

# Bingo w Android Studio

Aleksander Brzykcy, Wojciech Barnaś

# Android Studio

- Oficjalne środowisko programistyczne na platformę Android zbudowane na podstawie oprogramowania IntelliJ IDEA od JetBrains
- Może zostać zainstalowane na systemach Windows, macOS i opartych na Linuxie
- Przed wydaniem Android Studio, głównym środowiskiem programistycznym dla systemu Android było Eclipse



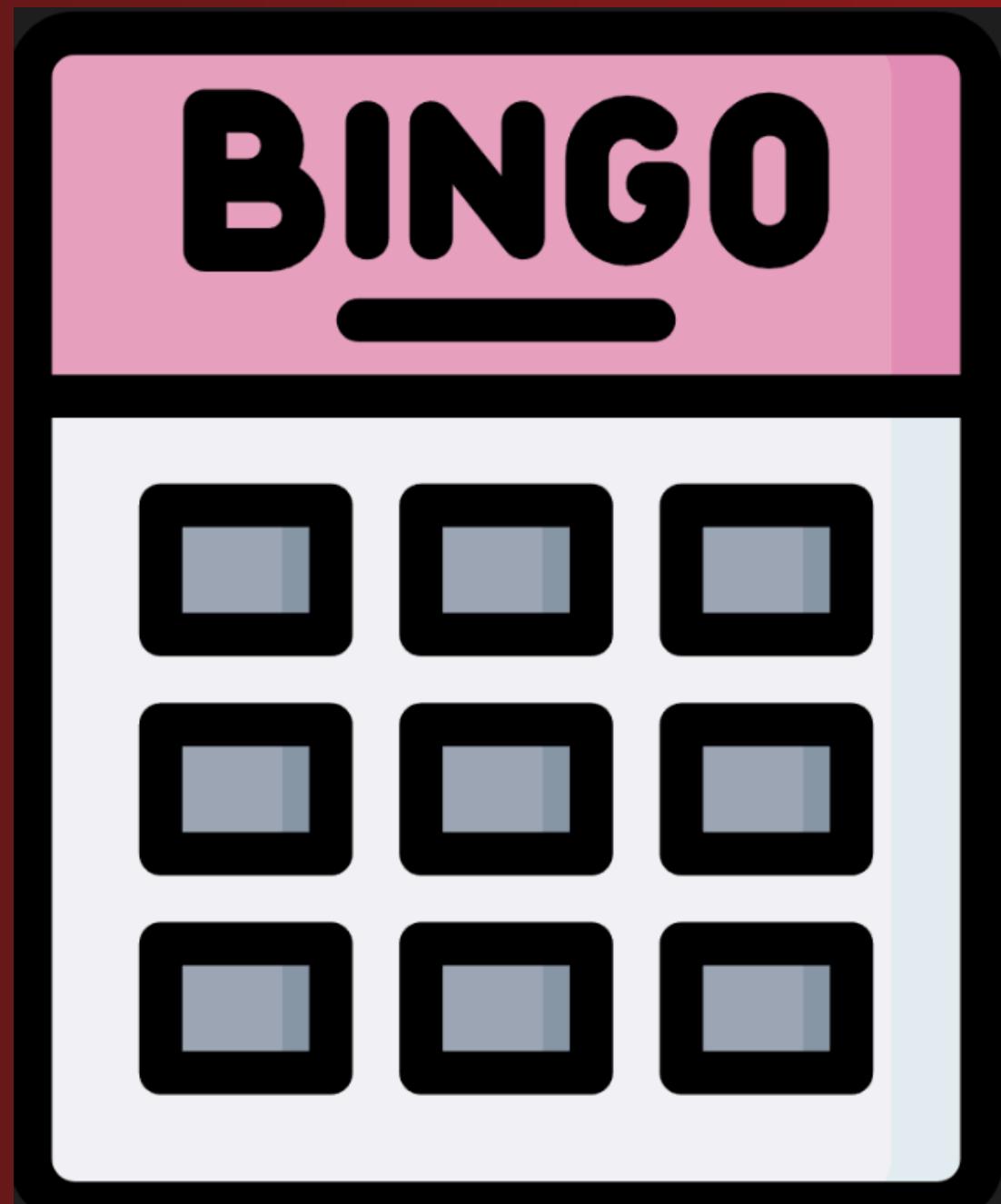
# Android Studio

- Android Studio zaprezentowano 15 maja 2013 roku na konferencji Google I/O
- Pierwsze stabilne wydanie (1.0) nastąpiło w grudniu następnego roku
- Środowisko obsługuje języki programowania: Java, C++, Kotlin
- Od 2019 roku Kotlin jest preferowanym językiem (nie usunięto wsparcia dla pozostałych)
- Środowisko wspiera emulator Androida, dzięki któremu aplikacje można testować bez dostępu do urządzenia mobilnego



# Omówienie projektu

- Gra Bingo w Android Studio, z losowanymi liczbami z zakresu 1-50
- Tryb Singleplayer
- Tryb Multiplayer, umożliwiający rozgrywkę dwóm graczom znajdującym się w tym samym “pokoju” w bazie Firebase Realtime Database.



# Co to Bingo?



# Co to Bingo?

- Klasycznie Bingo to gra, w której gracze na wydrukowanej papierowej planszy z liczbami zakreślają liczby wylosowane przez prowadzącego grę - callera
- Gra jest znana w dwóch wersjach: z planszą o rozmiarach 5x5 - BINGO 75 lub 3x9 - BINGO 90.
- Pierwsza osoba, której wylosowane będą wszystkie liczby na planszy (BINGO 90) lub wszystkie liczby w linii (BINGO 75) wykrzykuje „Bingo！”, informując innych o wygranej.

# Co to Bingo?

- Każdy gracz otrzymuje planszę z siatką zawierającą kombinację liczb. Zwycięzca ogłasza uformowanie linii z liczb (BINGO 75) lub wylosowane wszystkich liczb (BINGO 90-kulowe) na planszy.
- W każdej turze prowadzący wylosowuje numerowaną kulę z pojemnika i ogłasza liczbę wszystkim gracjom. Kula jest odkładana na bok, aby nie mogła być wylosowana ponownie. Każdy gracz szuka na swojej karcie wylosowanej liczby i zaznacza ją.
- Większość graczy zaznacza cyfry na wielu planszach naraz; 30 plansz nie jest rzeczą niezwykłą. Z powodu dużej liczby plansz używanych przez każdego gracza, hale mają oddzielny stolik dla każdego gracza, na którym mocuje plansze za pomocą taśmy klejącej. Aby zaznaczać cyfry szybciej używane są specjalne markery zwane dabber.

# Od czego by tu zacząć...

## StartActivity.java

```
import android.view.View;
import android.widget.Button;

public class StartActivity extends Activity {

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

        Button singlePlayerButton = findViewById(R.id.singlePlayerButton);
        Button multiplayerButton = findViewById(R.id.multiplayerButton);

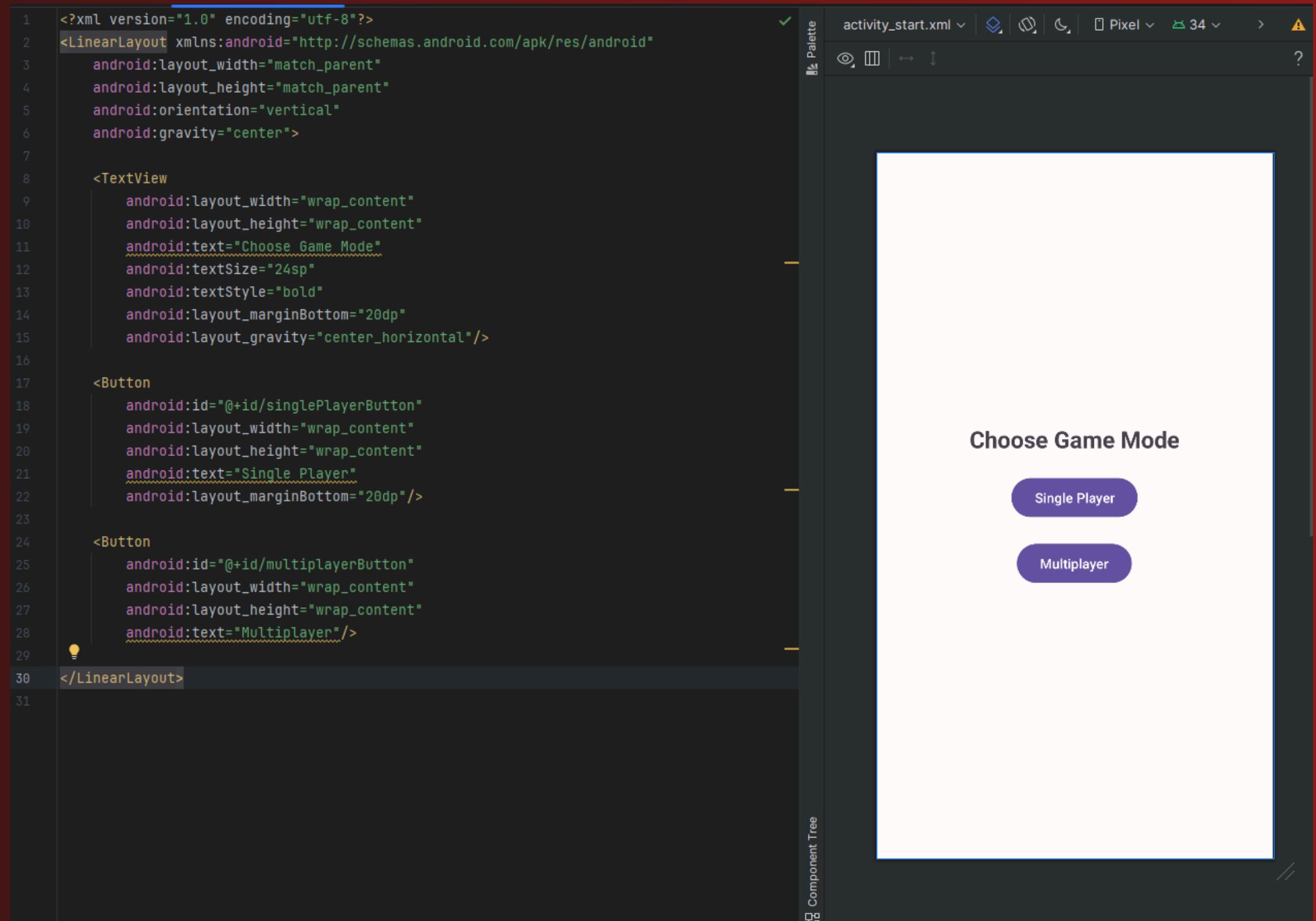
        singlePlayerButton.setOnClickListener(v -> startMainActivity(false));
        multiplayerButton.setOnClickListener(v -> startMultiplayerActivity(true));
    }

    private void startMainActivity(boolean isMultiplayer) {
        Intent intent = new Intent(this, MainActivity.class);
        intent.putExtra("isMultiplayer", isMultiplayer);
        startActivity(intent);
    }

    private void startMultiplayerActivity(boolean isMultiplayer) {
        Intent intent = new Intent(this, MultiplayerActivity.class);
        intent.putExtra("isMultiplayer", isMultiplayer);
        startActivity(intent);
    }
}
```

# activity\_start.xml

- Widok po uruchomieniu aplikacji
- Wybór trybu rozgrywki



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical"
6     android:gravity="center">
7
8     <TextView
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:text="Choose Game Mode"
12        android:textSize="24sp"
13        android:textStyle="bold"
14        android:layout_marginBottom="20dp"
15        android:layout_gravity="center_horizontal"/>
16
17     <Button
18         android:id="@+id/singlePlayerButton"
19         android:layout_width="wrap_content"
20         android:layout_height="wrap_content"
21         android:text="Single Player"
22         android:layout_marginBottom="20dp"/>
23
24     <Button
25         android:id="@+id/multiplayerButton"
26         android:layout_width="wrap_content"
27         android:layout_height="wrap_content"
28         android:text="Multiplayer"/>
29
30 </LinearLayout>
31
```

# activity\_start.xml

- Przykład użycia widoku pola tekstowego

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Choose Game Mode"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_marginBottom="20dp"
        android:layout_gravity="center_horizontal"/>

    <Button
        android:id="@+id/singlePlayerButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Single Player"
        android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/multiplayerButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Multiplayer"/>

</LinearLayout>
```

# activity\_start.xml

- Przykład użycia przycisków

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Choose Game Mode"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_marginBottom="20dp"
        android:layout_gravity="center_horizontal"/>

    <Button
        android:id="@+id/singlePlayerButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Single Player"
        android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/multiplayerButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Multiplayer"/>

</LinearLayout>
```

The screenshot shows the Android Studio interface with the XML code for the `activity_start.xml` layout. The code defines a vertical `LinearLayout` containing a `TextView` and two `Button` elements. The `TextView` has the text "Choose Game Mode". The first `Button` is labeled "Single Player" and the second is labeled "Multiplayer". Both buttons have a `layout_marginBottom="20dp"` attribute. A red box highlights the XML code for the two buttons, and a red arrow points from this box to a red box highlighting the "Single Player" and "Multiplayer" buttons in the preview window. The preview window shows a white screen with the text "Choose Game Mode" at the top, followed by two purple rounded rectangular buttons labeled "Single Player" and "Multiplayer".

```
1 package com.example.bingo;
2
3 > import ...
4
5
6 D</> public class StartActivity extends Activity {
7     // private WifiDirectManager wifiDirectManager = new WifiDirectManager(this);
8
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_start);
14
15        Button singlePlayerButton = findViewById(R.id.singlePlayerButton);
16        Button multiplayerButton = findViewById(R.id.multiplayerButton);
17
18        singlePlayerButton.setOnClickListener(v -> startMainActivity(isMultiplayer: false));
19        multiplayerButton.setOnClickListener(v -> startMultiplayerActivity(isMultiplayer: true));
20
21    }
22
23
24    1 usage
25    private void startMainActivity(boolean isMultiplayer) {
26        Intent intent = new Intent(packageContext: this, MainActivity.class);
27        intent.putExtra(name: "isMultiplayer", isMultiplayer);
28        startActivity(intent);
29    }
29
30    1 usage
31    private void startMultiplayerActivity(boolean isMultiplayer) {
32        /wifiDirectManager.discoverPeers();
33        Intent intent = new Intent(packageContext: this, MultiplayerActivity.class);
34        intent.putExtra(name: "isMultiplayer", isMultiplayer);
35        startActivity(intent);
36
37    }
38}
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical"
6     android:gravity="center">
7
8     <TextView
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"
11        android:text="Choose Game Mode"
12        android:textSize="24sp"
13        android:textStyle="bold"
14        android:layout_marginBottom="20dp"
15        android:layout_gravity="center_horizontal"/>
16
17     <Button
18         android:id="@+id/singlePlayerButton"
19         android:layout_width="wrap_content"
20         android:layout_height="wrap_content"
21         android:text="Single Player"
22         android:layout_marginBottom="20dp"/>
23
24     <Button
25         android:id="@+id/multiplayerButton"
26         android:layout_width="wrap_content"
27         android:layout_height="wrap_content"
28         android:text="Multiplayer"/>
29
30 </LinearLayout>
31
```

# AndroidManifest.xml

- Każdy projekt aplikacji musi mieć plik AndroidManifest.xml z dokładnie taką nazwą w katalogu głównym zbioru źródłowego projektu
- Musi między innymi zadeklarować:
  - Komponenty aplikacji, w tym cała aktywność, usługi, odbiorniki i dostawcy treści.
  - Uprawnienia wymagane przez aplikację do korzystania z chronionych części systemu lub innych aplikacji.
  - Sprzęt i funkcje oprogramowania, których wymaga aplikacja, wpływają na to, na jakich urządzeniach można ją zainstalować z Google Play

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      package="com.example.bingo">
5
6      <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
7      <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
8      <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
9      <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
10     <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
11     <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
12     <uses-permission android:name="android.permission.INTERNET" />
13     <uses-permission android:name="android.permission.NEARBY_WIFI_DEVICES" />
14     <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
15     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
16
17
18     <application
19         android:allowBackup="true"
20         android:dataExtractionRules="@xml/data_extraction_rules"
21         android:fullBackupContent="@xml/backup_rules"
22         android:icon="@drawable/b_obrazek"
23         android:label="Bingo"
24         android:roundIcon="@drawable/bingo_obrazek"
25         android:supportsRtl="true"
26         android:theme="@style/Theme.Bingo"
27         tools:targetApi="31">
28         <activity
29             android:name=".MainActivity"
30             android:exported="true"
31             android:label="Bingo"
32             android:screenOrientation="landscape"
33             android:theme="@style/Theme.Bingo">
34             <intent-filter>
35                 <action android:name="android.intent.action.MAIN" />
36
37                 <category android:name="android.intent.category.LAUNCHER" />
38             </intent-filter>
39
40         </activity>
```

## Zadeklarowanie aktywności - odpowiednie klasy

```
<activity
    android:name=".StartActivity"
    android:exported="true"
    android:label="Bingo"
    android:screenOrientation="landscape"
    android:theme="@style/Theme.Bingo">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>

</activity>
<activity
    android:name=".MainActivity"
    android:exported="true"
    android:label="Bingo"
    android:screenOrientation="landscape"
    android:theme="@style/Theme.Bingo" />
<activity
    android:name=".MultiplayerGameLogic"
    android:exported="true"
    android:label="Bingo"
    android:screenOrientation="landscape"
    android:theme="@style/Theme.Bingo" />
<activity
    android:name=".MultiplayerActivity"
    android:label="Multiplayer Activity"
    android:theme="@style/Theme.Bingo"
    android:screenOrientation="landscape" />
```

# Najważniejsze pliki

## activity\_main.xml

- Plik ten odpowiada za główny layout aplikacji
- Zawiera elementy takie jak: przyciski, widoki tekstu, widoki obrazów itp.

The screenshot shows the Android Studio interface with the code editor open to the `activity_main.xml` file. The code defines a ConstraintLayout with various views like a toolbar, button, text view, and progress bar. To the right, the design tab displays a 5x5 grid of purple rounded rectangles labeled 1 through 25, representing the bingo board. Below the design tab, the component tree shows a hierarchy of buttons corresponding to the grid cells.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    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"
    android:fitsSystemWindows="true"
    tools:context=".MainActivity">

    <Toolbar
        android:id="@+id/toolbar"
        android:layout_width="0dp"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        android:elevation="4dp"
        android:theme="?attr/actionBarTheme"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <Button
            android:id="@+id/replayButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toTopOf="@+id/bingoImage"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            android:layout_marginTop="10dp"
            android:text="Replay"
            />

        <TextView
            android:id="@+id/toolbar_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/white"
            android:textSize="20sp" />

    <ProgressBar
        android:id="@+id/progress_bar"
        ...>
```

# MainActivity.java

- Plik ten odpowiada za funkcjonalność aplikacji
- Nasłuchiwacze kliknięć, przypisywanie tekstów do przycisków itp.

```
1 package com.example.bingo;
2
3 > import ...
22
23 </> public class MainActivity extends Activity {
24
25     2 usages
26     private static final Logger logger = LogManager.getLogger(MultiplayerGameLogic.class);
27     no usages
28     private boolean isMultiplayerMode = false;
29     3 usages
30     private boolean isGameStarted = false;
31     no usages
32     private String winnerID = null;
33     1 usage
34     private HashMap<String, Boolean> playerStatusMap;
35     6 usages
36     private final int[] buttonIds = {
37         R.id.button1, R.id.button2, R.id.button3, R.id.button4,
38         R.id.button5, R.id.button6, R.id.button7, R.id.button8,
39         R.id.button9, R.id.button10, R.id.button11, R.id.button12,
40         R.id.button13, R.id.button14, R.id.button15, R.id.button16,
41         R.id.button17, R.id.button18, R.id.button19, R.id.button20,
42         R.id.button21, R.id.button22, R.id.button23, R.id.button24,
43         R.id.button25
44     };
45
46     6 usages
47     private ProgressBar progressBar;
48     5 usages
49     private TextView toolbarTitle;
50     3 usages
51     private CountDownTimer countDownTimer;
52     3 usages
53     private Random random;
54     4 usages
55     private int selectedNumber;
56     6 usages
57     private ArrayList<Integer> availableNumbers;
58     5 usages
59     private boolean gameActive = true;
60     4 usages
```

## Pliki te współpracują ze sobą:

Przypisanie konkretnego layoutu

```
<Button  
    android:id="@+id/replayButton"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintBottom_toTopOf="@+id/bingoImage"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    android:layout_marginTop="10dp"  
    android:text="Replay"  
/>
```

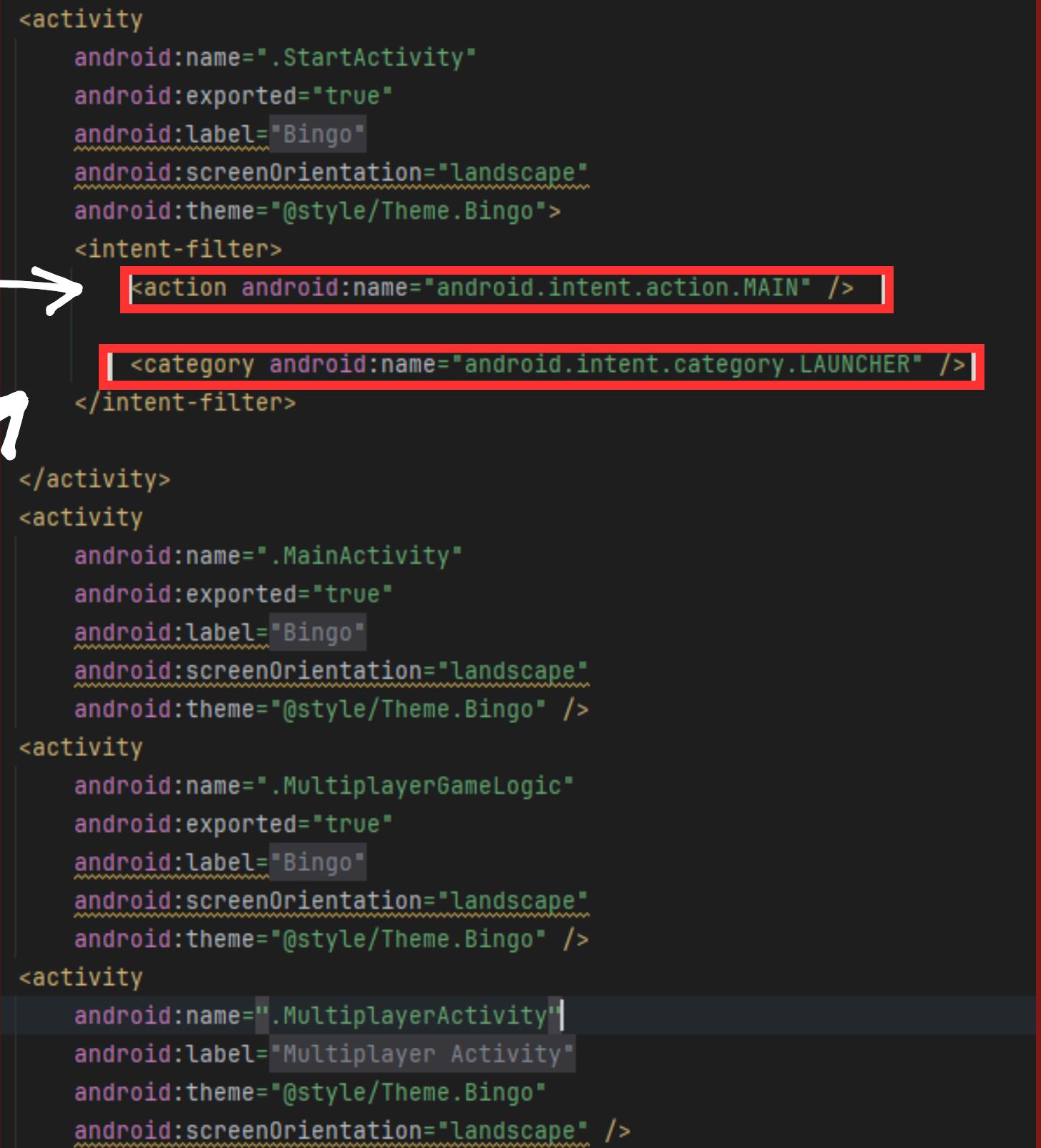
Przypisywanie widoków

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    boolean isMultiplayer = getIntent().getBooleanExtra("isMultiplayer", false);  
  
    bingoImage = findViewById(R.id.bingoImage);  
    replayButton = findViewById(R.id.replayButton);  
  
    replayButton.setOnClickListener(replayButtonClickListener);  
  
    bingoImage.setVisibility(View.GONE);  
    replayButton.setVisibility(View.GONE);  
  
    progressBar = findViewById(R.id.progress_bar);  
    toolbarTitle = findViewById(R.id.toolbar_title);  
    randomNumberTextView = findViewById(R.id.randomNumberTextView);  
  
    random = new Random();  
    playerStatusMap = new HashMap<>();  
  
    startGame();  
}
```

## Filtr intencji

Definiuje, że aktywność z tym filtrem intencji jest główną aktywnością aplikacji. Oznacza to, że ta aktywność będzie uruchamiana jako pierwsza, gdy użytkownik uruchomi aplikację.

Ta linia określa, że aktywność z tym filtrem intencji jest także dostępna z ekranu głównego urządzenia, czyli jest to ikona aplikacji widoczna w menu aplikacji lub na pulpicie.



```
<activity
    android:name=".MainActivity"
    android:exported="true"
    android:label="Bingo"
    android:screenOrientation="landscape"
    android:theme="@style/Theme.Bingo">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".MultiplayerGameLogic"
    android:exported="true"
    android:label="Bingo"
    android:screenOrientation="landscape"
    android:theme="@style/Theme.Bingo" />
<activity
    android:name=".MultiplayerActivity"
    android:label="Multiplayer Activity"
    android:theme="@style/Theme.Bingo"
    android:screenOrientation="landscape" />
```

# Inicjalizacja planszy z przyciskami

Przypisanie losowych numerów do przycisków w Bingo

```
88     private void initializeGame() {
89         availableNumbers = new ArrayList<>();
90         for (int i = 1; i <= 50; i++) {
91             availableNumbers.add(i);
92         }
93
94         ArrayList<Integer> randomNumbers = new ArrayList<>(availableNumbers);
95         Collections.shuffle(randomNumbers, random);
96
97         // Przypisanie unikalnych tekstów do przycisków
98         for (int i = 0; i < buttonIds.length; i++) {
99             final Button button = findViewById(buttonIds[i]);
100            button.setText(String.valueOf(randomNumbers.get(i)));
101
102            int color = Color.GRAY;
103            button.setBackgroundColor(color);
104            button.setOnClickListener(buttonClickListener);
105        }
106    }
107 }
```

# Licznik czasu pojedyńczej rundy

- Timer odliczający 5 sekund - czas na kliknięcie wylosowanej liczby
- Po 5 sekundach timer jest zakrywany i zastępuje go tekst, po 3 sekundach jest losowana kolejna liczba.

```
108     private void startTimer() {
109         countDownTimer = new CountDownTimer( millisInFuture: 6000, countDownInterval: 1000 ) {
110             no usages
111             @SuppressLint("SetTextI18n")
112             @Override
113             public void onTick(long millisUntilFinished) {
114                 long seconds = millisUntilFinished / 1000;
115                 progressBar.setProgress((int) millisUntilFinished);
116                 toolbarTitle.setText("Time left: " + seconds + "s");
117                 randomNumberTextView.setText(("Select Number: " + selectedNumber));
118
119                 // Jeżeli to pierwsza sekunda, generuj nową liczbę i wyświetl ją w toolbarze
120                 if (seconds == 5) {
121                     generateRandomNumber();
122                     toolbarTitle.setText("Time left: " + seconds + "s");
123                     randomNumberTextView.setText(("Select Number: " + selectedNumber));
124                 }
125             }
126         }
127         no usages
128         @Override
129         public void onFinish() {
130             progressBar.setVisibility(ProgressBar.GONE);
131             toolbarTitle.setText("Time's up!");
132             gameActive = false;
133             progressBar.postDelayed(() -> {
134                 progressBar.setVisibility(ProgressBar.VISIBLE);
135                 startTimer();
136                 gameActive = true;
137             }, delayMillis: 3000);
138         }.start();
139     }
140 }
```

# Działanie przycisku Replay

- Rozpoczęcie nowej rozgrywki po kliknięciu przycisku Replay.
- Zakrycie obrazu Bingo i przycisku Replay

```
164     private final View.OnClickListener replayButtonClickListener = v -> {
165         // Przywróć przyciski do stanu początkowego
166
167         gameActive = true;
168         isGameStarted = false;
169
170         for (int buttonId : buttonIds) {
171             Button button = findViewById(buttonId);
172             button.setVisibility(View.VISIBLE);
173         }
174
175         // Ukryj obrazek i przycisk Replay
176         bingoImage.setVisibility(View.GONE);
177         replayButton.setVisibility(View.GONE);
178
179
180         initializeGame();
181         startTimer();
182     };
183 }
```

# Działanie przycisków na planszy

- Funkcjonalność przycisków z numerami
- Zmiana koloru poprawnych, klikniętych przycisków na żółty
- Jeśli metoda checkBingo zwraca jedynkę to wywołuje metodę wyświetlającą Bingo

```
185     private final View.OnClickListener buttonClickListener = v -> {
186         Button button = (Button) v;
187
188         if (!gameActive) {
189             return; //jesli czas uplynal zakończ obsługę kliknięcia
190         }
191         if (button.getText().toString().equals(String.valueOf(selectedNumber))) {
192             if (button.getTag() == null || (int) button.getTag() == 0) {
193                 // Zmiana koloru na żółty po kliknięciu
194                 int color = Color.YELLOW;
195                 button.setBackgroundColor(color);
196
197                 button.setTag(1);
198
199             } else {
200                 // Powrót do pierwotnego koloru po drugim kliknięciu
201                 int color = Color.GRAY; // Tu możesz użyć koloru pierwotnego
202                 button.setBackgroundColor(color);
203                 button.setTag(0);
204             }
205
206
207             button.requestLayout();
208
209             // Sprawdź, czy wszystkie przyciski w pionie lub poziomie są kliknięte i żółte
210             if (checkBingo()) {
211                 // Jeśli tak, wyświetl napis "BINGO"
212                 displayBingo();
213
214             }
215     };
```

# Losowanie “kul” z liczbami

- Metoda generująca losową liczbę, taką, która jeszcze się nie pojawiła. W przypadku braku takiej liczby (jeśli użytkownik nie zaznaczył odpowiednich liczb), zostanie wywołana metoda showLoseScreen

```
234     private void generateRandomNumber() {  
235         if (checkBingo()){  
236             logger.info("Bingo game won");  
237             return;  
238         }  
239         if(attempts>=MAX_ATTEMPTS){  
240             showLoseScreen();  
241         }  
242         ArrayList<Integer> numbersList = new ArrayList<>(availableNumbers);  
243         Collections.shuffle(numbersList, random);  
244  
245         // Iteruj przez liste wylosowanych liczb, aż znajdziesz taką, która jest dostępna  
246         for (Integer number : numbersList) {  
247             if (availableNumbers.contains(number)) {  
248                 selectedNumber = number;  
249                 availableNumbers.remove(number);  
250                 attempts++;  
251             }  
252         }  
253     }  
254  
255     // Jeśli nie znaleziono dostępnej liczby, obsłuż tę sytuację  
256     logger.error( message: "generateRandomNumber", p0: "No available number found after shuffling.");  
257 }  
258 }
```

# Sprawdzanie BINGO

- Sprawdzenie czy wystąpiło Bingo w poziomie lub pionie

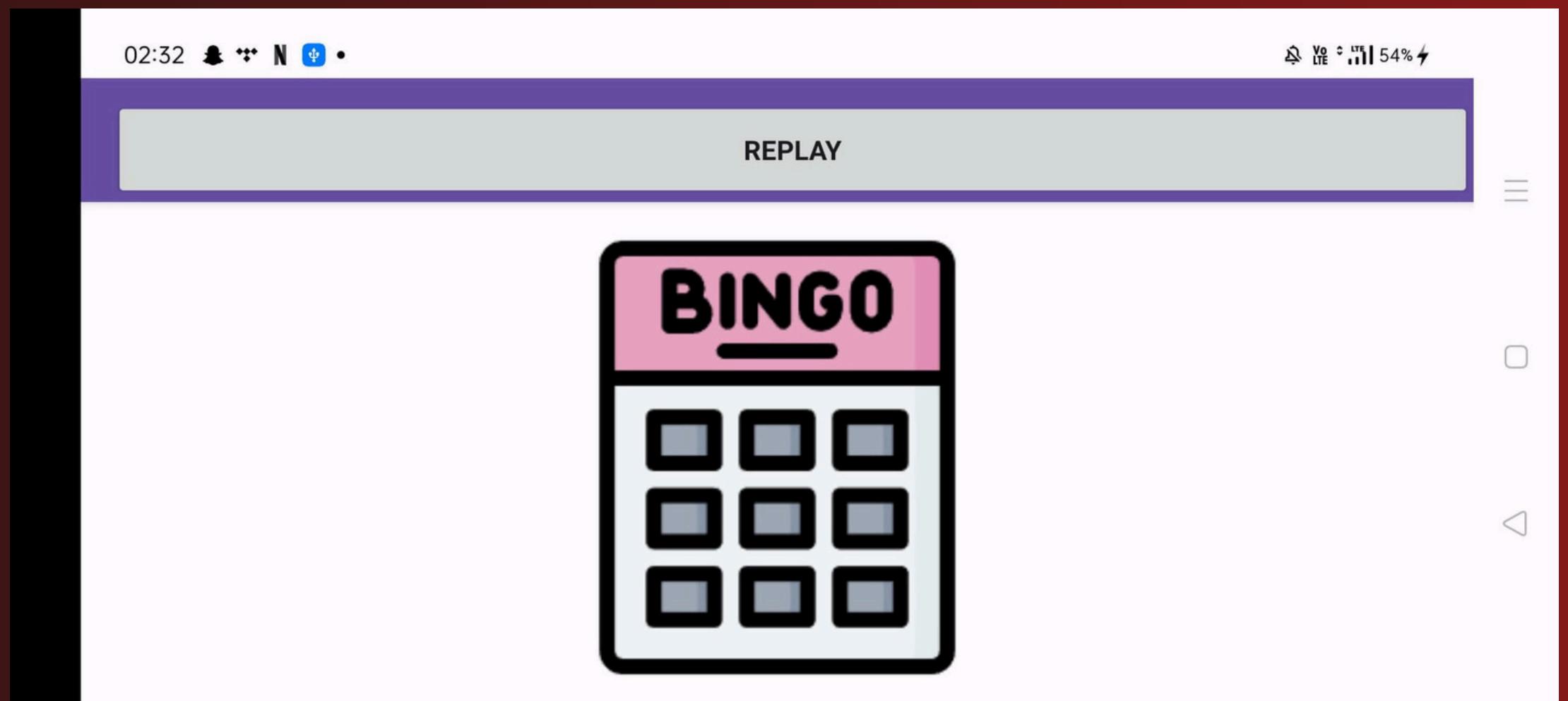
```
240     private boolean checkBingo() {  
241         // Check for horizontal BINGO  
242         for (int i = 0; i < 5; i++) {  
243             boolean isBingo = true;  
244             for (int j = 0; j < 5; j++) {  
245                 Button button = findViewById(buttonIds[i * 5 + j]);  
246                 if (button.getTag() == null || (int) button.getTag() == 0) {  
247                     isBingo = false;  
248                     break;  
249                 }  
250             }  
251             if (isBingo) {  
252                 return true;  
253             }  
254         }  
255         // Check for vertical BINGO  
256         for (int i = 0; i < 5; i++) {  
257             boolean isBingo = true;  
258             for (int j = 0; j < 5; j++) {  
259                 Button button = findViewById(buttonIds[j * 5 + i]);  
260                 if (button.getTag() == null || (int) button.getTag() == 0) {  
261                     isBingo = false;  
262                     break;  
263                 }  
264             }  
265             if (isBingo) {  
266                 return true;  
267             }  
268         }  
269     }  
270     return false;  
271 }  
272 }  
273 }
```

# Główna funkcjonalność

- W przypadku końca gry, przyciski są ukrywane i na ekran wskakuje obraz Bingo oraz przycisk Replay, który pozwala na ponowną rozgrywkę

```
151     private void displayBingo() {  
152         gameActive = false;  
153         for (int buttonId : buttonIds) {  
154             Button button = findViewById(buttonId);  
155             button.setVisibility(View.GONE);  
156         }  
157         progressBar.setVisibility(View.GONE);  
158         toolbarTitle.setVisibility(View.GONE);  
159         bingoImage.setVisibility(View.VISIBLE);  
160         replayButton.setVisibility(View.VISIBLE);  
161     }  
162 }
```

- Ekran w przypadku wygranej

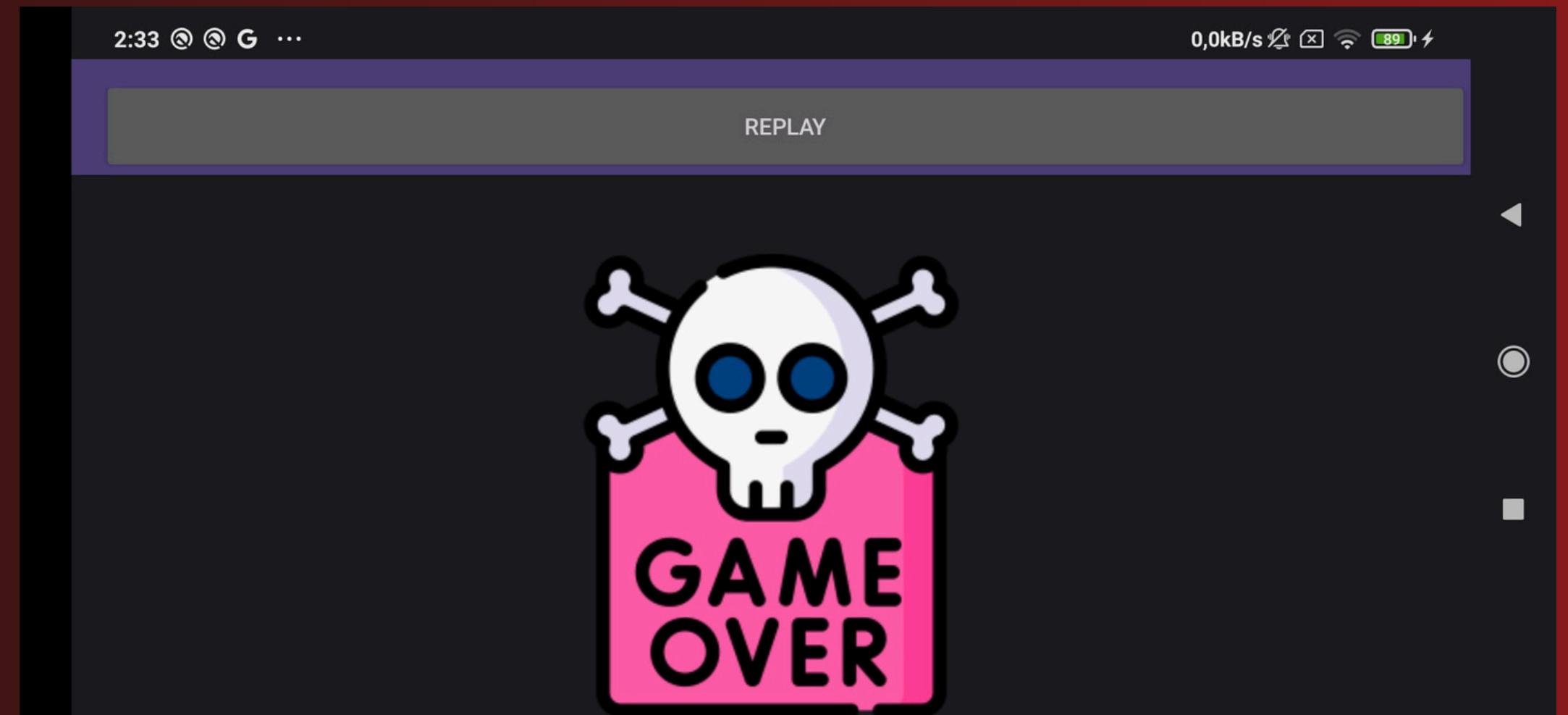


# Główna funkcjonalność

- Ekran, który pojawi się w przypadku przegranej

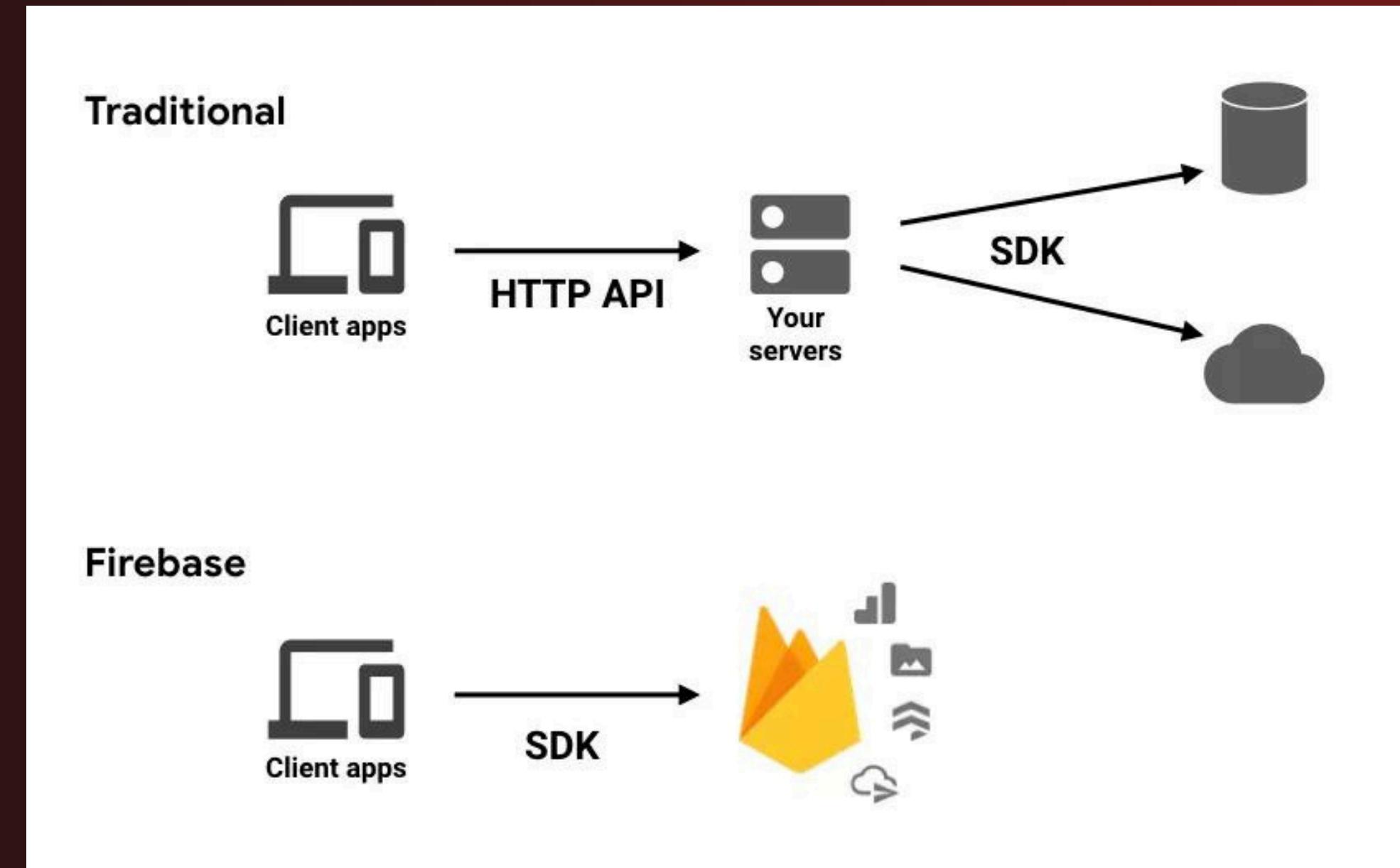
```
220     private void showLoseScreen() {
221         try {
222             for (int buttonId : buttonIds) {
223                 Button button = findViewById(buttonId);
224                 button.setVisibility(View.GONE);
225             }
226             toolbarTitle.setVisibility(View.GONE);
227             loseMessageTextView.setVisibility(View.VISIBLE);
228             replayButton.setVisibility(View.VISIBLE);
229             logger.info("showLoseScreen: Lose screen displayed");
230         } catch (Exception e) {
231             logger.error( message: "showLoseScreen: Exception caught: {}", e.getMessage());
232         }
233     }
```

- Ekran w przypadku przegranej



# Firebase

# Firebase



# Firebase



Build better apps



Auth



Hosting



Cloud Functions



ML Kit



Cloud Firestore



Realtime Database



Cloud Storage



Improve app quality



Crashlytics



Performance Monitoring



Test Lab



Grow your app



Analytics



Remote Config



Predictions



A/B Testing



Cloud Messaging



Dynamic Links



In-app Messaging

# Autentykacja

The screenshot shows the Firebase Authentication console interface. At the top, there are tabs for 'Bingo' (selected), 'Authentication', 'Sign-in method', 'Templates', 'Usage', 'Settings', and 'Extensions'. Below the tabs is a search bar and an 'Add user' button. A table lists two users:

Identifier	Providers	Created	Signed In	User UID
wiet@wp.pl	Email	May 4, 2024	May 4, 2024	86UwKVrJLXNnlhxAgEDtIJ0w...
abcd@wp.pl	Email	May 4, 2024	May 4, 2024	m0hDlZdxWHUP9Alu8sJfZHW...

At the bottom, there are buttons for 'Rows per page' (set to 50), '1 - 2 of 2', and navigation arrows.

## MultiplayerActivity.java

```
firebaseAuth = FirebaseAuth.getInstance();

String email = editTextNickname.getText().toString().trim(); // Użyj pola nickname jako adresu e-mail
String password = editTextGamePassword.getText().toString().trim(); // Użyj pola gamePassword jako hasła

firebaseAuth.createUserWithEmailAndPassword(email, password)

firebaseAuth.signInWithEmailAndPassword(email, password)

@Override
public void onComplete(@NonNull Task<AuthResult> task) {
    if (task.isSuccessful()) {
        // Logowanie zakończone sukcesem
        FirebaseUser user = firebaseAuth.getCurrentUser();
        if (user != null) {
            // Zalogowano użytkownika, pobierz jego UID
            String uid = user.getUid();
            // Użyj UID do identyfikacji użytkownika
            // Możesz również zapisać nick użytkownika w bazie danych Firebase
            joinGame(uid, email, nickname);
        } else {
            // Użytkownik jest null, obsłuż ten przypadek
            Toast.makeText(context, MultiplayerActivity.this, "User is null", Toast.LENGTH_SHORT).show();
        }
    } else {
        // Logowanie nie powiodło się, wyświetl komunikat o błędzie
        Toast.makeText(context, MultiplayerActivity.this, "Authentication failed", Toast.LENGTH_SHORT).show();
    }
}
});
```

# Autentykacja

The screenshot shows a user interface for selecting a sign-in provider. At the top right is a button labeled "Add new provider". Below it, there are two tabs: "Provider" and "Status", with "Provider" being the active tab. A close button "X" is located in the top right corner of the main content area.

The main content area is titled "Select a sign-in provider (Step 1 of 2)". It is divided into three columns:

- Native providers**:
  - Email/Password ✓ (selected)
  - Phone
  - Anonymous ✓
- Additional providers**:
  - Google
  - Facebook
  - Play Games
  - Game Center
  - Apple
  - Github
  - Microsoft
  - Twitter
  - Yahoo
- Custom providers**:
  - OpenID Connect
  - SAML

At the bottom left is a "Email/Password" link, and at the bottom right is a "Enabled" status indicator with a checkmark.

# Baza danych

The screenshot shows the Firebase Realtime Database interface. At the top, there's a navigation bar with 'Bingo' and various icons. Below it, the title 'Realtime Database' is displayed, followed by a navigation menu with 'Data' (selected), 'Rules', 'Backups', 'Usage', and 'Extensions'. A prominent banner at the top urges users to 'Protect your Realtime Database resources from abuse, such as billing fraud or phishing' with a 'Configure App Check' button.

The main area shows a tree view of database data. The URL in the browser bar is <https://bingo-a1100-default-rtbd.firebaseio.com/>. The data structure under 'games' is as follows:

- games
  - wiet123
    - players
      - 86UwKVrJLXNnihxAgEDtIJ0wbxs2
        - won: false
      - m0hDlZdxWHUP9Alu8sJfZHwlbN63
        - won: false

# MultiplayerGameLogic.java

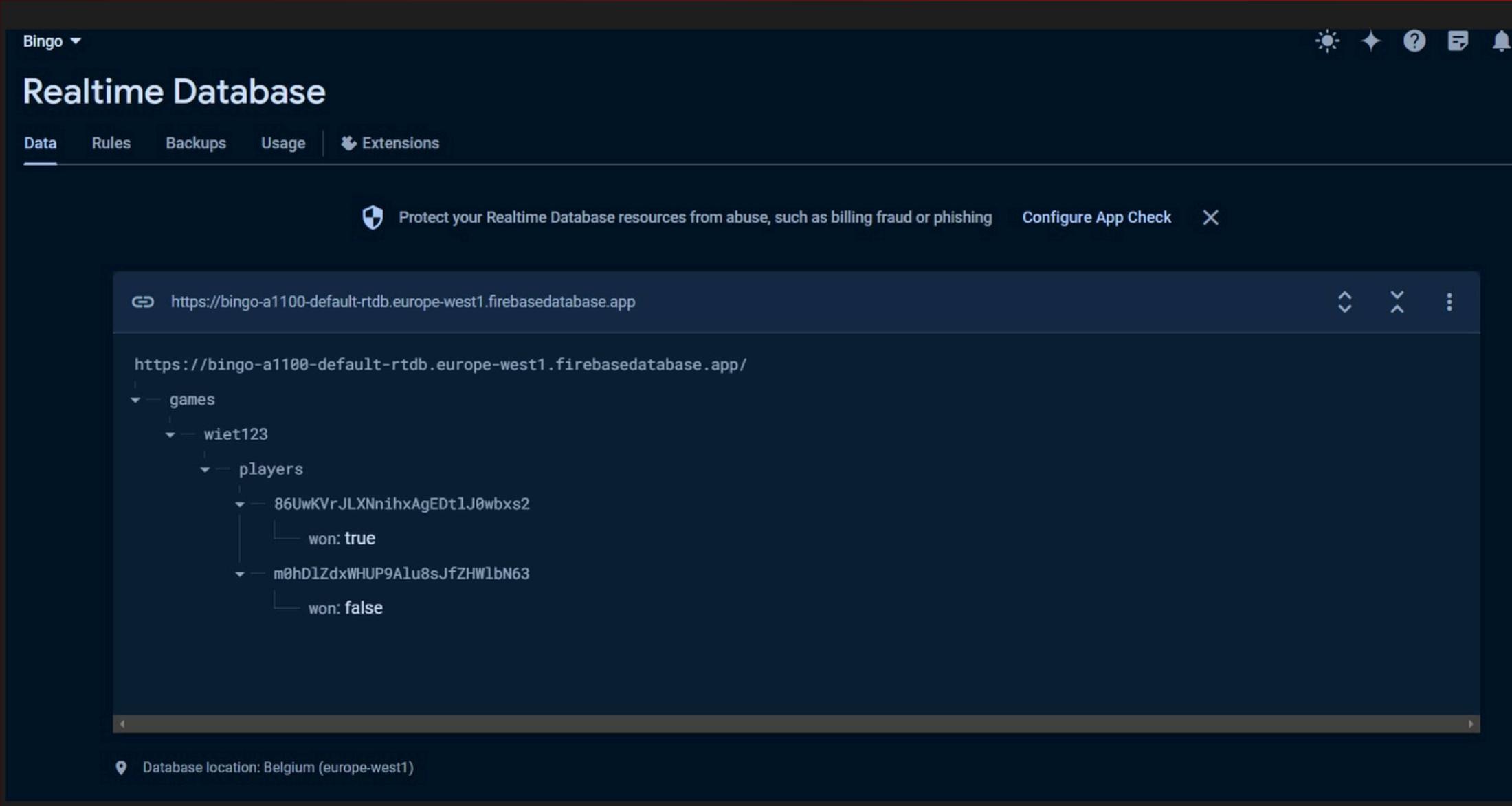
## FirebaseManager.java

```
public class FirebaseManager {  
    FirebaseDatabase database = FirebaseDatabase.getInstance();  
  
    public FirebaseManager(Context context) {  
        this.context = context;  
        String databaseURL = database.getReference().toString();  
        Log.d("FirebaseURL", databaseURL);  
        firebaseAuth = FirebaseAuth.getInstance();  
        gamesRef = FirebaseDatabase.getInstance().getReference("games");  
    }  
  
    public void joinGame(String gamePassword, String uid, String nickname, OnCompleteListener<Void> onCompleteListener) {  
        DatabaseReference playersRef = gamesRef.child(gamePassword).child("players");  
  
        // Ustaw początkowy status wygranej na "false" dla nowo dołączonego gracza  
        HashMap<String, Object> playerData = new HashMap<>();  
        playerData.put("won", false);  
  
        // Dodaj gracza do listy graczy w pokoju gry  
        playersRef.child(uid).setValue(playerData)  
            .addOnCompleteListener(task -> {  
                if (task.isSuccessful()) {  
                    // Jeśli dodanie gracza zakończyło się sukcesem, zwróć sukces do metody wywołującej  
                    onCompleteListener.onComplete(task);  
                } else {  
                    // W przypadku niepowodzenia dodania gracza, zwróć niepowodzenie do metody wywołującej  
                    onCompleteListener.onComplete(task);  
                }  
            });  
    }  
}
```

```
private void checkPlayersJoined(String gamePassword) {  
    gamesRef.child(gamePassword).child("players").addValueEventListener(new ValueEventListener() {  
        @Override  
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {  
            if (dataSnapshot.getChildrenCount() >= 2) {  
                // Obaj gracze dołączyli do gry, zacznią grę  
                startGame();  
            }  
        }  
        @Override  
        public void onCancelled(@NonNull DatabaseError databaseError) {  
            // Obsługa błędu  
        }  
    });  
}
```

Sprawdza czy jest chociaż 2 graczy

# Wykorzystanie bazy danych do ustanowienia zwycięzcy



# MultiplayerGameLogic.java

```
private void displayBingo() {
    gameActive = false;
    for (int buttonId : buttonIds) {
        Button button = findViewById(buttonId);
        button.setVisibility(View.GONE);
    }
    progressBar.setVisibility(View.GONE);
    toolbarTitle.setVisibility(View.GONE);
    randomNumberTextView.setVisibility(View.GONE);
    bingoImage.setVisibility(View.VISIBLE);
    replayButton.setVisibility(View.VISIBLE);

    // Pobierz UID aktualnie zalogowanego użytkownika
    String uid = firebaseAuth.getCurrentUser().getUid();

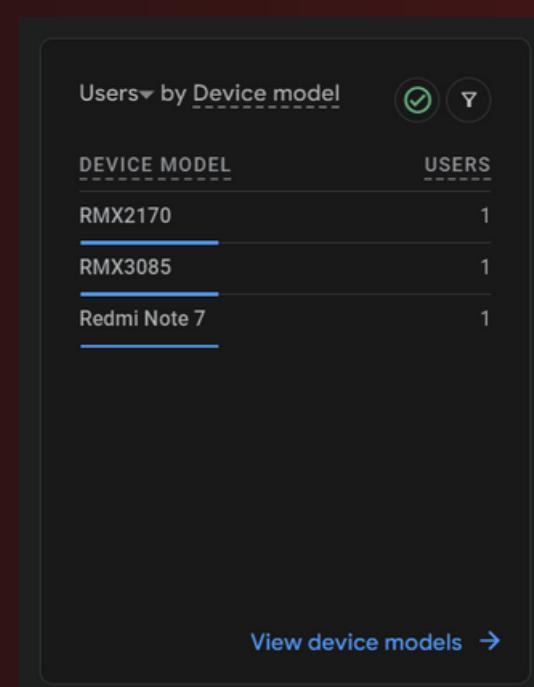
    // Ustaw status wygranej na "true" dla aktualnie zalogowanego gracza
    firebaseManager.setPlayerWinStatus(gamePassword, uid, true, new OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
            if (task.isSuccessful()) {
                // Status wygranej gracza został pomyślnie ustawiony na "true"
                // Możesz wykonać dodatkowe czynności, jeśli są wymagane

            } else {
                // Nie udało się ustawić statusu wygranej gracza
                // Obsłuż tę sytuację, jeśli jest to konieczne
            }
        }
    });
}
```

```
private void checkAndShowLoseScreen(String gamePassword) {
    DatabaseReference playersRef = FirebaseDatabase.getInstance().getReference("games")
        .child(gamePassword).child("players");

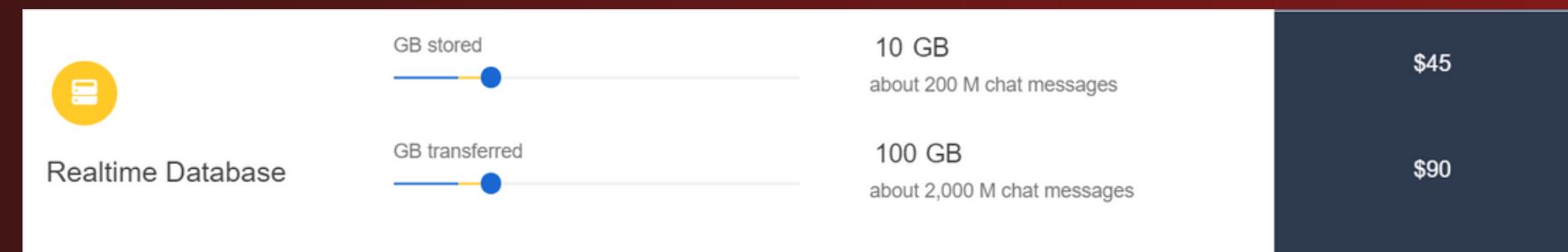
    playersRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            boolean anyPlayerWon = false;
            for (DataSnapshot playerSnapshot : dataSnapshot.getChildren()) {
                boolean hasWon = playerSnapshot.child("won").getValue(Boolean.class);
                if (hasWon) {
                    anyPlayerWon = true;
                    break;
                }
            }
            if(anyPlayerWon) {
                for (DataSnapshot playerSnapshot : dataSnapshot.getChildren()) {
                    boolean hasWon = playerSnapshot.child("won").getValue(Boolean.class);
                    if (!hasWon) {
                        showLoseScreen();
                    } else{
                        //displayBingo();
                        //Toast.makeText(this, "Wygrałeś jak cos", Toast.LENGTH_SHORT).show();
                    }
                }
            }
        }
    });
}
```

# Co ciekawego w Firebase?



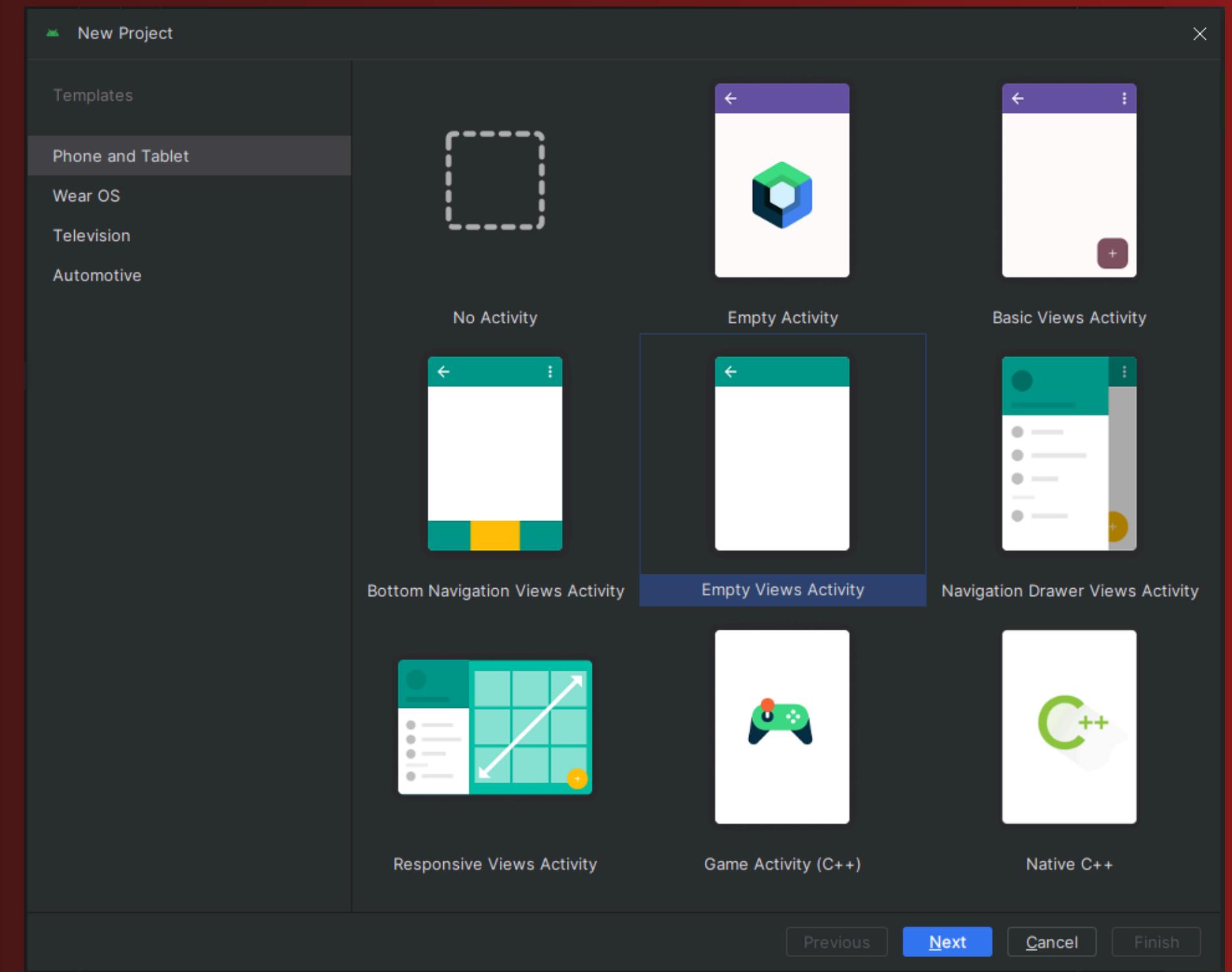
# Koszty implementacji Firebase

 Realtime Database	GB stored	1 GB about 20 M chat messages	No cost
	GB transferred	10 GB about 200 M chat messages	No cost
 Authentication	Phone Auth - All regions	Billed per SMS sent See <a href="#">current rates</a>	-
	Identity Platform Pricing Monthly active users (excluding SAML/OIDC)	50,000 MAUs Active users per month	No cost
	Monthly active users - SAML/OIDC	50 MAUs Active users per month	No cost



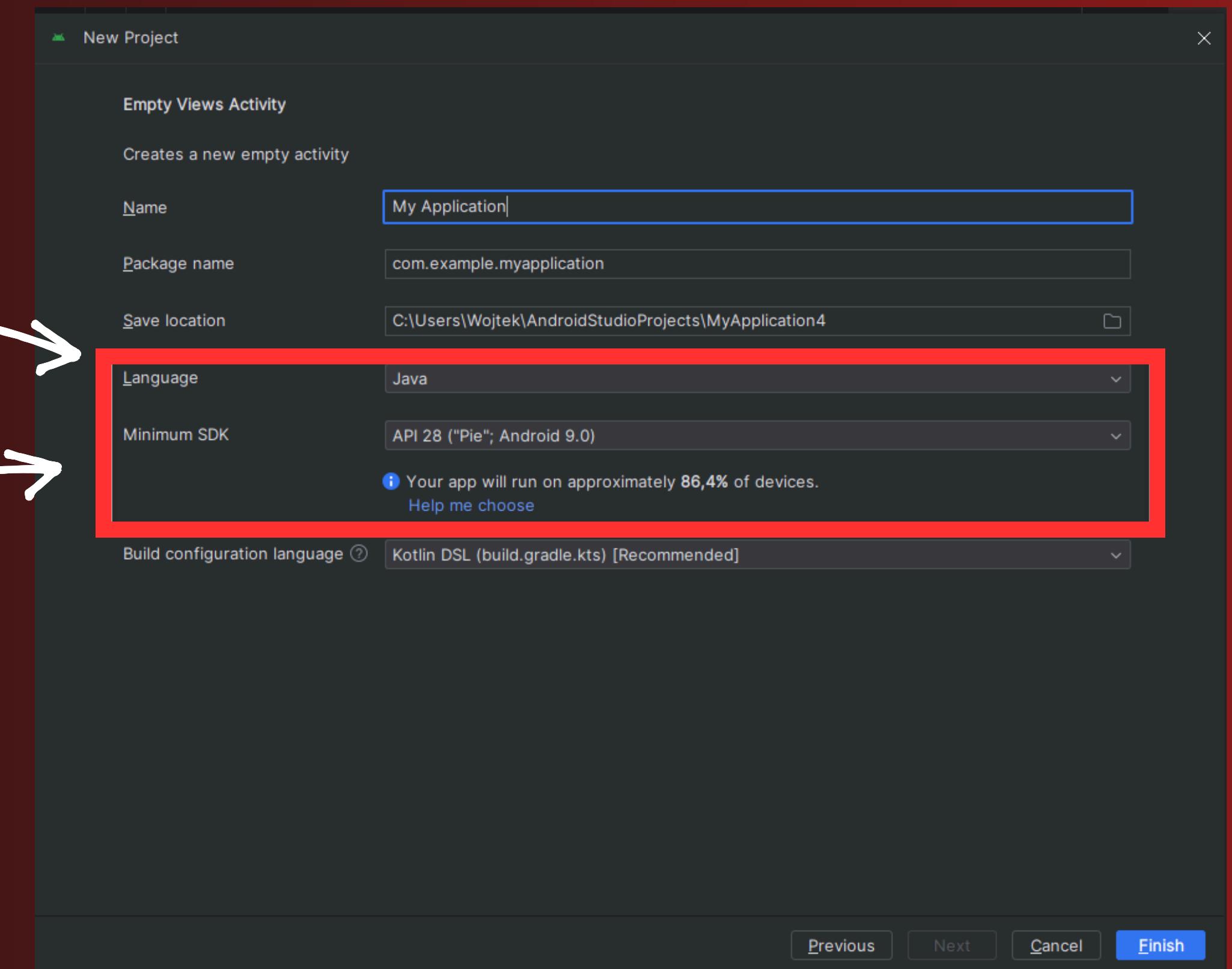
# Wstęp do części praktycznej

# Otwieramy nowy projekt - Empty Views Activity



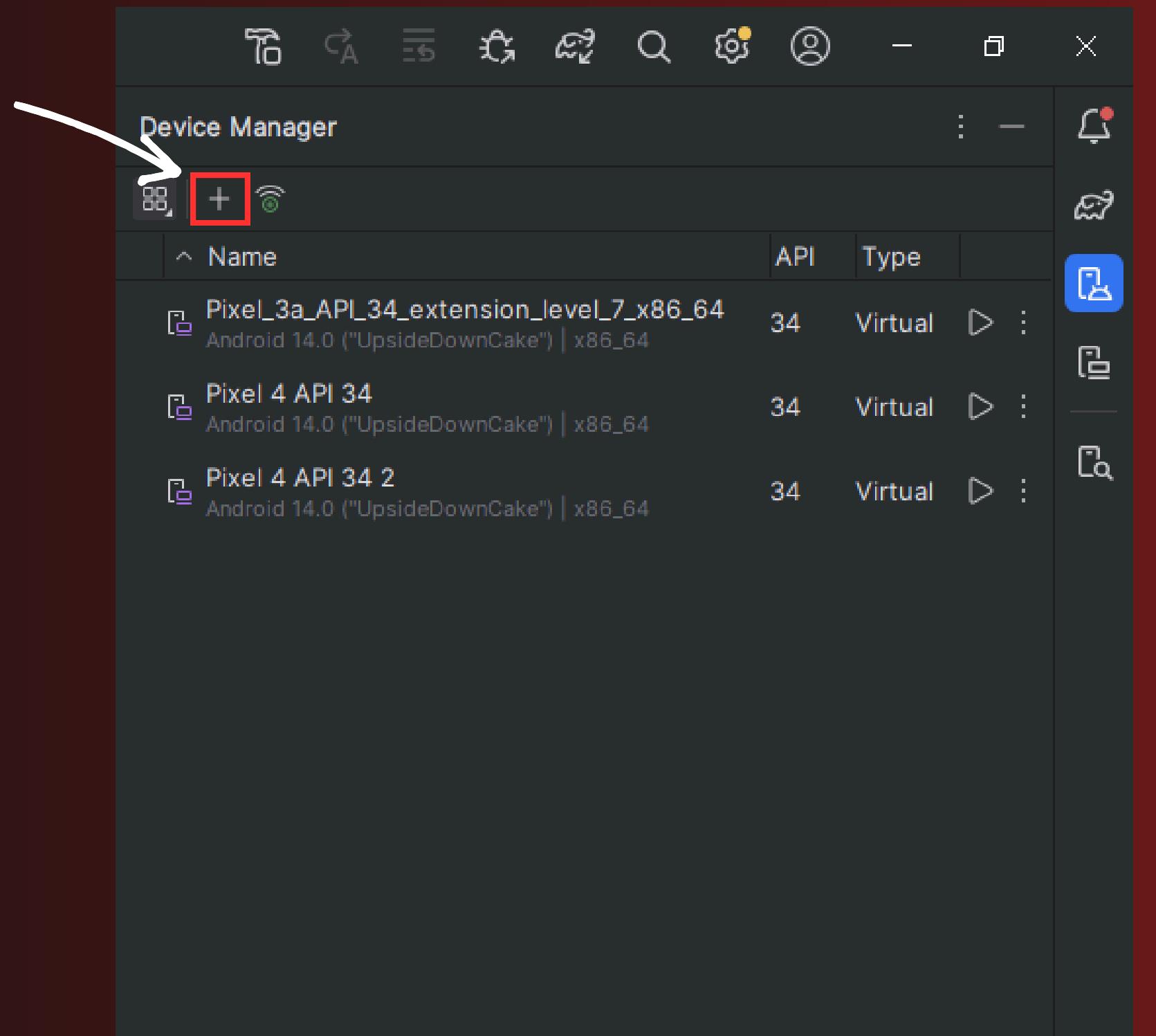
Wybieramy język Java

Wybieramy minimalną wymaganą wersję Androida na urządzeniu, by korzystał z aplikacji



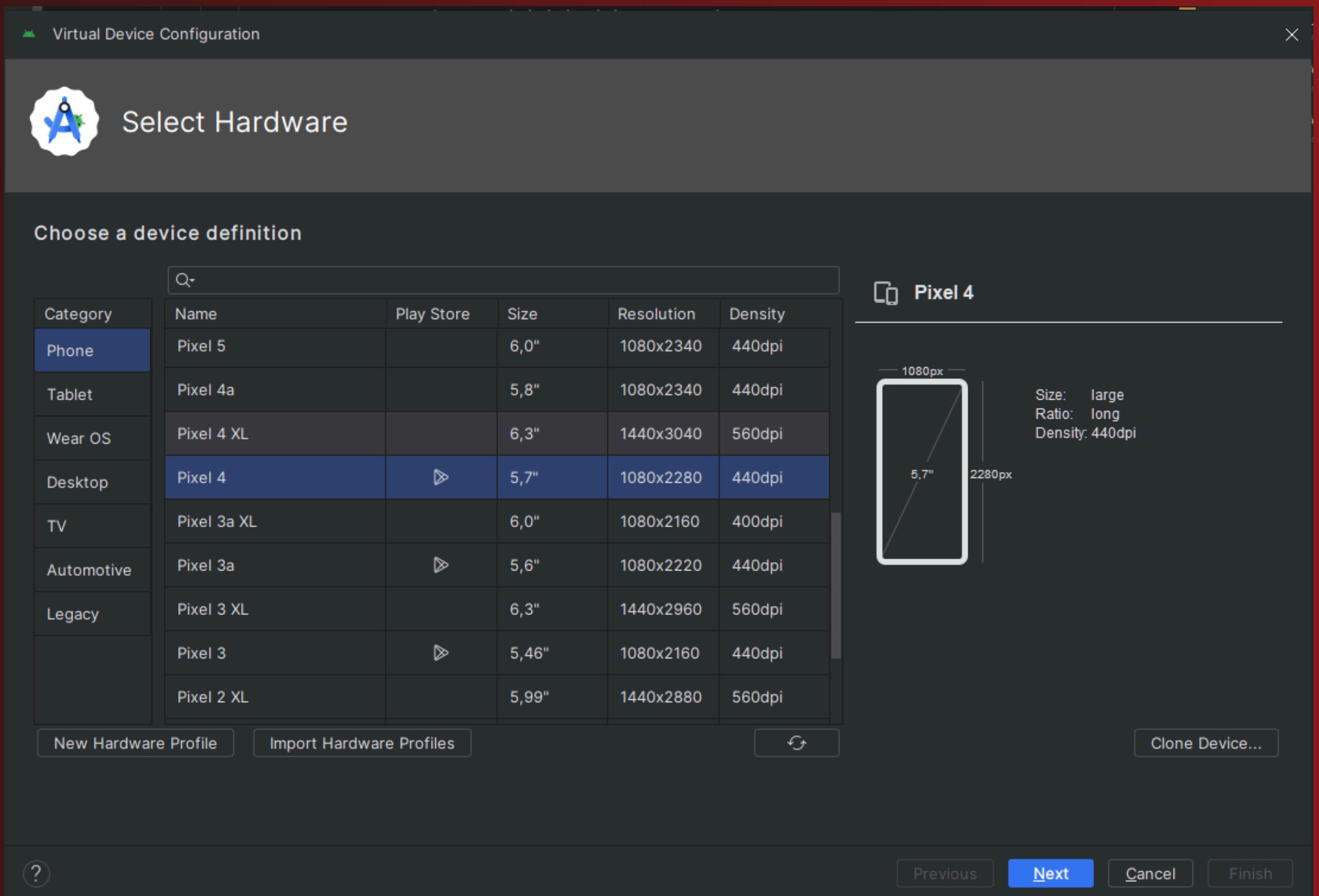
# Emulator Android

Create Virtual Device



Device Manager

# Wybór rodzaju urządzenia i modelu



Wybór wersji systemu  
Androida - wybrać taką  
by spełniała wymagania  
aplikacji

Virtual Device Configuration

### System Image

Select a system image

Recommended   x86 Images   Other Images

Release Name	API Level	ABI	Target
VanillaIceCream	VanillaIceCream	x86_64	Android API VanillaIceCream (Google Play)
UpsideDownCakePrivacyS...	UpsideDownCak...	x86_64	Android API UpsideDownCakeP...
TiramisuPrivacySandbox	TiramisuPrivacys...	x86_64	Android 14.0 (Google Play)
<b>UpsideDownCake</b>	<b>34</b>	<b>x86_64</b>	<b>Android 14.0 (Google Play)</b>
Tiramisu	33	x86_64	Android 13.0 (Google Play)
Sv2	32	x86_64	Android 12L (Google Play)
S	31	x86_64	Android 12.0 (Google Play)
R	30	x86	Android 11.0 (Google Play)
Q	29	x86	Android 10.0 (Google Play)
Pie	28	x86	Android 9.0 (Google Play)
Oreo	27	x86	Android 8.1 (Google Play)

UpsideDownCake

API Level  
**34**

Type  
**Google Play**

Android  
**14.0**

Google Inc.

System Image  
**x86\_64**

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?  
See the [API level distribution chart](#)

?

Previous   **Next**   Cancel   Finish

# Aplikacja na urządzeniu fizycznym

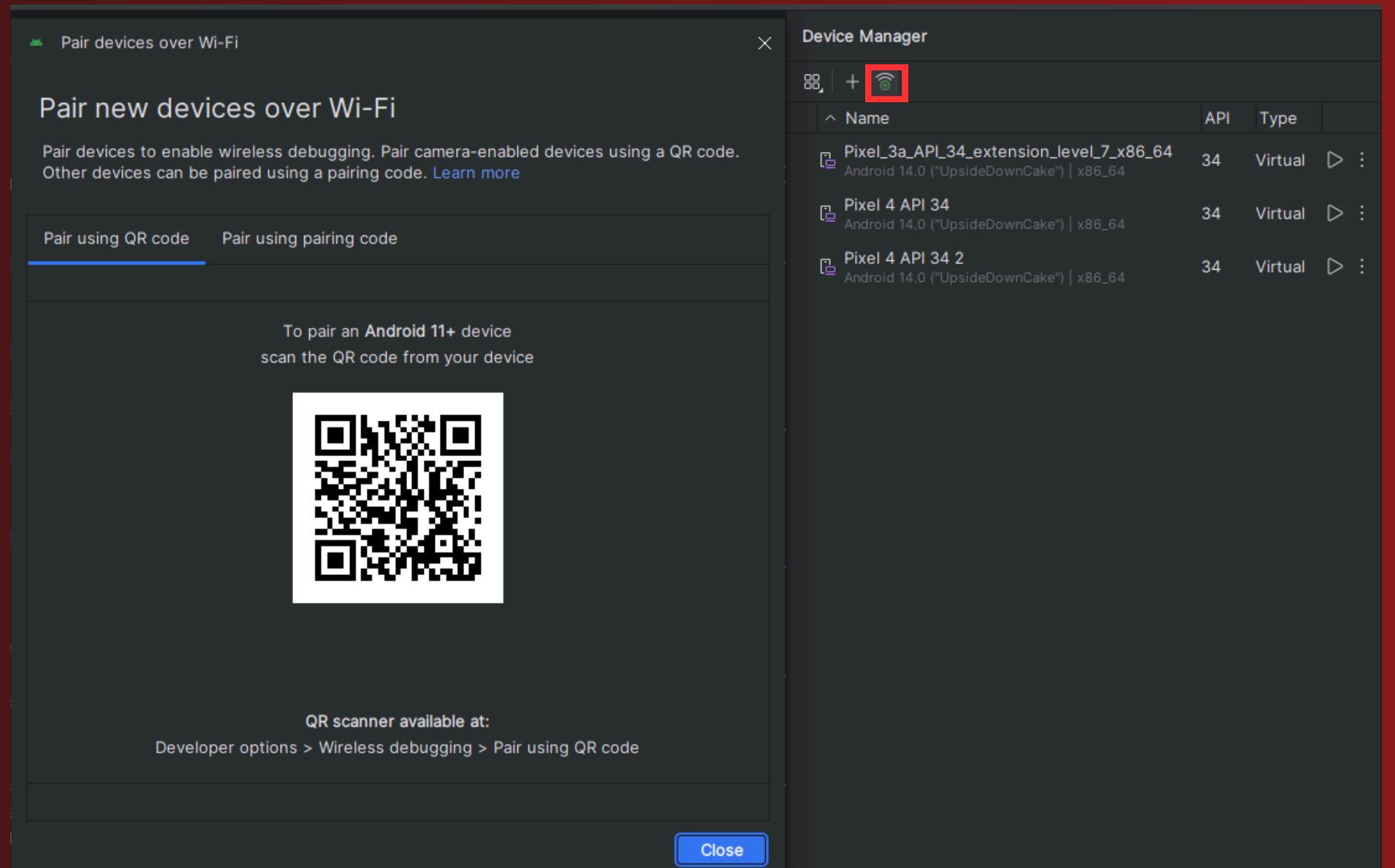
## Opcja przez USB

- Na urządzeniu należy wejść do Ustawienia --> Informacje o telefonie/O telefonie, następnie kliknąć 7 razy w Numer komilacji, aby od blokować opcje programisty (na niektórych urządzeniach może być trochę inaczej)
- Wejść w Ustawienia --> System --> Opcje programisty (może być to w innym miejscu w ustawieniach)
- Włączyć Debugowanie USB
- Podłączyć urządzenie kablem USB do komputera, w sekcji Device Manager powinien pojawić się podłączony telefon

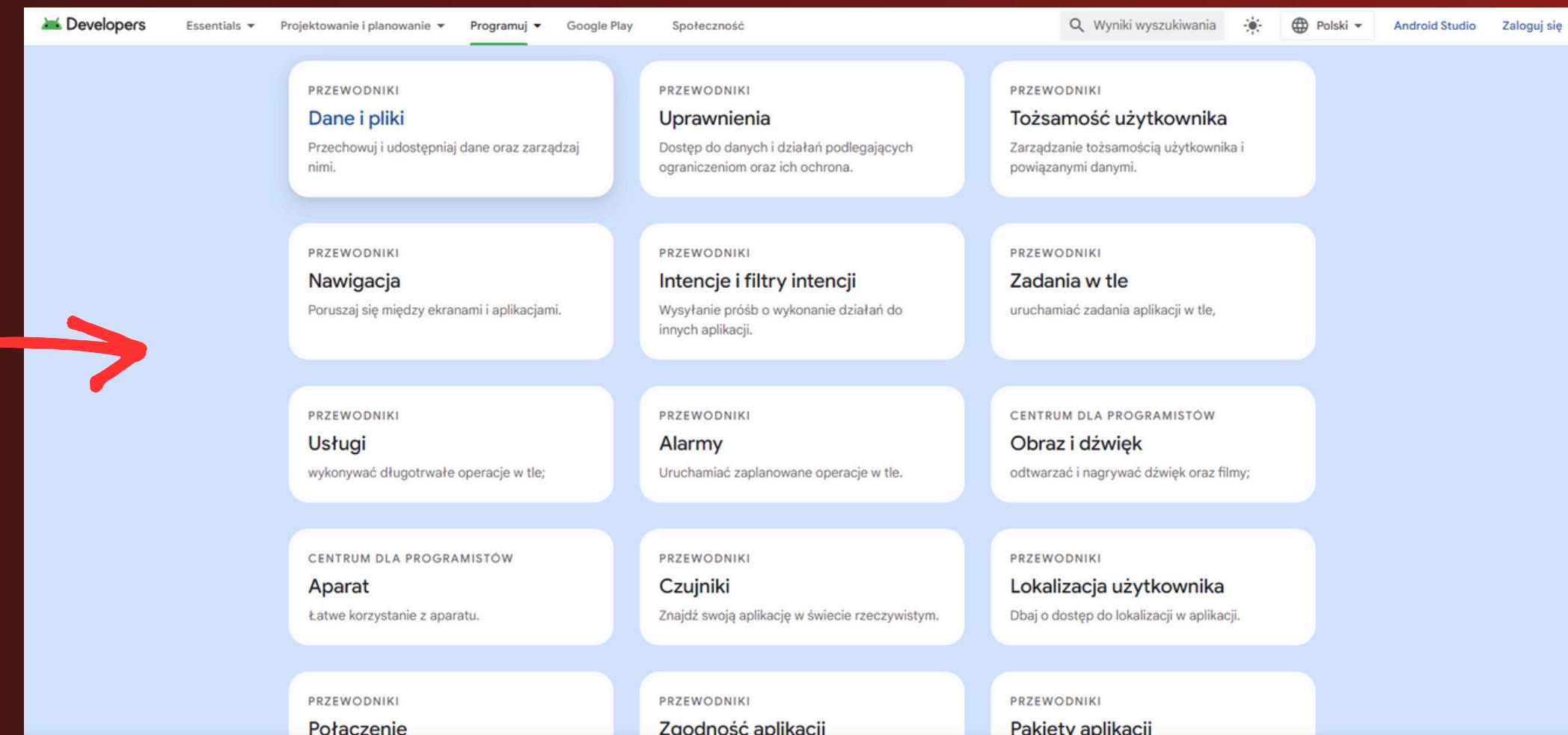
# Aplikacja na urządzeniu fizycznym

## Opcja przez Wifi

- Tak samo odblokować opcje programisty
- Wejść w Ustawienia --> System --> Opcje programisty (może być to w innym miejscu w ustawieniach)
- Włączyć Debugowanie bezprzewodowe
- W Device Manager wejść w opcję Pair Devices Using Wi-Fi i zeskanować kod QR wchodząc w Debugowanie bezprzewodowe --> Sparuj urządzenie przy pomocy kodu QR



Przydatnym może okazać się manual  
Android Studio  
<https://developer.android.com/?hl=pl>,  
gdzie w sekcji Programuj znajdują się  
różne poradniki, przewodniki



- Przykładowo w sekcji Połączenie --> Wi-fi Direct (P2P) można dowiedzieć się jak skonfigurować połączenie urządzeń używając opcji Wifi Direct
- Znajdują się tam fragmenty kodu, które należy dodać itp

**Przegląd Przewodniki**

Wszystkie główne obszary Przewodniki dotyczące połączeń

Dodaj obsługę protokołów

- Sieć
  - Wykonanie operacji sieciowych
- Wi-Fi
  - Skonfiguruj skanowanie Wi-Fi
  - Dodaj Wi-Fi Direct (P2P)
  - Wdroż Wi-Fi Aware
  - Konfigurowanie lokalizacji Wi-Fi przy użyciu interfejsu RTT API
  - Używaj hotspotu tylko lokalnie
  - Poproś o dostęp do urządzeń
- Odkrywaj i kontaktuj się
  - Informacje o wykrywaniu usług sieciowych (NSD)
  - Użyj NSD

**Tworzenie połączeń P2P za pomocą Wi-Fi Direct**

Używaj Wi-Fi Direct do wykrywania usług

**Konfigurowanie uprawnień aplikacji**

Aby używać Wi-Fi Direct, dodaj do pliku manifestu uprawnienia `ACCESS_FINE_LOCATION`, `CHANGE_WIFI_STATE`, `ACCESS_WIFI_STATE` i `INTERNET`. Jeśli Twoja aplikacja jest kierowana na Androida 13 (poziom interfejsu API 33) lub nowszego, dodaj też do pliku manifestu uprawnienie `NEARBY_WIFI_DEVICES`. Wi-Fi Direct nie wymaga połączenia z internetem, ale korzysta ze standardowych gniazd Java, które wymaga uprawnienia `INTERNET`. Aby korzystać z Wi-Fi Direct, musisz mieć te uprawnienia:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.android.nsdchat"
  ...
  <!-- If your app targets Android 13 (API level 33)
       or higher, you must declare the NEARBY_WIFI_DEVICES permission. -->
  <uses-permission android:name="android.permission.NEARBY_WIFI_DEVICES"
  <!-- If your app derives location information from Wi-Fi APIs,
       don't include the "usesPermissionFlags" attribute. -->
  android:usesPermissionFlags="neverForLocation" />

  <uses-permission
    android:required="true"
    android:name="android.permission.ACCESS_FINE_LOCATION"
    <!-- If any feature in your app relies on precise location information,
        don't include the "maxSdkVersion" attribute. -->
    android:maxSdkVersion="32" />
  <uses-permission
    android:required="true"
    android:name="android.permission.ACCESS_WIFI_STATE" />
  <uses-permission
    android:required="true"
    android:name="android.permission.CHANGE_WIFI_STATE" />
  <uses-permission
    android:required="true"
    android:name="android.permission.INTERNET" />
  ...
```

Oprócz poprzednich uprawnień następujące interfejsy API wymagają również włączenia trybu lokalizacji:

- `discoverPeers`
- `discoverServices`

**Na tej stronie**

- Konfigurowanie uprawnień aplikacji
- Konfigurowanie odbiornika i menedżera peer-to-peer
- Rozpocznij odkrywanie grupy porównawczej
- Pobierz listę aplikacji równorzędnych
- Połącz się z grupą porównawczą
- Tworzenie grupy