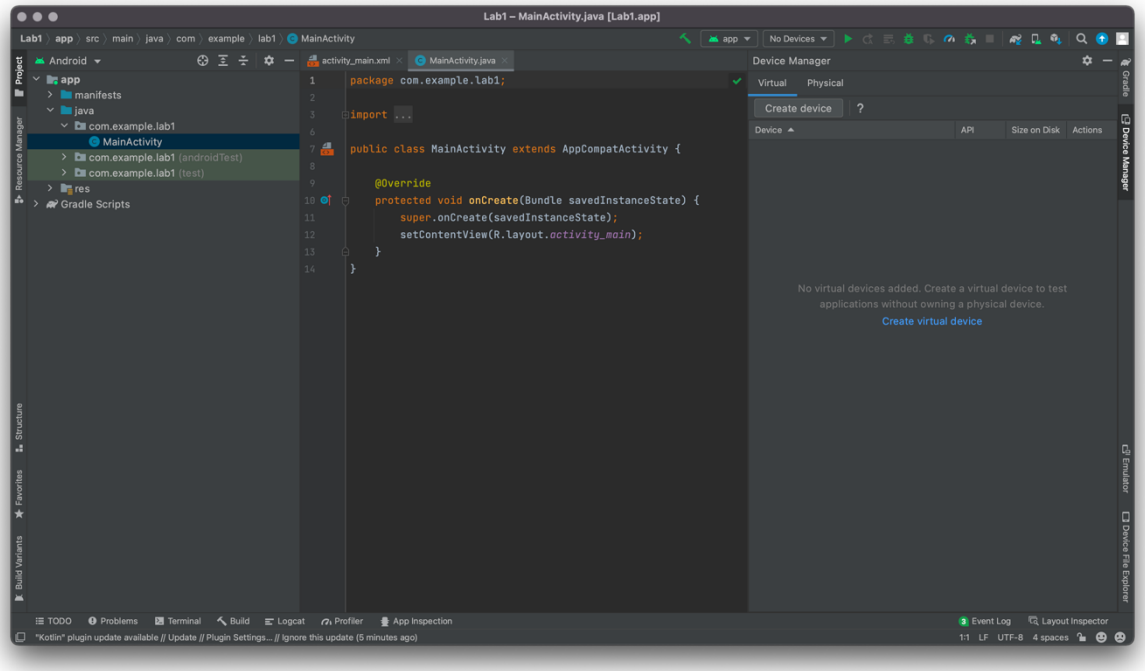
Klikamy Finish i czekamy aż Android Studio skonfiguruje projekt. Przy pierwszym uruchomieniu Android studio zaraz po instalacji może to potrwać dłuższą chwilę. Gdy nasza aplikacja będzie gotowa, powinniśmy zobaczyć okno z kodem aktywności głównej oraz drzewo projektu po lewej stronie:



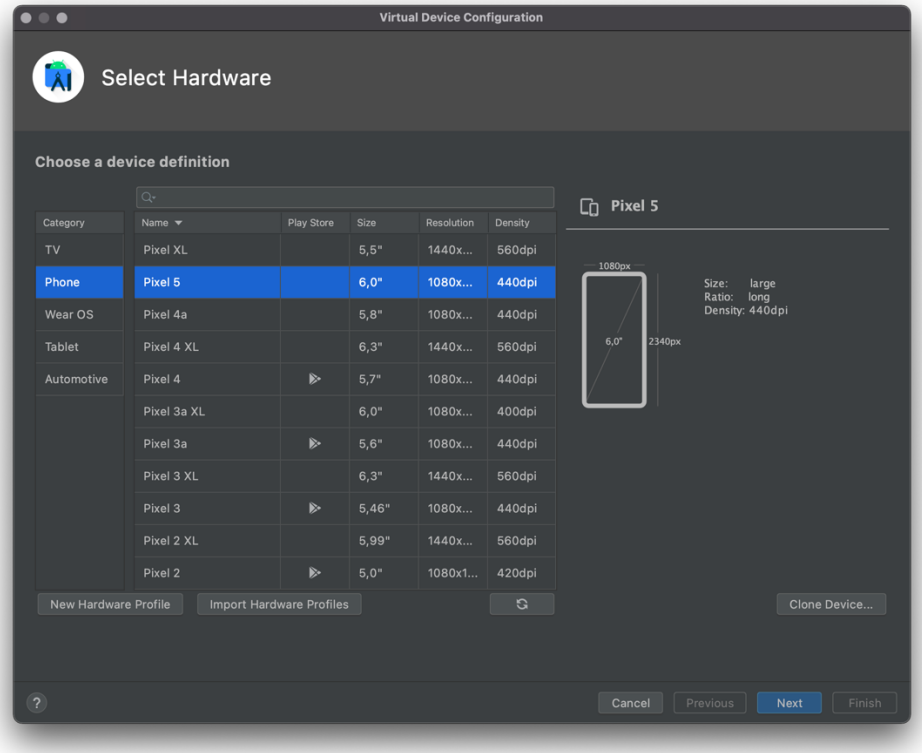
Jest to nasza pierwsza działająca aplikacja, żeby to sprawdzić potrzebujemy jednak urządzenia. Android Studio udostępnia Emulator, dzięki któremu możemy przetestować naszą aplikację. Aby skorzystać z emulatora, musimy dodać nowe urządzenie, w tym celu uruchamiamy Device Menager (w starszych wersjach AVD Menager):



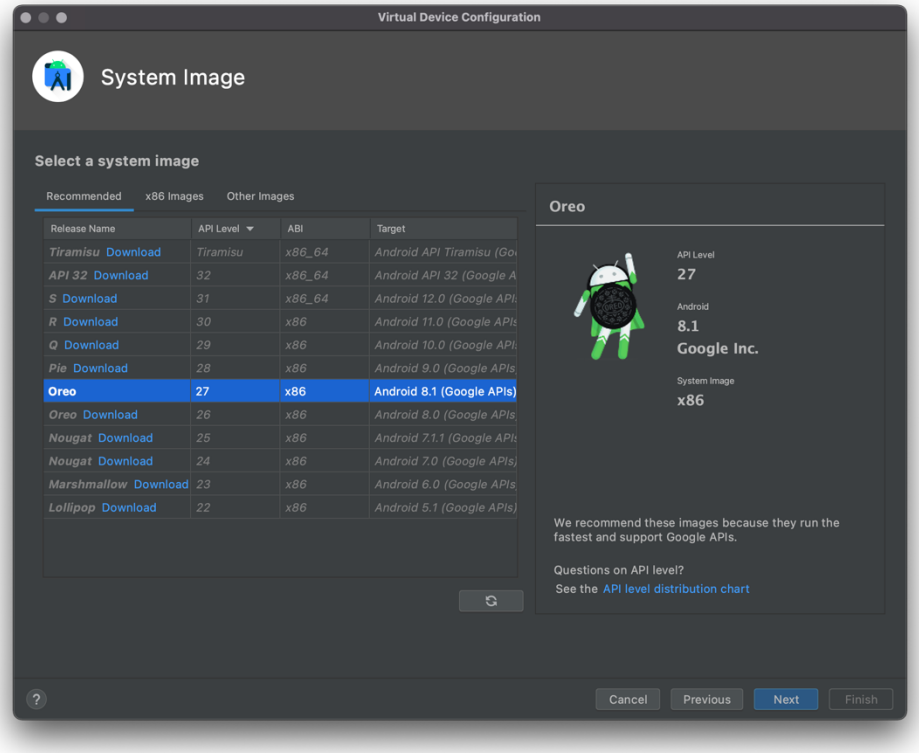
Po prawej stronie wyświetli nam się panel Menagera, na którym klikamy Create device:



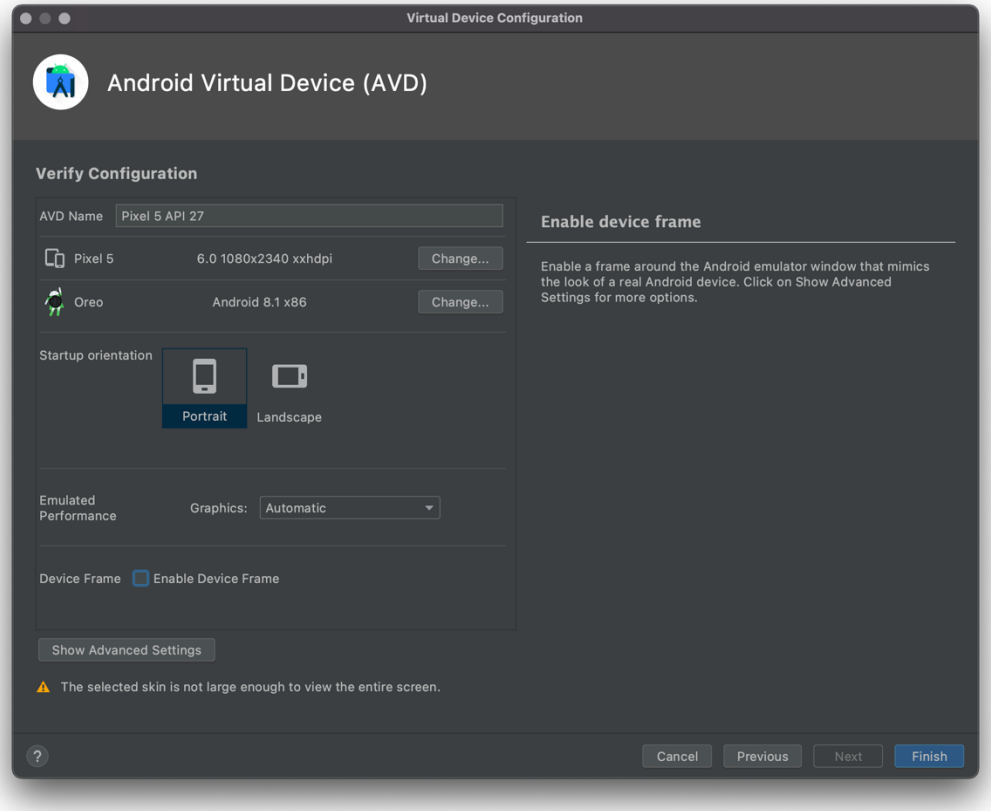
Pierwsze okno to wybór urządzenia, ja wybrałem Google Pixel 5 i nie zmieniałem niczego w ustawieniach pamięci. Przy bardziej wymagających projektach i wydajnym komputerze, można pozwolić sobie na zwiększenie parametrów wirtualnego urządzenia.



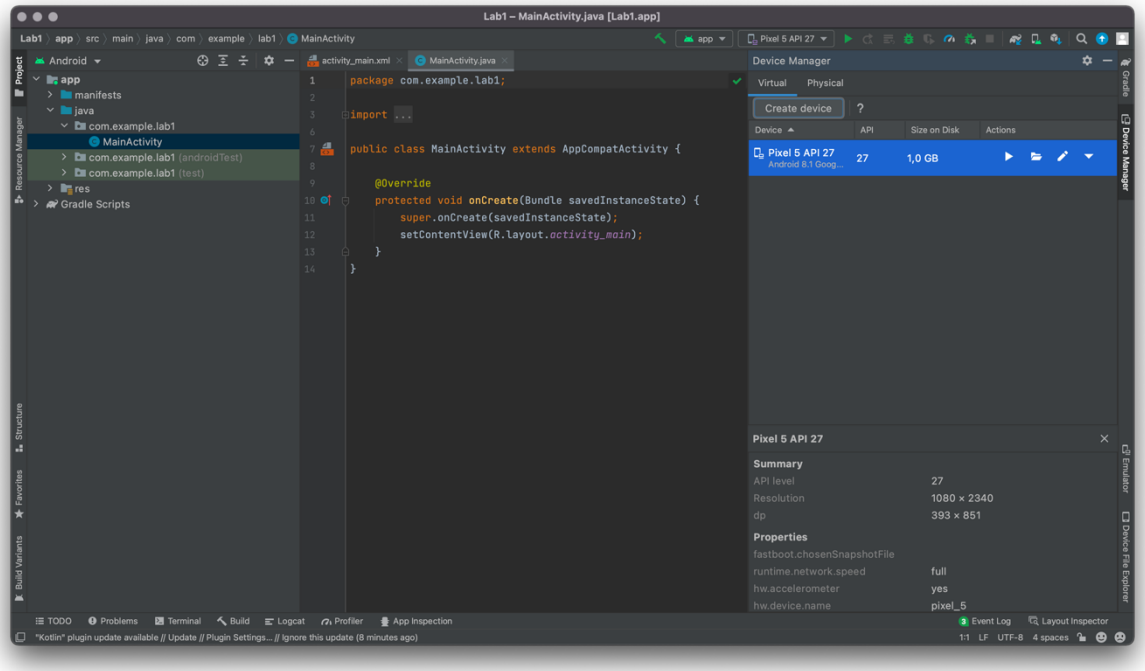
Dalej wybieramy obraz systemu naszego urządzenia, ja wybrałem Oreo, trzeba pobrać obraz systemu.



Po wciśnięciu Next widzimy okno podsumowania i wyboru orientacji. Nie zmieniamy nic, klikamy Finish.

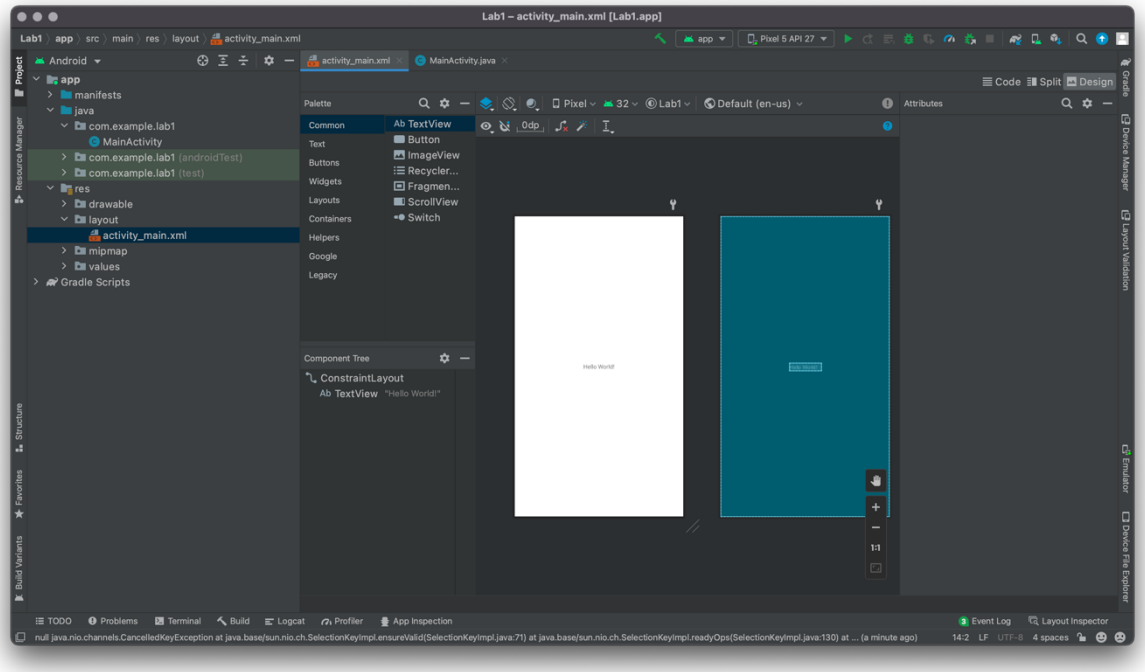


Teraz nasze urządzenie jest dostępne:



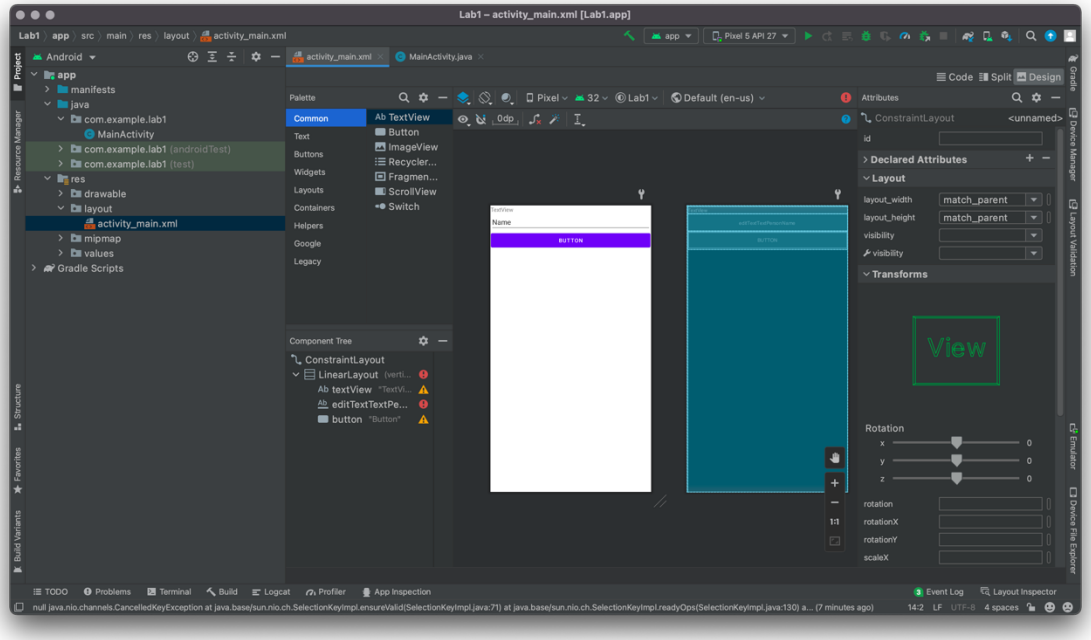
Wracamy do tworzenia naszej aplikacji.

Zaprojektujemy interfejs naszej aktywności głównej, tym celu przechodzimy do folderu res i pliku activity_main.xml:



Widzimy podstawowy widok Design oraz przykładowy interfejs z napisem Hello world. Klikamy na napis i usuwamy go naciskają delete. Projektujemy swój interfejs, z zakładki Layouts przeciągamy i upuszczamy na interfejs LinearLayout (vertical), następnie to samo robimy z następującymi komponentami (kolejność dokładnie taka sama):
TextView
PlainText
Button.

Powinno to wyglądać tak:



Dalej przechodzimy do widoku kodu (Code, lub Split), a kod naszego interfejsu powinien wyglądać następująco:

activity_main.xml MainActivity.java

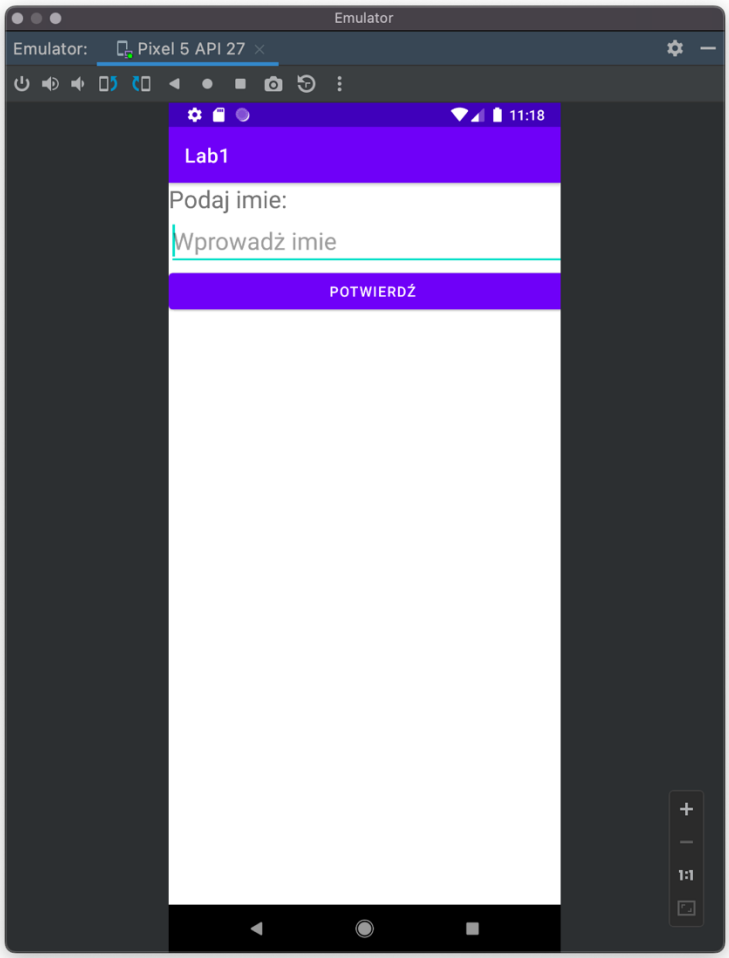
```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="409dp"
        android:layout_height="729dp"
        android:orientation="vertical"
        tools:layout_editor_absoluteX="1dp"
        tools:layout_editor_absoluteY="1dp">

        <TextView
            android:id="@+id/textView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="TextView" />

        <EditText
            android:id="@+id/editTextTextPersonName"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName"
            android:text="Name" />

        <Button
            android:id="@+id/button1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Button" />
    </LinearLayout>
```

Trochę go modyfikujemy, zmieniając kilka pól i dodając nowe:

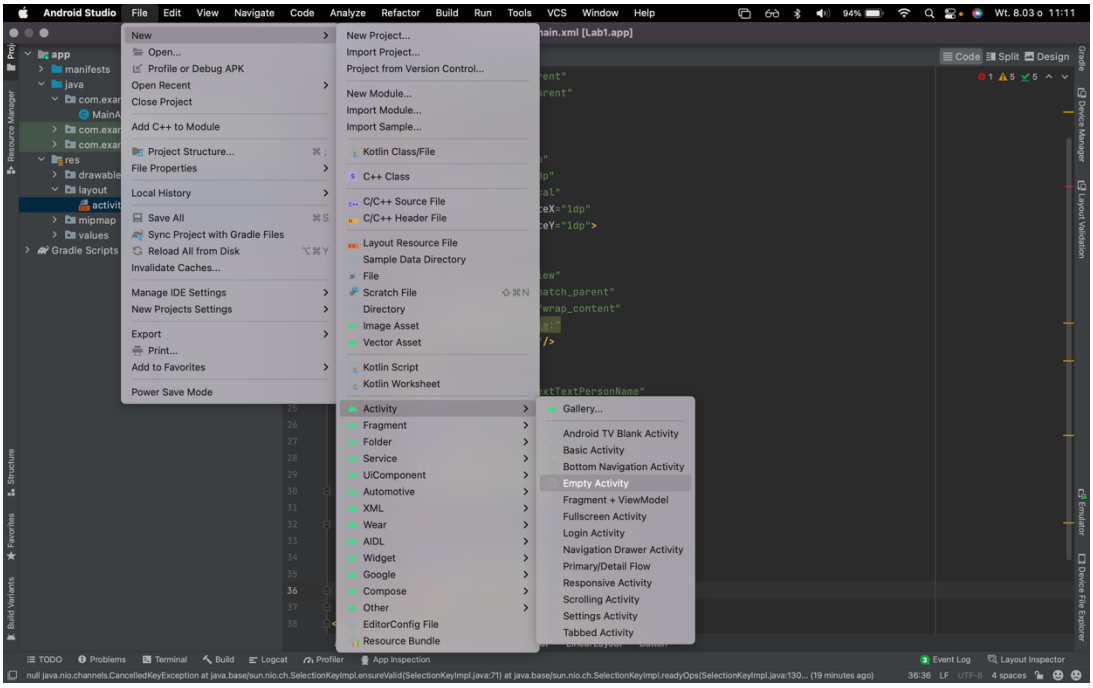
activity_main.xml MainActivity.java

```
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       tools:context=".MainActivity">
8
9       <LinearLayout
10          android:layout_width="409dp"
11          android:layout_height="729dp"
12          android:orientation="vertical"
13          tools:layout_editor_absoluteX="1dp"
14          tools:layout_editor_absoluteY="1dp">
15
16          <TextView
17              android:id="@+id/textView"
18              android:layout_width="match_parent"
19              android:layout_height="wrap_content"
20              android:text="Podaj imie:"
21              android:textSize="24sp"/>
22
23          <EditText
24              android:id="@+id/editTextTextPersonName"
25              android:layout_width="match_parent"
26              android:layout_height="wrap_content"
27              android:ems="10"
28              android:inputType="textPersonName"
29              android:hint="Wprowadź imie"
30              android:textSize="24sp"/>
31
32          <Button
33              android:id="@+id/button1"
34              android:layout_width="match_parent"
35              android:layout_height="wrap_content"
36              android:text="Potwierdź" />
37      </LinearLayout>
```

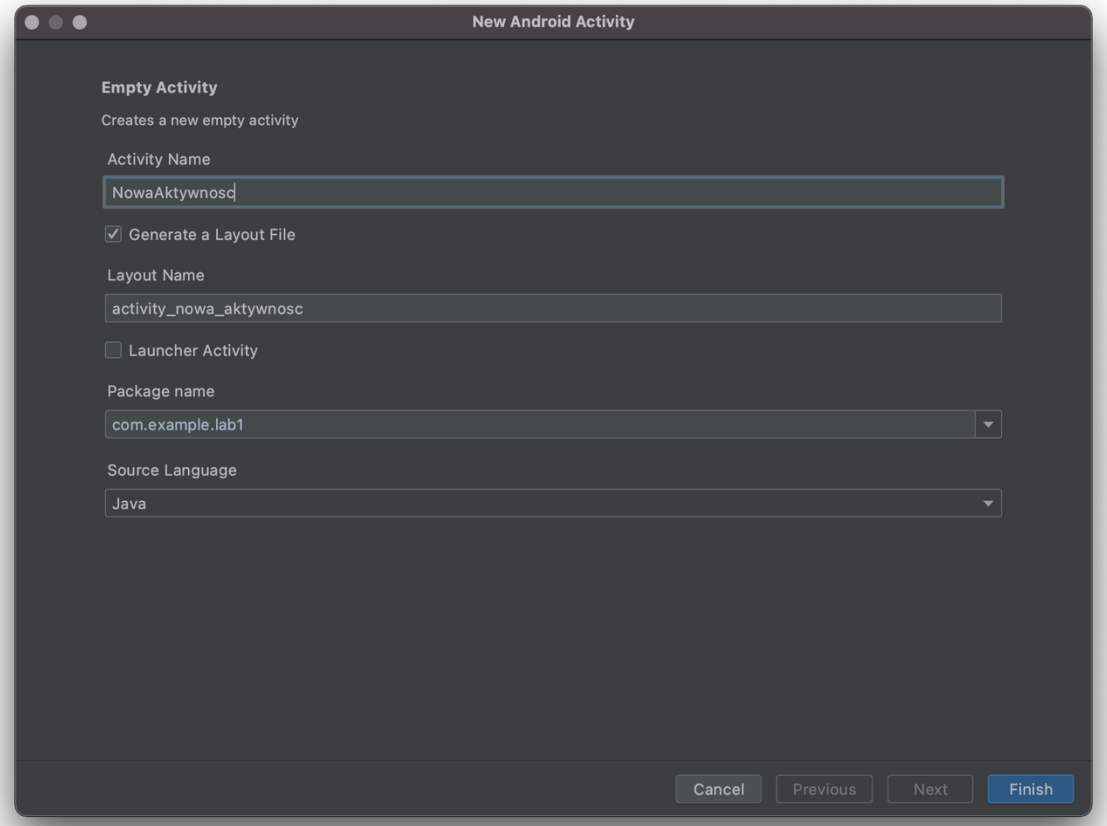
Pora uruchomić aplikację. Jeśli wszystko zrobiliśmy poprawnie, powinna po uruchomieniu wyglądać tak:



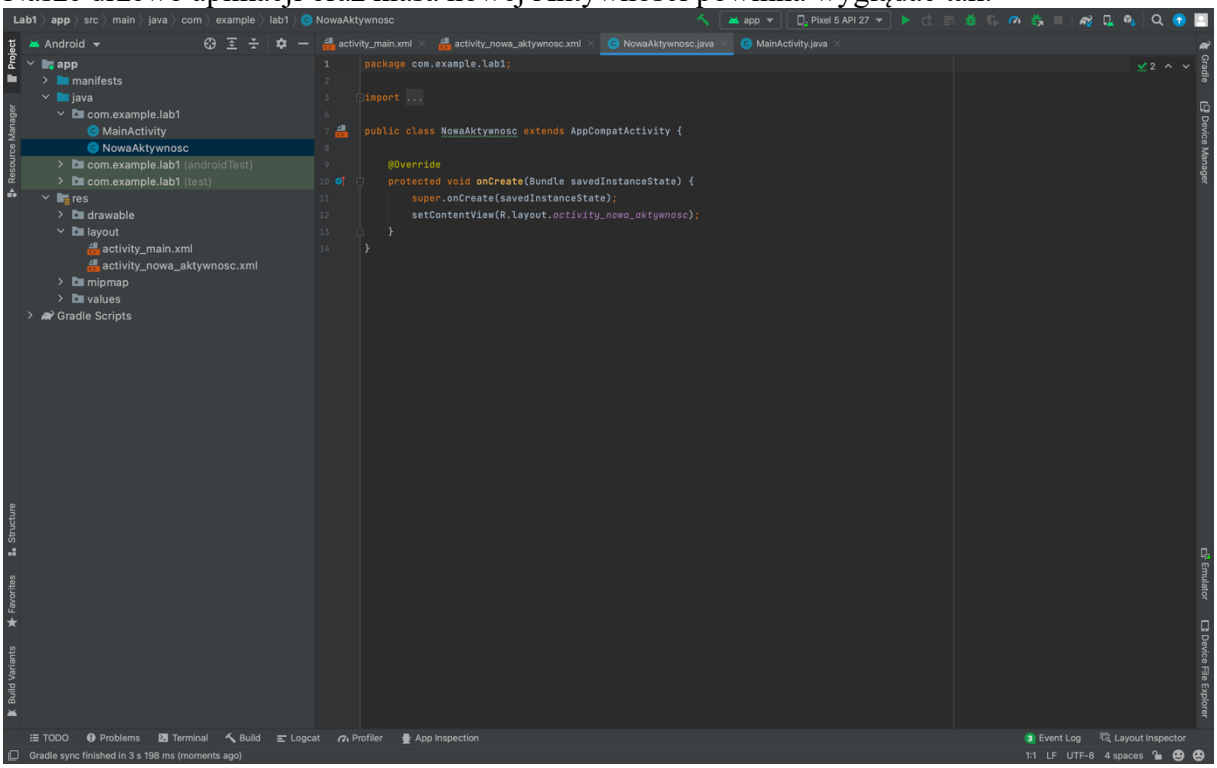
Czas stworzyć nową aktywność wybierając File → New → Activity → Empty Activity.



Nadajemy nazwę dla nowej aktywności i klikamy Finish.



Nasze drzewo aplikacji oraz klasa nowej Aktywności powinna wyglądac tak:



Aby wywołać nową aktywność na pierwszy plan, musimy użyć jakiegoś komponentu. Oczywisty wydaje się wybór komponentu Button z MainActivity. Najpierw modyfikujemy kod klasy MainActivity:

activity_main.xml | activity_nowa_aktywnosc.xml | NowaAktywnosc.java | MainActivity.java

```java
package com.example.lab1;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void nowaAktywnosc(View view){
        Intent intent = new Intent( packageContext: this, NowaAktywnosc.class);
        startActivity(intent);
    }
}
```

Utworzyliśmy nową metodą, która tworzy obiekt intencji (Intent). Intencja pozwala uruchamiać nowe aktywności oraz umożliwia komunikacje między nimi. Ostatnim krokiem jest podpięcie wywołania metody nowaAktywnosc do komponentu Button:

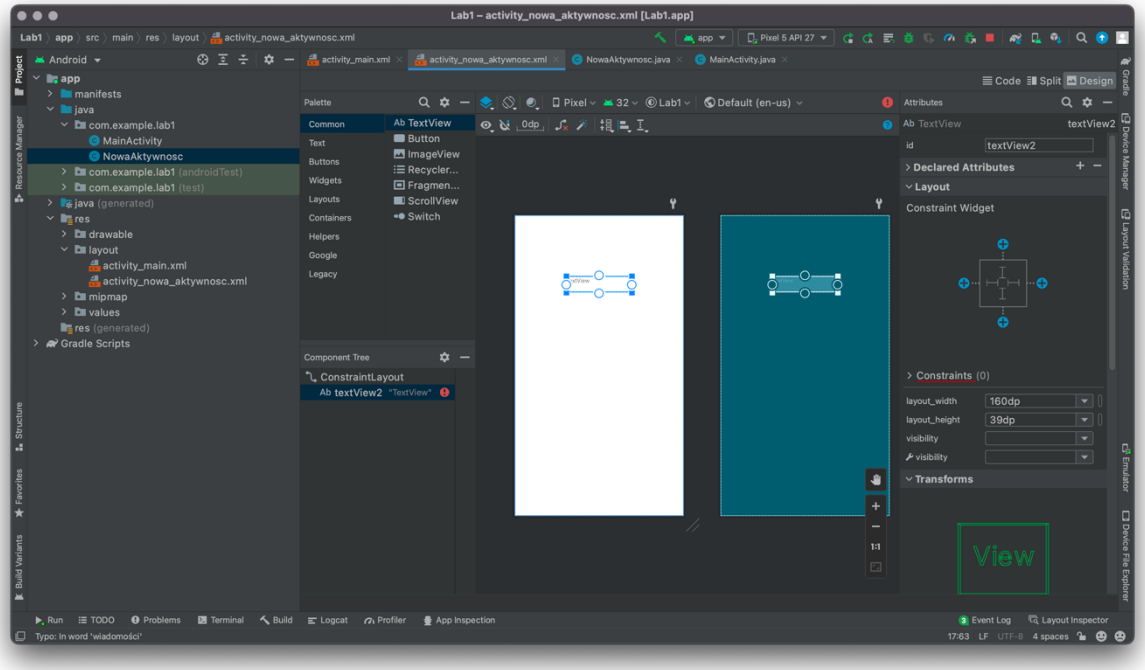
```
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Potwierdź"
        android:onClick="nowaAktywnosc"/>
</LinearLayout>
```

Można uruchomić naszą aplikację. Po naciśnięciu przycisku „Potwierdź” wyświetlimy pustą aktywność. Zmodyfikujmy ją tak, aby wyświetlała tekst wpisany w głównej aktywności. W tym celu musimy wysłać wiadomość jaką będzie wpisany łańcuch znaków do nowo wywołanej aktywności. Skorzystamy z intencji:

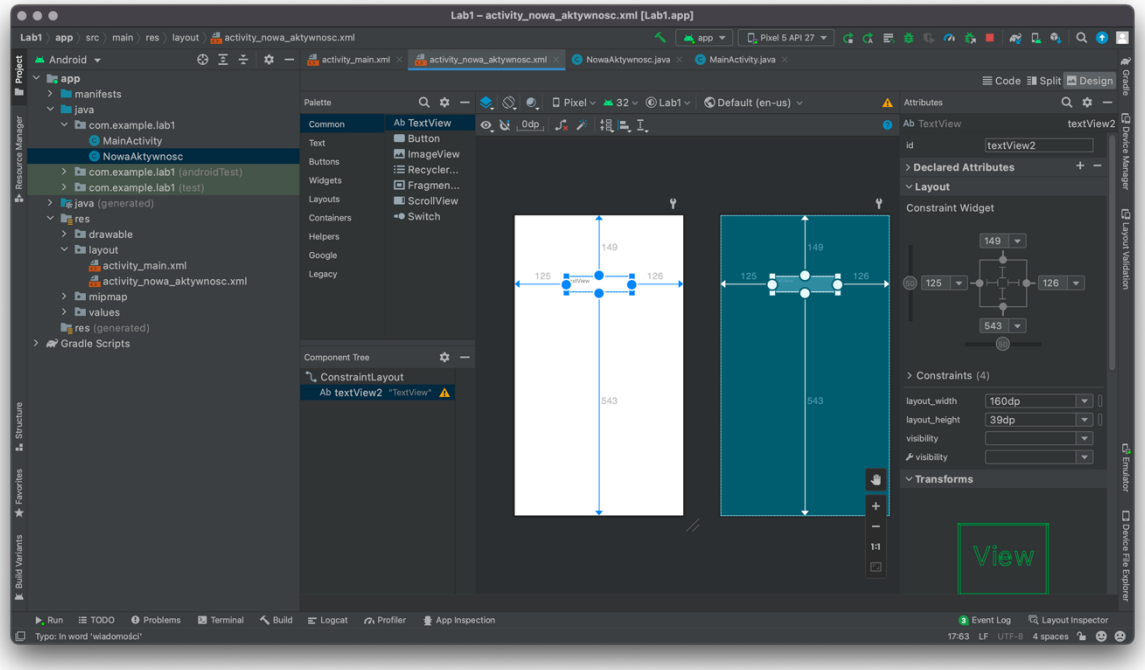
activity_main.xml | activity_nowa_aktywnosc.xml | NowaAktywnosc.java | MainActivity.java

```
1   package com.example.lab1;
2
3   import ...
9
10  public class MainActivity extends AppCompatActivity {
11
12      public final static String klucz = "com.example.lab1";
13      //klucz, którego musimy użyć aby indentyfikować wiadomość
14
15      @Override
16      protected void onCreate(Bundle savedInstanceState) {
17          super.onCreate(savedInstanceState);
18          setContentView(R.layout.activity_main);
19      }
20
21      public void nowaAktywnosc(View view){
22          Intent intent = new Intent( packageContext: this, NowaAktywnosc.class);
23
24          EditText editText = (EditText) findViewById(R.id.editTextTextPersonName);
25          //tworzymy obiekt EditText i łączymy go z id komponentu
26          String message = editText.getText().toString();
27          //Tworzymy łańcuch znaków, do którego pobieramy wiadomość
28          //wpisaną w EditText
29          intent.putExtra(klucz, message);
30          //wrzucamy do intencji dodatkową wiadomość do przekazania,
31          //razem z kluczem
32
33          startActivity(intent);
34      }
35  }
```

W nowej aktywności modyfikujemy jej wygląd. Dodajemy komponent TextView:



TextView jest wyciągnięte na płaszczyznę bez layoutu, który by automatycznie układał komponenty, domyślnie jest to ConstraitLayout, który wymaga zakotwiczenia elementu. Najłatwiej zrobić to za pomocą nawigatora po prawej stronie, trzeba wcisnąć wszystkie cztery „plusiki”.



Teraz musimy odebrać naszą wiadomość, a tekst przekazać do komponentu TextView.

Lab1 – NowaAktywnosc.java [Lab1.app]

NowaAktywnosc | app | Pixel 5 API 27

activity_main.xml | activity_nowa_aktywnosc.xml | NowaAktywnosc.java | MainActivity.java

```java
package com.example.lab1;

import ...

public class NowaAktywnosc extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_nowa_aktywnosc);

        Intent intent = getIntent();
        String message = intent.getStringExtra(MainActivity.klucz);
        //Stworzenie obiektu Intent i przechwycenie wiadomości

        TextView textView = (TextView) findViewById(R.id.textView2);
        textView.setText(message);
        //przekazanie napisu do pola textView
    }
}
```

Możemy uruchomić naszą aplikację i przetestować jej działanie.

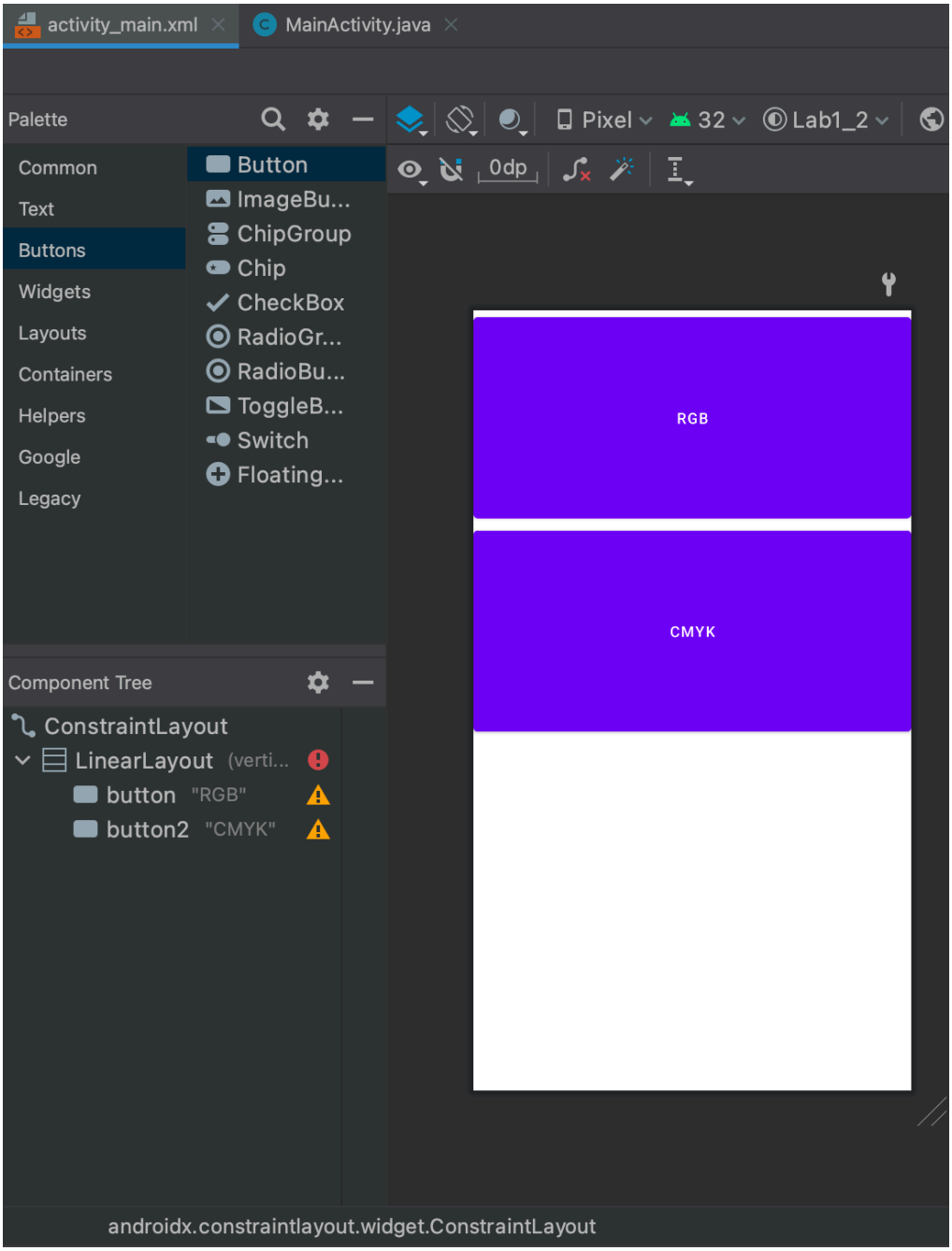
Zadania:

1. Stwórz aplikację mającą kilka aktywności (oprócz głównej) uruchamiane za pomocą przycisku. Niech aplikacja będzie mini encyklopedią opisującą różne typy danych w języku Java, gdzie po naciśnięciu danego typu uzyskamy krótki opis w nowej aktywności.
2. Stwórz aplikację, w której po wpisaniu liczby, w nowych (osobnych) aktywnościach wyświetlany jest jej kwadrat oraz sześcian.

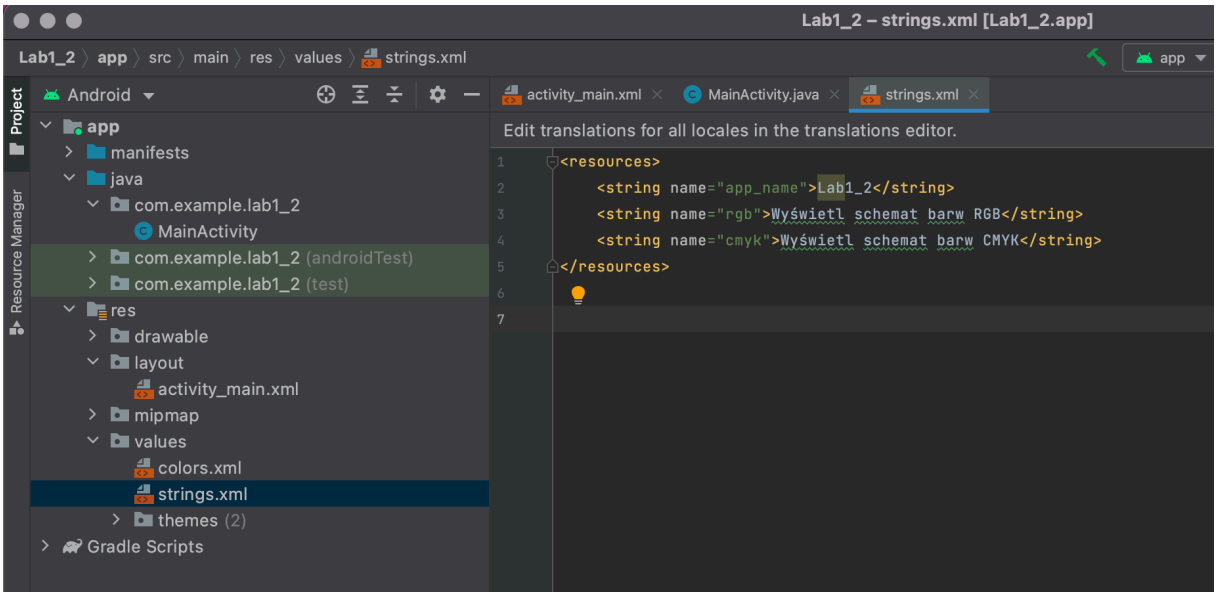
Część II

Folder res zawiera wiele plików odpowiedzialnych za wygląd naszej aplikacji. Jeśli wykorzystujemy wiele języków lub wielokrotnie wykorzystujemy te same łańcuchy znaków, przydatne może być wykorzystanie pliku „strings.xml”. Plik ten przechowuje łańcuchy znaków skojarzone ze swoim ID.

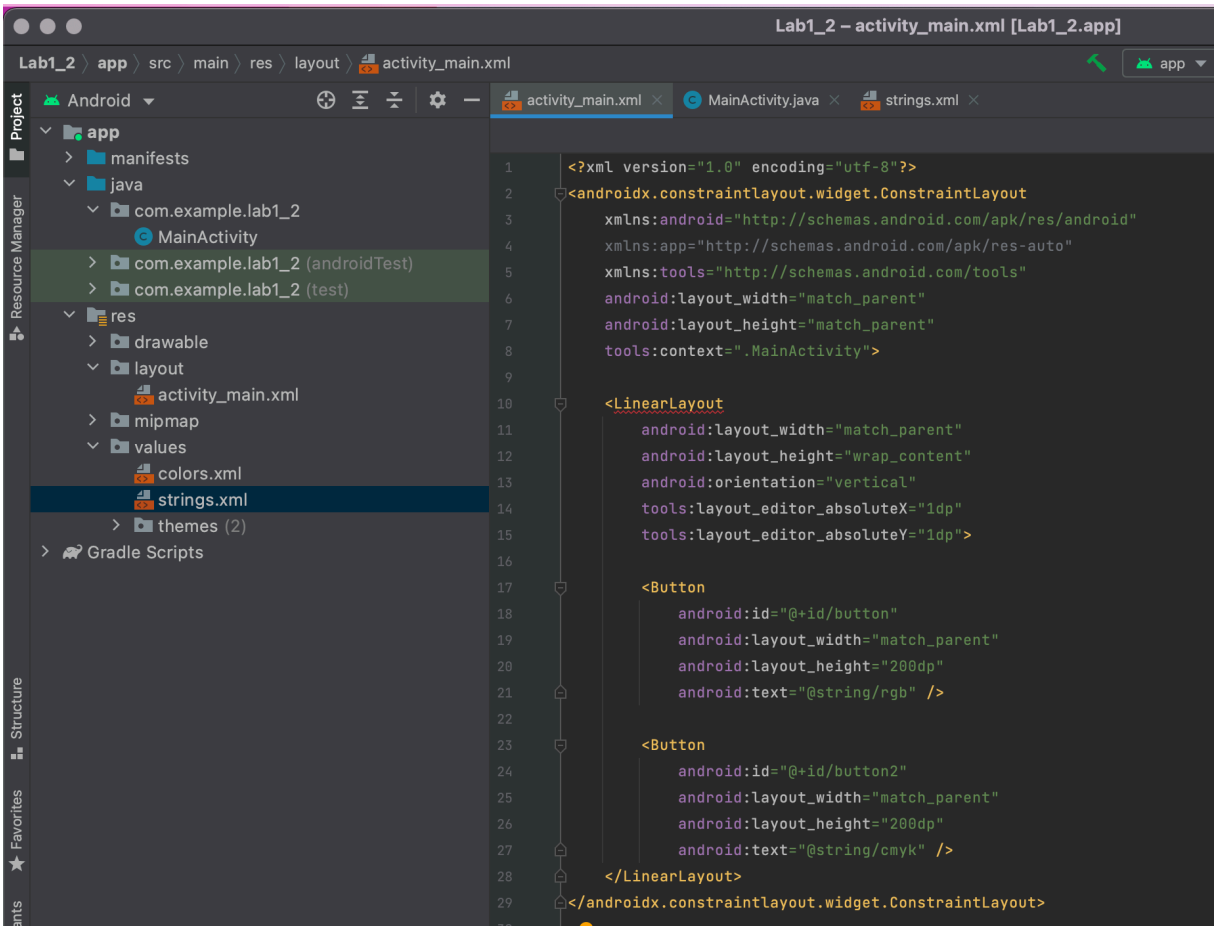
Stwórzmy nową aplikację, interfejs aktywności główneu powinien wyglądać tak:



Zaglądamy do pliku strings.xml, zmodyfikujemy napisy na guzikach za pomocą napisów zdefiniowanych wcześniej w tym pliku:



Następnie zmieniamy main_activity.xml zmieniając napisy na guzikach:



Tworzymy nową aktywność i podpinamy ją pod guzik RGB, wykorzystujemy kilka dodatkowych atrybutów komponentu TextView:

android:background="(kod koloru)"
android:textColor="(kod koloru)"
android:gravity="center"

```
16  <TextView
17      android:id="@+id/textView"
18      android:layout_width="wrap_content"
19      android:layout_height="match_parent"
20      android:layout_weight="1"
21      android:text="Red"
22      android:gravity="center"
23      android:textColor="#AA4040"
24      android:background="#FF0400"/>
25
26  <TextView
27      android:id="@+id/textView2"
28      android:layout_width="wrap_content"
29      android:layout_height="match_parent"
30      android:layout_weight="1"
31      android:text="Blue"
32      android:gravity="center"
33      android:textColor="#2E49AC"
34      android:background="#005EFF"/>
35
36  <TextView
37      android:id="@+id/textView3"
38      android:layout_width="wrap_content"
39      android:layout_height="match_parent"
40      android:layout_weight="1"
41      android:text="Green"
42      android:gravity="center"
43      android:textColor="#4CAF50"
44      android:background="#8BC34A"/>
```

Zadanie:

Wykonaj te same kroki by stworzyć aktywność przedstawiającą schemat CMYK, powinna się uruchamiać po naciścięniu drugiego przycisku.