Uwagi wstępne

Niniejszy projekt będziemy rozwijać na zajęciach 6 i 7.

Projekt zakłada stworzenie aplikacji do grania w 2 osobowe gry (kółko i krzyżyk – TicTacToe, oraz 4 in a row)

Na ćwiczeniach laboratoryjnych nr 7 zadaniem będzie rozwinąć aplikację o obsługę gry kółko i krzyżyk.

Ostatnie zajęcia (lab 8) są ostatnią szansą na dokończenie i ulepszenie aplikacji, bądź oddanie zaległych projektów.

Do komunikacji będziemy wykorzystywać RESTful API dostępne na: games.antons.pl zwracające dane w postaci tabel JSON

Na końcu instrukcji znajduje się link do plików projektu – instrukcja po kolei opisuje czynności i kod realizujący dane czynności

Czym jest RESTful API?

ang. Representational State Transfer,

jest to zdalny dostęp do pewnych zasobów z wykorzystaniem pełnego protokołu http (czyli metody nie tylko POST i GET, ale także PUT, DELETE i inne)

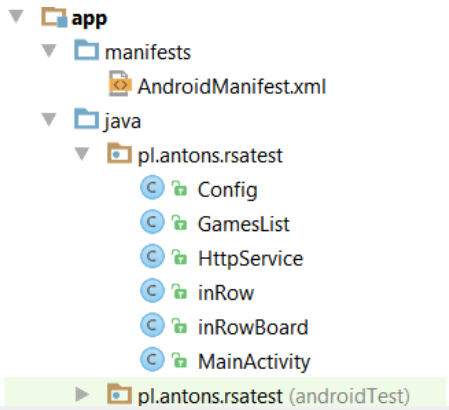
istotne dane przesyłane są także w nagłówkach http (headers)

za pomocą np. http://wst.mytechlabs.com/ *można przetestować zapytania do API (np. zbadać jego strukturę)*

każda gra jest pojedynczym zasobom udostępnianym przez API

Tworzenie struktury Aplikacji

1. Utwórz bazową Activity (Empty) MainActivity
2. Utwórz blank/basic Activity GamesList
3. Utwórz empty Activity „inRow”
4. Utwórz javaClass Config
5. Utwórz Service HttpService (podobnie jak się tworzy nowe Activity)
6. Utwórz nowy adapter (javaClass) „inRowBoard”



MainActivity będzie pozwalała wybrać użytkownikowi w którą grę chce zagrać

GamesList jest spisem aktywnie dostępnych gier do grania oraz możliwością stworzenia nowej gry

inRow będzie odpowiedzialna za wyświetlanie planszy i grę

klasa Config będzie tworzyć i obsługiwać bazę danych z konfiguracją aplikacji

HttpService jest usługą umożliwiającą zautomatyzowanie wysyłania żądań http wybraną metodą i odbioru tych zapytań.

inRowBoard jest adapterem do obsługi pojedynczej rozgrywki

Tworzenie zasobów

1. Utwórz w pliku res/values/strings.xml następujące zasoby tekstowe:
    a. refresh
    b. your_turn
    c. wait
    d. error
    e. connection
    f. network_error
    g. new_game
    h. empty_list
    i. win
    j. lose

```
<resources>
    <string name="app_name">HTTP Games</string>
    <string
name="title_activity_games_list">GamesList</string>
    <string name="refresh">Pobieranie listy
gier</string>
    <string name="your_turn">Twoja kolej.\nWybierz
kolumnę by dodać swój krążek.\n\n</string>
    <string name="wait">Poczekaj na
przeciwnika.\nRegularnie odświeżaj
planszę.\n\n</string>
    <string name="error">Ruch niemożliwy.\nSpróbuj
ponownie.\n\n</string>
    <string name="connection">Trwa przesyłanie
danych na serwer.\nPoczekaj chwilkę.\n\n</string>
    <string name="network_error">Wystąpił błąd z
siecią.\nWróć do listy dostępnych gier.\n\n</string>
    <string name="new_game">Rozpocznij nową
grę.\nWybierz kolumnę by dodać swój
krążek.\n\n</string>
<string name="empty_list">Niestety brak gier, w
które mógłbyś zagrać.\nUtwórz nową!</string>
<string name="win">Gratulacje, wygrałeś tą
rozgrywkę.\nZagraj ponownie!</string>
<string name="lose">Niestety, przegrałeś tą
rozgrywkę.\nZagraj ponownie!</string>
</resources>
```

2. Utwórz folder zasobów menu
3. Do foldera menu dodaj plik zasobu „game_menu”
4. Wprowadź do pliku game_menu.xml treść z kodu obok

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/an
droid"
    xmlns:app="http://schemas.android.com/apk/res-
auto">
    <item
        android:title="Odśwież"
        android:id="@+id/textBar"
     android:icon="@android:drawable/ic_popup_sync"
        android:onClick="refresh"
        app:showAsAction = "always|withText"
        />
</menu>
```

5. W folderze /res/drawable utwórz 3 pliki drawable resource file o nazwie „circle”, „player1”, „player2”

circle.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/a
pk/res/android">
    <item>
        <shape
xmlns:android="http://schemas.android.com/a
pk/res/android"
            android:shape="oval">
            <stroke
                android:width="1dp"
android:color="@color/colorPrimary"/>
        </shape>
    </item>
</selector>
```

player1.xml:

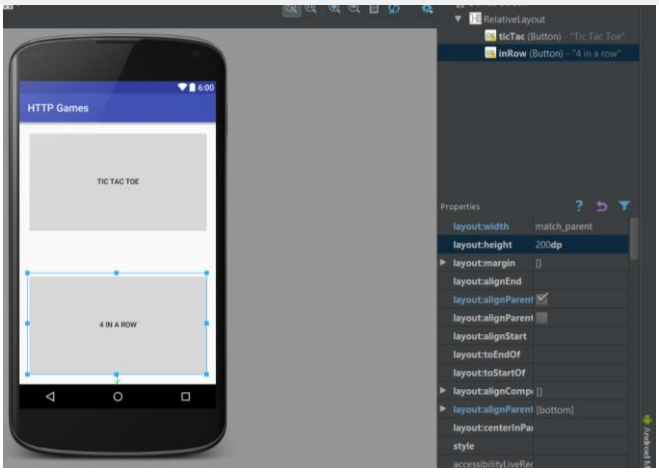
```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/apk/res/an
droid">
    <item>
        <shape
xmlns:android="http://schemas.android.com/apk/res/an
droid"
            android:shape="oval">
            <solid
android:color="@color/colorAccent"/>
        </shape>
    </item>
</selector>
```

player2.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/apk/res/an
droid">
    <item>
        <shape
xmlns:android="http://schemas.android.com/apk/res/an
droid"
            android:shape="oval">
            <solid
android:color="@color/material_deep_teal_500"/>
        </shape>
    </item>
</selector>
```

MainActivity

1. W bazowej aktywności utwórz 2 DUŻE przyciski wyboru gier:
   a. TicTacToe
   b. 4 in Row
2. Utwórz metodę public void gameList(View v) i przypisz ją do obydwu przycisków (parametr onClick)
3. Wewnątrz metody gameList dodaj kod tworzący intencję do uruchomienia Activity GamesList
4. Przy uruchamianiu dodaj „extras" o ID przycisku kliknięte przez użytkownika (by móc zidentyfikować listę których gier wyświetlić)




```
public void gameList(View v)
{
    Intent intencja = new Intent(
            getApplicationContext(),
            GamesList.class);
    intencja.putExtra("gra", v.getId());
    startActivity(intencja);
}
```

Config

1. Utwórz klasę Config dziedziczącą po SQLiteOpenHelper
2. Utwórz zmienną **private static final int *DATABASE_VERSION*** = 1;
3. W konstruktorze klasy wywołaj konstruktor rodzicielski z odpowiednimi parametrami
4. W metodzie onCreate utwórz strukturę tabeli config zawierającej 3 pola:
   a. _id integer primary key autoincrement
   b. privKey blob not null
   c. pubKey string not null
5. Wywołaj utworzenie tabeli
6. Wywołaj matodę, która uzupełni utworzoną tabelę

```
public Config(Context context) {
    super(context, "AppConfig",
            null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase
database) {
    String DATABASE_CREATE =
            "create table config " +
                    "(_id integer
primary key autoincrement," +
                    "privKey blob not
null," +
                    "pubKey string not
null);";

    database.execSQL(DATABASE_CREATE);
    addKeys(database);
}
```

7. Utwórz metodę uzupełniającą tabelę o nazwie addKeys przyjmującą obiekt bazodanowy
8. Wywołaj metodę generowania kluczy RSA (prywatnego i publicznego)
9. Przypisz klucz prywatny do odpowiedniego pola w bazie danych
10. Przypisz klucz publiczny do

```
private void addKeys(SQLiteDatabase db) {
    KeyPair keys = generateKeys();
    ContentValues values = new
ContentValues();
    values.put("privKey",
keys.getPrivate().getEncoded());
    values.put("pubKey",
Base64.encodeToString(keys.getPublic().getEnco
```

odpowiedniego pola w bazie danych (klucz powinien być przekonwertowany do postaci stringu base64)

```
ded(), Base64.DEFAULT));
    db.insert("config", null, values);
}
```

11. Utwórz metodę generującą parę kluczy RSA o nazwie generateKeys()
12. Ustaw algorytm na RSA
13. Ustaw długość klicza na 512

```
public static KeyPair generateKeys() {
    KeyPair keyPair = null;
    try {
        KeyPairGenerator keygen =
KeyPairGenerator.getInstance("RSA");
        keygen.initialize(512);
        keyPair = keygen.generateKeyPair();
    } catch (NoSuchAlgorithmException e) {
        e.printStackTrace();
        return null;
    }
    return keyPair;
}
```

14. Utwórz metodę getPublic zwracającą wartość klucza publicznego potrzebnego do weryfikacji podpisu

```
public String getPublic(){
    try {
        SQLiteDatabase db =
                this.getReadableDatabase();
        Cursor cursor =
                db.query("config",
                        new
String[]{"pubKey",},
                        null, null,
                        null, null,
                        null, null);
        if (cursor != null)
            cursor.moveToFirst();
        db.close();
        return cursor.getString(0);
    }catch(Exception ex){ }
    return null;
}
```

15. Utwórz metodę getPrivate zwracającą klucz prywatny przechowywany w bazie danych, celem dokonania podpisu elektronicznego
16. Pamiętaj o dokonaniu konwersji danych binarnych na obiekt klucza (PrivateKey)

```
private PrivateKey getPrivate(){
    SQLiteDatabase db =
            this.getReadableDatabase();
    Cursor cursor =
            db.query("config",
                    new String[] {
"privKey",},
                    null,null,
                    null,null,
                    null,null);
    if (cursor != null)
        cursor.moveToFirst();
    PrivateKey privateKey=null;
    try {
        KeyFactory kf =
KeyFactory.getInstance("RSA");
        privateKey = kf.generatePrivate(new
PKCS8EncodedKeySpec(cursor.getBlob(0)));
    }catch(Exception ex){}
    db.close();
    return privateKey;
}
```

17. Utwórz metodę wykonującą podpis cyfrowy za pomocą posiadanych kluczy RSA (kodowanie kluczem prywatnym) o nazwie sign
18. Wiadomość zaszyfrowaną kluczem prywatnym można odszyfrować jedynie posiadając klucz publiczny.

```
public String sign(String msg){
    try {
        Cipher cipher =
Cipher.getInstance("RSA/ECB/PKCS1Padding");
        cipher.init(Cipher.ENCRYPT_MODE,
getPrivate());
        String sign =
Base64.encodeToString(cipher.doFinal(msg.getBy
tes()), Base64.DEFAULT);
        return sign;
    }catch(Exception ex){}
    return null;
}
```

19. Dodaj metodę onUpgrade wywoływaną przy modyfikacji bazy danych

```
@Override
public void onUpgrade(SQLiteDatabase db, int
oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS config");
    onCreate(db);
}
```

HttpService

1. Utwórz Service HttpService dziedziczący po IntentService
2. Stwórz kilka zminnych statycznych final służące za flagi:
    a. int GAMES_LIST = 1
    b. int IN_ROW = 2
    c. int REFRESH = 3
    d. int GAME_INFO = 4
    e. String URL = „URL”
    f. String METHOD = „Method”
    g. String PARAMS = „Params”
    h. String RETURN = „Return”
    i. String RESPONSE = „Response”
    j. String LINES = http://games.antons.pl/lines/
    k. String XO = http://games.antons.pl/xo/
    l. int GET = 1
    m. int POST = 2
    n. int PUT = 3
3. Utwórz konstruktor, który wywoła konstruktor rodzicielski (super) z jakąś wartością

```
public class HttpService extends
IntentService {

public static final int GAMES_LIST = 1;
public static final int IN_ROW = 2;
public static final int REFRESH = 3;
public static final int GAME_INFO = 4;
public static final String URL = "URL";
public static final String METHOD =
"Method";
public static final String PARAMS =
"Params";
public static final String RETURN =
"Return";
public static final String RESPONSE =
"Response";
public static final String LINES =
"http://games.antons.pl/lines/";
public static final String XO =
"http://games.antons.pl/xo/";
public static final int GET = 1;
public static final int POST = 2;
public static final int PUT = 3;

    public HttpService() {
        super("HTTP calls handler");
    }
```

4. Przeciąż metodę onHandleIntent(Intent)
5. Utwórz obiekt URL
6. Przygotuj obiekt HttpURLConnection
7. Ustaw nagłówki żądania:
    a. Metoda
    b. Podpis URL
    c. Klucz Publiczny
    d. Parametry

```
String urlstr =
intent.getStringExtra(HttpService.URL);
URL url = new URL(urlstr);
HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
switch
(intent.getIntExtra(HttpService.METHOD,1)){
    case HttpService.POST:
        conn.setRequestMethod("POST");
        break;
    case HttpService.PUT:
        conn.setRequestMethod("PUT");
        break;
    default:
        conn.setRequestMethod("GET");
}
Config conf = new
Config(getApplicationContext());
conn.setRequestProperty("PKEY",
conf.getPublic().replace("\n",""));
conn.setRequestProperty("SIGN",conf.sign(urlst
r).replace("\n",""));

String params =
intent.getStringExtra(HttpService.PARAMS);
if(params!=null) {
    conn.setDoOutput(true);
    OutputStreamWriter writer = new
OutputStreamWriter(conn.getOutputStream());
    writer.write(params);
    writer.flush();
```

```
    writer.close();
}
```

8. Wyślij żądanie
9. Odbierz kod odpowiedzi
10. Utwórz bufor odczytu odpowiedzi

```
conn.connect();

int responseCode = conn.getResponseCode();

BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
```

11. Skonwertuj odpowiedź na String
12. Zamknij bufory odczytu odpowiedzi oraz połączenia

```
String response = "";
String line;
while ((line = reader.readLine()) != null) {
    response += line;
}
reader.close();
conn.disconnect();
```

13. Wyślij odpowiedź w Intencji jako wynik działania usługi
14. Obuduj polecenia blokiem try…catch

```
Intent returns = new Intent();
returns.putExtra(HttpService.RESPONSE,
response);
PendingIntent reply =
intent.getParcelableExtra(HttpService.RETURN);
reply.send(this, responseCode, returns);
```

inRowBoard

1. Utwórz adapter o nazwie inRowBoard dziedziczący po BaseAdapter
2. Utwórz 3 zmienne prywatne:
   a. Context context
   b. int player
   c. int[][] board = new int[6][7]
3. Utwórz konstruktor, który na podstawie Stringu z historią ruchów wygeneruje planszę do gry
   a. Rozbicie stringu z historią na poszczególne ruchy
   b. Wykonanie kolejnych ruchów, kolejnych graczy
4. Ustal którym graczem jest aktywny użytkownik

```
public class inRowBoard extends
BaseAdapter {
    private Context context;
    private int player;
    private int[][] board = new
int[6][7];

    public inRowBoard(Context cont,
String moves) {
        context=cont;

        int mvs = 0;
        for (String
move:moves.split("(?!^)")) {
            if(move!="")

this.move(Integer.parseInt(move),
mvs++%2);
        }
        player = mvs%2;
    }
```

5. Utwórz prywatną metodę symulującą ruch w grze move(int, int)
   a. Dla wybranej kolumny sprawdź, który wiersz jest wolny (wartość 0)
   b. Na wolnym elemencie wstaw wartość gracza (1 lub 2)

```
private boolean move(int col, int player)
{
    int row = 0;
    try {
        while (board[row][col] != 0)
            row++;
        board[row][col]=player+1;
    }catch(Exception ex){
        return false;
    }
    return true;
}
```

6. Utwórz publiczną metodę wykonania ruchu przez aktualnego gracza o nazwie add

```
public inRowBoard add(long col){
    if(this.move((int)col, player))
        return this;
    return null;
}
```

7. Przeciąż wymagane metody w adapterze
    a. int getCount()
    b. Object getItem(int)
    c. long getItemId(int)

```
@Override
public int getCount() {
    return 6*7;}
@Override
public Object getItem(int position){
    return position%7;}
@Override
public long getItemId(int position){
    return position%7;}
```

8. Przeciąż wymaganą metodę **public** View getView(**int** position, View convertView, ViewGroup parent) odpowiadającą za rysowanie pojedynczej komórki
9. Utwórz wewnątrz metody obiekt ImageView
10. Oblicz kolumnę i wiersz w tabeli board na podstawie zmiennej position
11. Ustaw odpowiedni obrazek w zależności od wartości planszy
    a. Pusto (0) – circle
    b. Gracz 1 (1) – player1
    c. Gracz 2 (2) – player2
12. Ustaw wielkość obrazka

```
@Override
public View getView(int position, View
convertView, ViewGroup parent) {
    ImageView iv = new ImageView(context);

    int col = position%7;
    int row = 5-position/7;

    switch (board[row][col]){
                case 0:
iv.setImageResource(R.drawable.circle);
                    break;
                case 1:
iv.setImageResource(R.drawable.player1);
                    break;
                case 2:
iv.setImageResource(R.drawable.player2);
                    break;
            }
    iv.setLayoutParams(new
LinearLayout.LayoutParams(120,120));
    return iv;
}
```

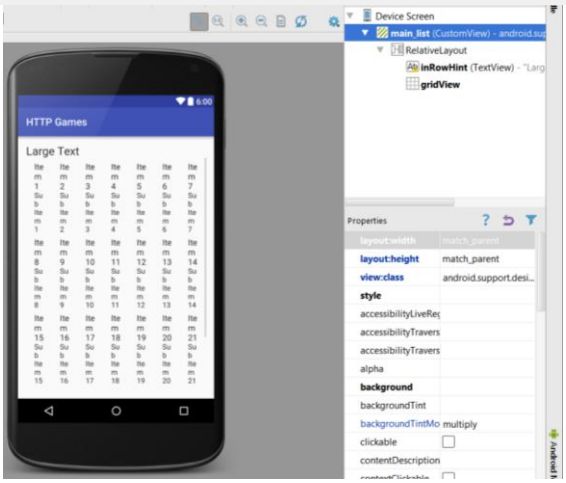
13. Stwórz metodę checkWin(), która sprawdzi, czy na planszy są obok siebie 4 takie same wartości
    a. W poszczególnych wierszach (poziomo)
    b. W poszczególnych kolumnach (pionowo)
    c. Horyzontalnie wznosząco
    d. Horyzontalnie opadająco
    e. Interesują nas tylko pola użytkowników
    f. Uważaj by nie wyjść poza zakres tablicy board
    g. Metoda powinna zwracać który gracz wygrał

```
public int checkWin(){
    int inRow = 0;

    for(int row=0; row<6; row++, inRow=0)
        for(int col=0;col<6;col++)

if(board[row][col]==board[row][col+1]) {
                inRow++;
                if(inRow==3 &&
board[row][col]!=0)
                    return board[row][col];
            }else
                inRow=0;

    for(int col=0;col<7;col++, inRow=0)
        for(int row=0;row<5;row++)

if(board[row][col]==board[row+1][col]){
                inRow++;
                if(inRow==3 &&
board[row][col]!=0)
                    return board[row][col];
            }else
                inRow=0;

    for(int posx=3; posx<6;posx++)
        for(int posy=0;posy<4;posy++) {
            inRow = 0;
            for (int x=posx, y=posy; x >0 &&
y < 6; x--, y++)
                if (board[x][y] == board[x -
1][y + 1]) {
                    inRow++;
                    if (inRow == 3 &&
board[x][y] != 0)
                        return board[x][y];
                } else
                    inRow = 0;
        }

    for(int posx=0; posx<3;posx++)
```

```
        for(int posy=0;posy<4;posy++) {
            inRow = 0;
            for (int x=posx, y=posy; x < 5 &&
y < 6; x++, y++)
                if (board[x][y] == board[x +
1][y + 1]) {
                    inRow++;
                    if (inRow == 3 &&
board[x][y] != 0)
                        return board[x][y];
                } else
                    inRow = 0;
        }
    return 0;
}
```

inRow

1. W widoku activity inRow utwórz kontrolkę zawierającą podpowiedzi do gry (id = inRowHint)
2. Pod kontrolką z podpowiedziami utwórz gridView, w którego parametrach ustaw numColumns na 7



3. W klasie inRow utwórz publiczne zmienne statyczne do sterowania procesem komunikacji:
    a. String STATUS
    b. String MOVES
    c. String GAME_ID
    d. String PLAYER
    e. int NEW_GAME
    f. int YOUR_TURN
    g. int WAIT
    h. int ERROR
    i. int CONNECTION
    j. int NETWORK_ERROR
    k. int WIN
    l. Int LOSE

```
public static final String STATUS = "Status";
public static final String MOVES = "Moves";
public static final String GAME_ID = "Game_id";
public static final String PLAYER = "Player";
public static final int NEW_GAME = 0;
public static final int YOUR_TURN = 1;
public static final int WAIT = 2;
public static final int ERROR = 3;
public static final int CONNECTION = 4;
public static final int NETWORK_ERROR = 5;
public static final int WIN = 6;
public static final int LOSE = 7;
```

4. Utwórz prywatne zmienne klasy reprezentujące trwającą grę:
    a. int status
    b. int game_id
    c. String moves
    d. int player

```
private int status;
private int game_id;
private String moves;
private int player;
```

5. W metodzie onCreate zainicjalizuj zmienne prywatne i uruchom komunikat początkowy

```
status=getIntent().getIntExtra(inRow.STATUS,
inRow.NEW_GAME);
game_id=getIntent().getIntExtra(inRow.GAME_ID,
inRow.NEW_GAME);
```

6. Utwórz i przypisz adapter inRowBoard do gridView
7. Stwórz listnera do obsługi przyciśnięcia elementu z gridView
8. Sprawdź, czy użytkownik może wykonać ruch (status jest różny od inRow.WAIT) (dalsze instrukcje tylko jeżeli może wykonać ruch)
9. Ustaw status na „ruch wykonany”
10. Ustaw komunikat na inRow.CONNECTION
11. Pobierz adapter inRowBoard z gridView (czyli aktywną grę)
12. Spróbuj wykonać ruch (metoda .add(kolumna))
13. Jeżeli ruch się wykona ponownie ustaw adapter do gridView, inaczej wyświetl komunikat błędu
14. Stwórz intencję dla usługi HTTPService
15. Utwórz PendingIntent jako intencję do odbioru wyniku działania usługi
16. Jeżeli użytkownik tworzy nową grę, ustaw w intencji Extras „URL” jako „LINES”, „METHOD” jako „POST”
17. Jeżeli gracz wykonuje ruch w istniejącej grze ustaw w intencji Extras „URL” jako „LINES”+game_id (czyli http://games.antons.pl/lines/id), „METHOD” jako „PUT”
18. Dodaj do intencji Extras „PARAMS” nową historię ruchów (w notacji moves=…)
19. Dodaj do intencji Extras „RETURN” utworzony wcześniej PendingIntent
20. Uruchom usługę

21. Dla metody onActivityResult (do obsługi danych powracających z usługi) utwórz dwa stany – jeżeli *requestCode==HTTPService.IN_ROW* oraz *requestCode==HTTPService.REFRESH*

```
moves = getIntent().getStringExtra(inRow.MOVES);
player=getIntent().getIntExtra(inRow.PLAYER, 1);
hints(status);

GridView gv=(GridView)findViewById(R.id.gridView);
gv.setAdapter(new inRowBoard(this,moves));


gv.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> arg0, View
arg1, int arg2, long arg3) {

        if(status!=inRow.WAIT)
        {
            status = inRow.WAIT;
            hints(inRow.CONNECTION);

            GridView gv = (GridView)
findViewById(R.id.gridView);
            inRowBoard game =
(inRowBoard)gv.getAdapter();

            if(game.add(arg3)!=null)
                gv.setAdapter(game);
            else
                hints(inRow.ERROR);

            Intent intencja = new Intent(
                    getApplicationContext(),
                    HttpService.class);

            PendingIntent pendingResult =
createPendingResult(HttpService.IN_ROW, new
Intent(),0);

            if(game_id == inRow.NEW_GAME)
            {
                intencja.putExtra(HttpService.URL,
HttpService.LINES);
                intencja.putExtra(HttpService.METHOD,
HttpService.POST);

            }else{
                intencja.putExtra(HttpService.URL,
HttpService.LINES+game_id);

intencja.putExtra(HttpService.METHOD,
HttpService.PUT);
            }

            intencja.putExtra(HttpService.PARAMS,
"moves=" + moves + arg3 );

       intencja.putExtra(HttpService.RETURN,
pendingResult);

            startService(intencja);
        }
    }
});
```

```
@Override
protected void onActivityResult(int requestCode, int
resultCode, Intent data)
{
    if(requestCode==HttpService.IN_ROW)
    {
...
}else
if(requestCode==HttpService.REFRESH){}...
```

22. Dla stanu HTTPService.REFRESH utwórz obiect JSONObject którzy sparsuje dane z intencji (HTTPService.RESPONSE)

```
JSONObject response = new
JSONObject(data.getStringExtra(HttpService.RESPONSE));
```

23. Pobierz z obiektu JSON pole „moves” (ruchy) i przypisz do zmiennej prywatnej
24. Ustaw nowy adapter inRowBoard do gridView

```
moves = response.getString("moves");
GridView gv = (GridView) findViewById(R.id.gridView);
inRowBoard game = new inRowBoard(this,moves);
gv.setAdapter(game);
```

25. Sprawdź, czy jest ruch gracza („status” odpowiedzi jest taki sam jak nr gracza)
26. Jeżeli tak, sprawdź czy gracz wygrał, czy przegrał, jeżeli gra ciągle się toczy ustaw status gry na inRow.YOUR_TURN i wyświetl odpowiedni kounikat
27. Jeżeli nie jest ruch gracza odczekaj 5s i odśwież grę
28. Obuduj wszystko blokiem try{}catch{}

```
if(response.getInt("status")==player){
    if(game.checkWin()==player) {
        hints(inRow.WIN);
    }else if(game.checkWin()!=0){
        hints(inRow.LOSE);
    }else {
        status = inRow.YOUR_TURN;
        hints(status);
    }
}else{
    Thread.sleep(5000);
    refresh(null);
}
```

29. Dla stanu *HTTPService.IN_ROW utwórz obiekt JSONObject odpowiedzi z usługi*
30. Sprawdź, czy nie wystąpił błąd (resultCode inny niż 200), jeżeli wystąpił wyświetl odpowiedni błąd
31. Jeżeli wszystko OK pobierz game_id z obiektu JSON o tej samej nazwie (pod warunkiem, że istnieje)
32. Następnie pobierz adapter inRowBoard z gridView
33. Sprawdź, status gry (czy wygrana, przegrana czy trwa)
34. Wyświetl odpowiedni komunikat zależny od statusu gry
35. Na końcu odczekaj 5s i odśwież grę

```
JSONObject response = new
JSONObject(data.getStringExtra(HttpService.RESPONSE));
if (resultCode == 200) {

    if(game_id==0)
        game_id = response.getInt("game_id");

    GridView gv = (GridView)
findViewById(R.id.gridView);
    inRowBoard game = (inRowBoard)gv.getAdapter();
    int game_status = game.checkWin();
    if (game_status==0)
            hints(inRow.WAIT);
    else{
        if(game_status==player)
            hints(inRow.WIN);
        else
            hints(inRow.LOSE);
    }

} else{
    if(resultCode==500)
        hints(inRow.NETWORK_ERROR);
    else
        hints(inRow.ERROR);
}
Thread.sleep(5000);
refresh(null);
```

36. Utwórz metodę tworzenia menu
37. Przypisz zasób game_menu do menu aktywności

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.game_menu, menu);
    return true;
}
```

38. Stwórz funkcję wywoływaną przyciskiem z menu „refresh”, która będzie odświeżać grę
39. Utwórz intencję do uruchomienia usługi HTTPService

```
public void refresh(MenuItem item){
   Intent intencja = new Intent(
            getApplicationContext(),
            HttpService.class);

    PendingIntent pendingResult =
createPendingResult(HttpService.REFRESH, new
Intent(),0);
```

40. Utwórz obiekt PendingIntent jako intencję do odbioru komunikatów z serwisu
41. Dodaj parametry jako Extras:
    a. URL
    b. METHOD
    c. RETURN
42. Uruchom usługę

```
    intencja.putExtra(HttpService.URL,
HttpService.LINES+game_id);
    //Set data - method of request
    intencja.putExtra(HttpService.METHOD,
HttpService.GET);
    //Set data - intent for result
    intencja.putExtra(HttpService.RETURN,
pendingResult);
    //Start unBound Service in another Thread
    startService(intencja);
}
```

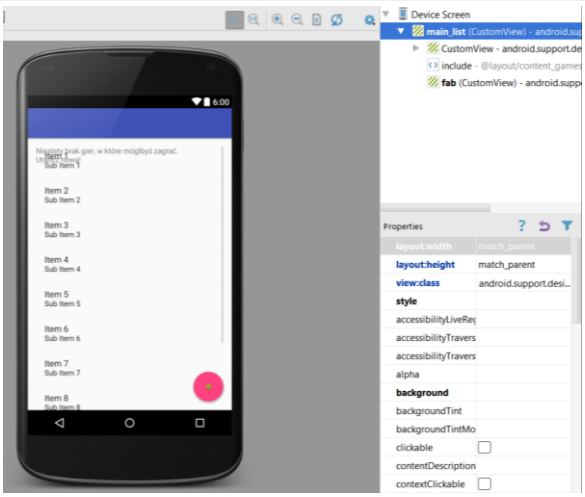
43. Utwórz pomocniczą metodę do wyświetlania komunikatów „hints”
44. Pobierz kontrolkę (TextView) hint
45. W zależności od przekazanego statusu ustaw test z odpowiedniego zasobu do kontrolki inRowHint

```
private void hints(int status){
    TextView hint =
(TextView)findViewById(R.id.inRowHint);
    switch(status){
        case inRow.YOUR_TURN:

hint.setText(getString(R.string.your_turn));
            break;
        case inRow.WAIT:
            hint.setText(getString(R.string.wait));
            break;
        case inRow.ERROR:
            hint.setText(getString(R.string.error));
            break;
        case inRow.CONNECTION:

hint.setText(getString(R.string.connection));
            break;
        case inRow.NETWORK_ERROR:

hint.setText(getString(R.string.network_error));
            break;
        case inRow.WIN:
            hint.setText(getString(R.string.win));
            break;
        case inRow.LOSE:
            hint.setText(getString(R.string.lose));
            break;
        default:

hint.setText(getString(R.string.new_game));
            break;
    }
}
```

GamesList

1. Zmień ikonkę FloatingButtona na plus (ic_input_add)
2. W Content_games_list dodaj kontener ListView rozciągniętą na całą długość i szerokość dostępnego ekranu (wartość fill_parent)
3. Dodaj kontrolkę TextView o id empty i text pobrany z zasobu empty_list
4. Dodaj kontrolkę ProgressBar:

```
<ProgressBar
    android:id="@+id/progressBar1"
  style="?android:attr/progressBarStyleLarge"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true" />
```



5. Obuduj kontrolkę ListView następującym tagiem (pamiętaj aby go później

zamknąć:

```
<android.support.v4.widget.SwipeRefreshLayout

xmlns:android="http://schemas.android.com/apk/
res/android"
    android:id="@+id/swipe_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

6. W pliku .java, dodaj globalną (dla klasy) zmienną private int game i w metodzie onCreate przypisz do niej wartość Extra „gra”

```
Bundle extras =
getIntent().getExtras();
game = extras.getInt("gra");
```

7. Stwórz listner do odświeżania listy w metodzie onCreate:

```
SwipeRefreshLayout swipeLayout =
(SwipeRefreshLayout)
findViewById(R.id.swipe_container);
swipeLayout.setOnRefreshListener(
        new
SwipeRefreshLayout.OnRefreshListener() {
            @Override
            public void onRefresh() {
                refreshGameList();
            }
        }
);
```

8. Również w onCreate uruchom (nieistniejącą póki co) metodę refreshGameList();
9. Zmień akcję wywoływaną po kliknięciu na floating action buton na kod z prawej kolumny (uruchomienie nowej gry)

```
Intent intencja=null;
switch (game) {
    case R.id.inRow:
        intencja = new
Intent(getApplicationContext(), inRow.class);
intencja.putExtra(inRow.STATUS,
inRow.NEW_GAME);
intencja.putExtra(inRow.MOVES, "");
        break;
    default:
        //TODO - when gamer choose TicTacToe
Game
        break;
}
startActivity(intencja);
```

10. Do metody onCreate dodaj jeszcze listner obsługi kliknięcia na listę gier
11. W listnerze tym:
    - Pokaż progressBar (staw jego widoczność na VISIBLE)
    - Przetwórz game_id wybranej gry z etykiety kliknięte opcji
    - Utwórz intencję do obsługi serwisu HTTPService
    - Utwórz obiekt PendingIntent by móc odebrać komunikat zwracany przez usługę
    - W zależności od wybranej przez gracza gry ustaw parametr Extras „URL”
    - Ustaw parametr METHOD
    - Usatw parametr RETURN
    - Uruchom usługę

```
ListView list =
(ListView)findViewById(R.id.listView);
list.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?>
arg0, View arg1, int arg2, long arg3) {

        ProgressBar spinner =
(ProgressBar)findViewById(R.id.progressBar1);
        spinner.setVisibility(View.VISIBLE);

        String game_id =
arg0.getItemAtPosition(arg2).toString().replac
e("ID: ","");

        Intent intencja = new Intent(
                getApplicationContext(),
                HttpService.class);

        PendingIntent pendingResult =
createPendingResult(HttpService.GAME_INFO, new
Intent(),0);

        if(game == R.id.inRow){
            intencja.putExtra(HttpService.URL,
HttpService.LINES+game_id);
        }else{
            //TODO - geting ticTacToe games
list
```

12. Utwórz metodę do odświeżania listy o nazwie refreshGameList()
    - Ustaw progressBar by był widoczny
    - Ustaw komunikat wskazujący, że aplikacja pobiera listę dostępnych gier z serwera (za pomocą zasobu string)
    - Utwórz intencję dla usługi HttpService
    - Utwórz obiekt PendingIntent do obsługi komunikatu powracającego z usługi
    - Wprowadź parametry żądania http, takie jak:
        i. URL (zależny od wybranej gry)
        ii. Metoda żądania
        iii. Intencja do odbioru zapytania
    - (wszystkie zmienne są dostępne z poziomu Serwisu HttpService)
    - Uruchom serwis

```
        }

        intencja.putExtra(HttpService.METHOD,
HttpService.GET);
        intencja.putExtra(HttpService.RETURN,
pendingResult);
        startService(intencja);
    }
});

public void refreshGameList(){
    ProgressBar spinner =
(ProgressBar)findViewById(R.id.progressBar1);
spinner.setVisibility(View.VISIBLE);
Snackbar.make(findViewById(R.id.main_list),
getString(R.string.refresh),
Snackbar.LENGTH_SHORT)
        .setAction("Action", null).show();

Intent intencja = new Intent(
        getApplicationContext(),
        HttpService.class);
PendingIntent pendingResult =
createPendingResult(HttpService.GAMES_LIST,
new Intent(),0);
if(game == R.id.inRow){
    intencja.putExtra(HttpService.URL,
HttpService.LINES);
}else{
    //TODO - geting ticTacToe games list
}
intencja.putExtra(HttpService.METHOD,
HttpService.GET);
intencja.putExtra(HttpService.RETURN,
pendingResult);
startService(intencja);
}
```

13. Stwórz metodę **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) by móc odebrać komunikat zwrócony przez HttpService i wyświetlić listę gier
14. Metoda ta działa w dwóch trybach:
    - HTTPService.GAMES_LIST
    - HTTPService.GAME_INFO
15. Dla trybu HTTPService.GAMES_LIST:
    - Ukryj progressBar ładowania
    - Utwórz obiekt JSONObject z odpowiedzi usługi
    - Dalsze czynności tylko, gdy dostępnych dla użytkownika gier jest więcej niż 0
    - Ukryj wiadomość o braku dostępnych gier
    - Pobierz obiekt JSONArray z obiektu „games”
    - Sparsuj tabelę z grami do postaci Listy Stringów
    - Ustaw stworzoną listę stringów

```
ProgressBar spinner = (ProgressBar)
findViewById(R.id.progressBar1);
spinner.setVisibility(View.GONE);
SwipeRefreshLayout swipeLayout =
(SwipeRefreshLayout)
findViewById(R.id.swipe_container);
swipeLayout.setRefreshing(false);

try {
    JSONObject response = new
JSONObject(data.getStringExtra(HttpService.RES
PONSE));

    if(response.getInt("games_count")>0)
    {
        TextView no_game =
(TextView)findViewById(R.id.empty);
        no_game.setVisibility(View.GONE);

        JSONArray games = new
JSONArray(response.getString("games"));
        ArrayList<String> items = new
ArrayList<String>();

        for(int i=0;
i<response.getInt("games_count");i++){
            JSONObject game =
games.getJSONObject(i);
            items.add("ID:
"+game.getString("id"));
        }

        ArrayAdapter<String> gamesAdapter =
```

jako ArrayAdapter i przypisz go do listView

```
                new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, items);
        ListView list =
(ListView)findViewById(R.id.listView);
        list.setAdapter(gamesAdapter);
    }
}catch (Exception ex){
    ex.printStackTrace();
}

ProgressBar spinner = (ProgressBar)
findViewById(R.id.progressBar1);
spinner.setVisibility(View.GONE);

if(game==R.id.inRow) {
    Intent intencja = new
Intent(getApplicationContext(), inRow.class);
    try {
        JSONObject response = new
JSONObject(data.getStringExtra(HttpService.RES
PONSE));

        intencja.putExtra(inRow.GAME_ID,
response.getInt("id"));

        if (response.getInt("status") == 0 &&
response.getInt("player1") == 2) {
            intencja.putExtra(inRow.STATUS,
inRow.YOUR_TURN);
        } else if (response.getInt("status")
== 1 && response.getInt("player1") == 1) {
            intencja.putExtra(inRow.STATUS,
inRow.YOUR_TURN);
        } else if (response.getInt("status")
== 2 && response.getInt("player1") == 2) {
            intencja.putExtra(inRow.STATUS,
inRow.YOUR_TURN);
        } else
            intencja.putExtra(inRow.STATUS,
inRow.WAIT);

        intencja.putExtra(inRow.PLAYER,
response.getInt("player1"));

        intencja.putExtra(inRow.MOVES,
response.getString("moves"));
        startActivity(intencja);

    } catch (Exception ex) {
        ex.printStackTrace();
    }
}else if(game==R.id.ticTac){
    //TODO - start chosen game for TicTacToe
}
```

16. Dla trybu HTTPService.GAME_INFO:
    - Ukryj progressBar
    - Sprawdź jaka gra została wybrana
    - Utwórz Intencję dla wybranej gry
    - W bloku try…catch utwórz obiekt JSON z odpowiedzi usługi
    - Ustaw parametry intencji wywołania gry takie jak:
        i. GAME_ID
        ii. STATUS
        iii. PLAYER
        iv. MOVES
    - Do ustawienia tych parametrów wykorzystaj dane pobrane z obiektu JSON
    - Jako status należy wyliczyć czy aktualnie jest ruch gracza
    - Uruchom usługę

AndroidManifest

1. Dodatkowo dodaj do pliku AndroidManifest.xml uprawnienia aplikacji do korzystania z sieci:

```
<uses-permission android:name=
"android.permission.INTERNET"
/>
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="pl.antons.rsatest">

    <uses-permission
android:name="android.permission.INTERNET" />

    <application
        . . .
    </application>

</manifest>
```

Uwagi końcowe

Kody wraz z komentarzami do kompletnego projektu dostępne są na https://github.com/asmolinski/GamesAPIClient

W w/w projekcie wykorzystano starą (niedziałającą obecnie) lokalizację API antons.pl/games aby dany kod poprawnie komunikował się z serwerem należy zmienić adres na games.antons.pl pozostała struktura endpointów pozostała bez zmian.