Aplicație Android

Adelina Maria Chirtes
Grupa 30233

Cuprins

Introducere

1. Context

Obiectivul acestei lucrari este realizarea unei aplicatii pentru telefoane mobile cu sistem de operare Android, facilitatile de programare oferite de acest SO. Limbajul de programare este Java. Vom explora facilitățile oferite de Android pentru dezvoltatori și vom implementa trei programe simple, de la un "Hello World" până la aplicații mai complexe, cum ar fi realizarea unui calculator simplu sau o aplicație care reprezinta un joc. Prin intermediul acestei lucrari practice, voi obține o înțelegere mai profundă a funcționalităților Android și voi putea să construiesc aplicații mai complexe în viitor.

2. Specificatii

Platforma de Dezvoltare: Android OS

Limbaj de Programare: Java

Nivel de Complexitate: Creșterea progresivă a complexității de la aplicații simple la cele mai complexe, în trei etape distincte.

3. Obiective

- Familiarizare cu Android Studio și Mediul de Dezvoltare
- Înțelegerea mediului de dezvoltare Android Studio și a resurselor disponibile pentru programatori.
- Competențe în Programarea Android cu Java
  Dezvoltarea abilităților în limbajul de programare Java, adaptat pentru platforma Android.
- Implementarea Funcționalităților Specifice
  Realizarea cu succes a fiecărei etape a proiectului, de la aplicații simple la cele mai complexe, demonstrând o înțelegere progresivă a facilităților Android.
- Creșterea Graduală a Complexității
  Abordarea progresivă a sarcinilor, de la cele mai simple la cele mai complexe, pentru a asigura o înțelegere detaliată a fiecărui pas.
- Experiență în Dezvoltarea Aplicațiilor Mobile Android
  Dobândirea experienței practice în dezvoltarea de aplicații Android, care poate servi ca bază pentru proiecte viitoare

Studiu bibliografic

Studiul bibliografic are ca obiectiv să ofere o perspectivă cuprinzătoare asupra dezvoltării de aplicații mobile pentru platforma Android, utilizând limbajul de programare Java. Proiectul propus urmărește trei etape distincte, începând de la o aplicație simplă "Hello World" și ajungând la aplicații mai complexe, precum partajarea de fotografii pe Facebook și monitorizarea procentajului bateriei telefonului.

Algoritmul se bazeaza pe cativa pasi importanti:

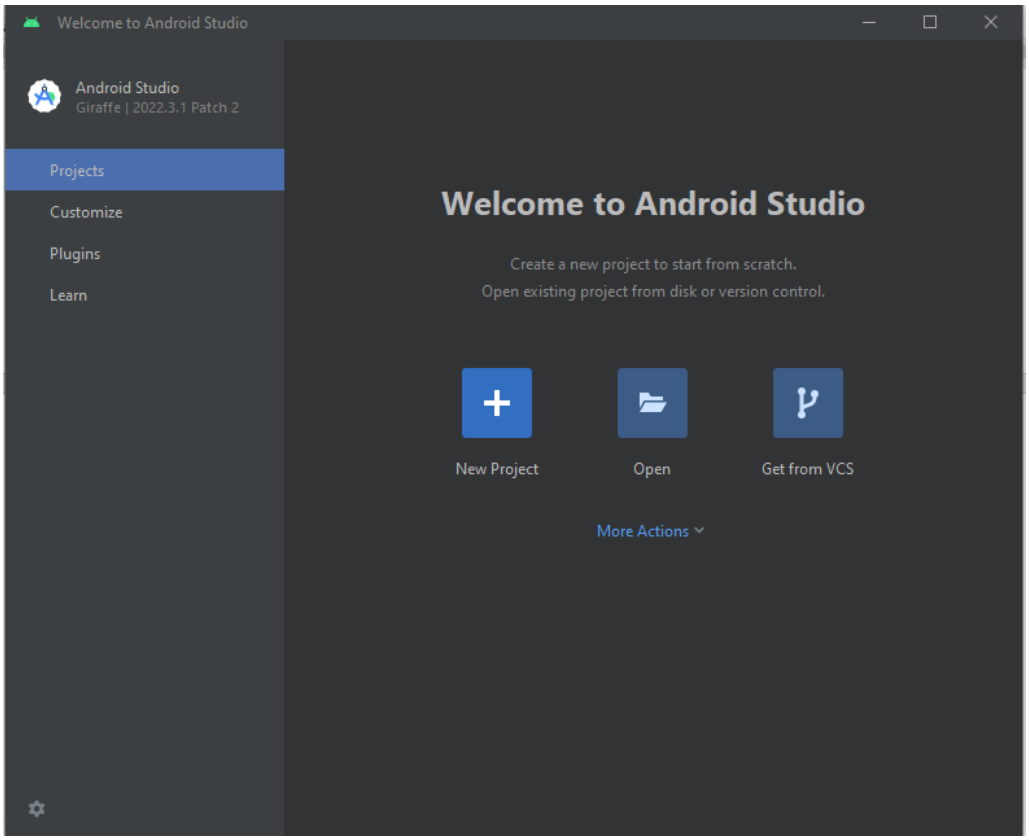
- Înțelegerea mediului de dezvoltare Android Studio și a platformei Android ca suport pentru dezvoltarea de aplicații mobile.
- Studierea limbajului de programare Java și modul în care este utilizat în dezvoltarea de aplicații mobile Android.
- Explorarea modului în care aplicațiile Android pot accesa și utiliza camera dispozitivului pentru capturarea de imagini.
- Înțelegerea modului în care aplicațiile Android pot interacționa cu platforma Facebook pentru partajarea de conținut
- Studierea API-urilor Android pentru a obține informații precise despre starea și nivelul bateriei dispozitivului.

Instalare Android Studio

Pentru aceasta lucrare trebuie sa instalam o aplicatie de tip Android, cum ar fi “Android Studio”. Aceasta aplicatie se gasesete in ultima versiune si o sa fie descarcata de pe urmatorul link: https://developer.android.com/studio, indiferent ce system de operare folosim.

Dupa ce s-a finalizat descarcarea, se pot urmari pasii de pe acest link pentru a face toate setarile de care avem nevoie pentru instalare https://www.youtube.com/watch?v=DM783YA0vbc&ab_channel=GeekyScript (pentru Windows 10/11, dar se gasesc informatii asemanatoare pentru toate sistemele de operare).

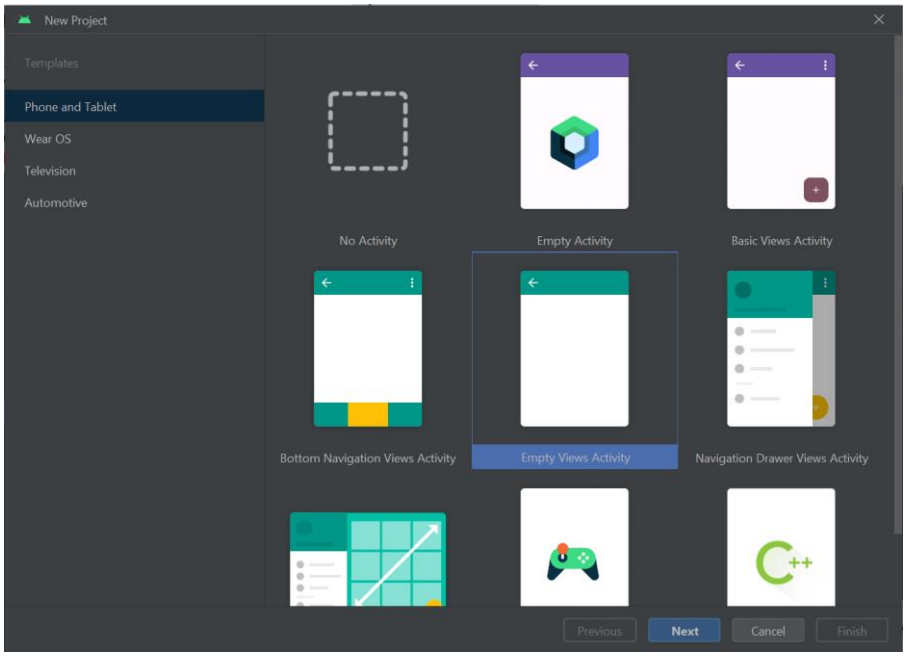
Dupa finalizarea instalarii Andoid Studio, fereastra principala arata in felul urmator:



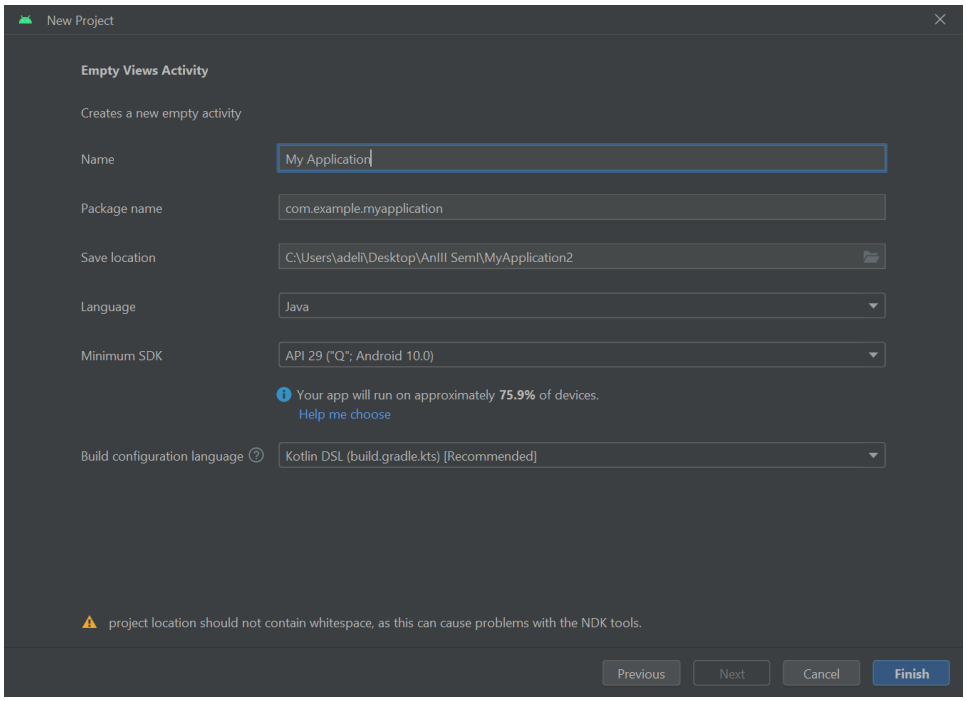
Crearea Aplicatiilor

1. Prima aplicatie

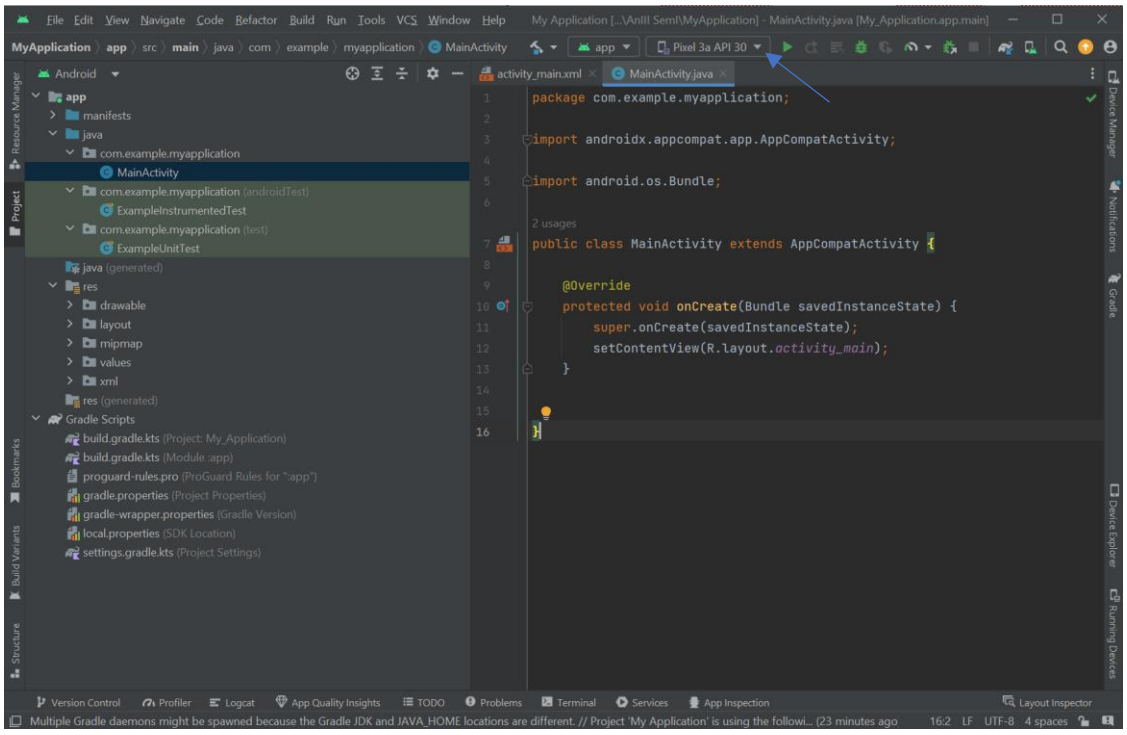
Aceasta prima aplicatie este una introductiva in acest mediu de dezvoltarea al aplicatiilor de tip Andoid, astfel o sa fie ceva simplu, cum ar fi afisarea unui mesaj. Pentru aceasta, dupa deschiderea plicatiei, in fereastra principala, apasam butonul **New Project** -> **Phone and Tablet** (stanga) -> **Empty Views Activity**.



Pasul urmator este sa apasam butonul **Next**, ajungand astfel pe pagina urmatoare, unde putem alege un nume sugestiv pentru aplicatia pe care dorim sa o facem, locatia unde se salveaza fisierul, limbajul de programare si versiunea de Android pe care dorim sa lucram. Sa nu uitam, limbajul in care dorim sa lucram este Java.

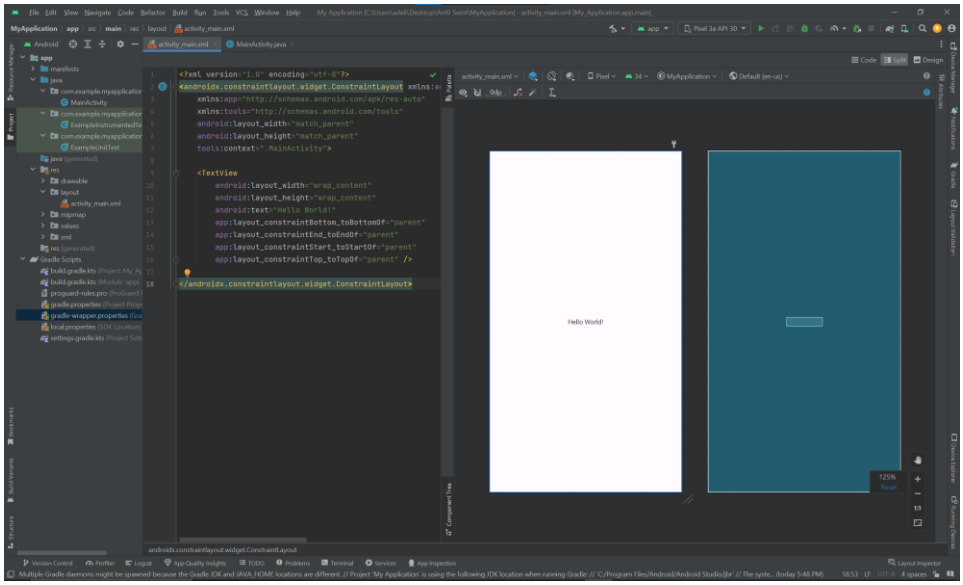


Pentru a finaliza pasii introductivi, este nevoie de apasarea butonului **Finish**, acest lucru facandu-se de 2 ori. Acum, aplicatia ar trebui sa se deschida, dar totusi trebuie sa asteptam putin pana cand se creaza si se deschid toate fisierele, fapt ce nu ar trebui sa ingrijoreze pe nimeni, aplicatia va functiona. In final, fereastra va arata asa:

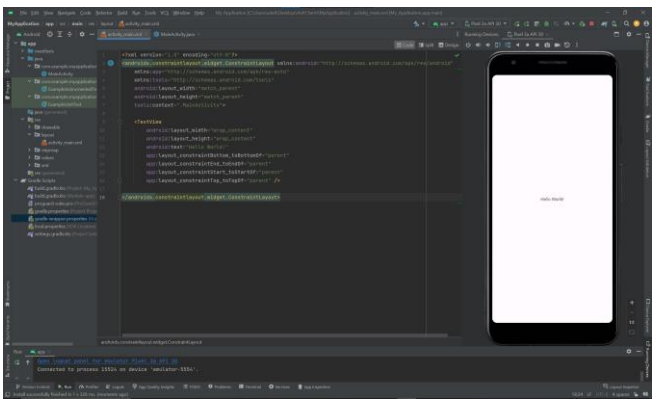


Odata facuti toti pasii prezentati mai sus, aplicatia Andoid care afiseaza un mesaj este pe cale sa fie finalizata, deoarece aceasta aplicatie a fost implementata automat, deci nu va trebui sa aducem modificari asupra codului pentru a vedea mesajul. Cu toate acestea, in imaginea de sus se poate observa o sageata care ne va ajuta sa simulam aceasta aplicatie. Este vorba despre un emulator, telefon care se comporta exact ca unul din viata reala. Se apasa pe **sageata** indicata -> **Device Manager** -> **Create Device** si se selecteaza versiunea de android pe care o preferam, in acest caz a fost selectat Pixel 3a API 30, Android 11.0 -> **Next** -> **Next** -> **Finish**.

Fisierul creat in Android Studio cu numele **activity_main.xml** permite vizualizarea atat a codului folosit pentru design(prin apasarea butonului **Code**), cat si design-ul in sine(prin apasarea butonului **Design**), cum o sa arate aplicatia noastra. Totusi, putem vedea si in timp real modificarile care se aduc asupra cosului in design(prin apasarea butonului **Split**).



Dupa ce v-ati asigurat ca totul este precum prezentat, puteti apasa butonul **Run** (butonul de langa cel folosit pentru a alege versiunea de android pe care am selectat-o), pentru a vedea cu adevarat ce se intampla in aplicatie.



Pentru a va obisnui cu ceea ce inseamna o aplicatie Andoid, puteti aduce modificari codului din imaginea de mai sus, schimband mesajul care va fi afisat de aplicatie sau pozitia acestuia.

2. Calculator simplu

Cea de-a doua aplicatie pe care dorim sa o facem este cea de a realiza un calculator care sa faca adunarea, scaderea, inmultirea si impartirea a doua numere. Pentru inceput, vom urmarii pasii prezentati la prima aplicatie pentru crearea fisierelor de care avem nevoie.

Mai apoi, vom deschide fisierul de layout(**activity_main.xml**)si stergem continutul acestuia. In acest fisier se va defini cum vor fi aranjate si afisate elementele interfetei utilizatorului in aplicatie. Codul va fi structurat in metode clare pentru fiecare operatie, facilitand intelegerea acestuia.

ConstraintLayout este un tip de layout care permite plasarea si alinierea usoara a elementelor intr-un mod flexibil, fiind utilizat pentru a defini pozitia si relatiile intre diferite elemente ale interfetei.

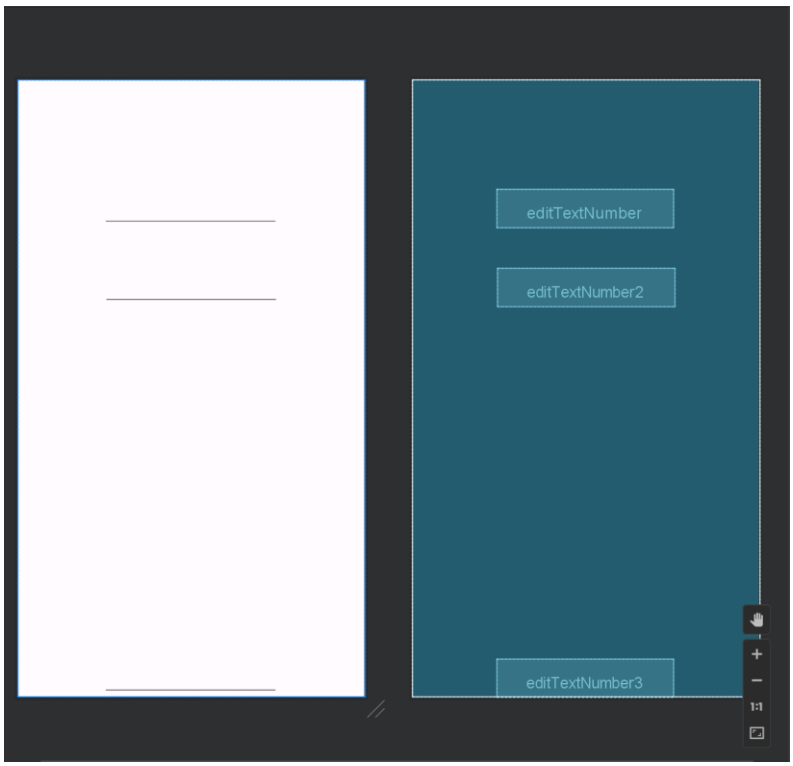
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
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

```

In cele ce urmeaza, vom defini 2 casete de text pentru cele doua numere pe care le vom introduce, urmate de inca o caseta pentru afisarea rezultatului: **editTextNumber, editTextNumber2, editTextNumber3.** Aceste casete de text vor avea diferite atribute, precum width, height, ems – numarul de caractere afisate, inputType – numerele. Atributele utilzate pentru a specifica cum sunt plasate si aliniate elementele in cadrul layout-ului vor fi imbinate cu cele care controleaza pozitionarea relativa a elementelor in raport cu parintele lor. Codul prezentat mai jos va fi pentru prima caseta de text, respectiv pentru primul numar pe care il vom introduce.

```
    <EditText
        android:id="@+id/editTextNumber"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="number"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.19" />
```

Pe baza acestui cod, va trebui facuta implementarea si pentru celelalte doua casete, respectiv pentru al doilea numar si pentru rezultatul care va fi furnizat. Aici, fiecare se poate juca cu imaginatia si sa faca implementari diferite, schimband anumite atribute, dupa preferinta. Dupa alegerea plasarii in interfata a casetelor pentru text, aceasta ar trebui sa arate asa:



Urmatorul pas este implementarea butoanelor pentru fiecare operatie matematica aleasa. Si acestea vor avea atribute specifice, precum width, height, marginTop, onClick – folosit pentru metoda apelata atunci cand butonul este apasat si text – textul afisat pe buton, pentru a sti ce operatie va fi efectuata.

```
33      <Button
34          android:id="@+id/button2"
35          android:layout_width="wrap_content"
36          android:layout_height="wrap_content"
37          android:layout_marginTop="56dp"
38          android:onClick="Add"
39          android:text="ADD(+)"
40          app:layout_constraintEnd_toEndOf="parent"
41          app:layout_constraintStart_toStartOf="parent"
42          app:layout_constraintTop_toBottomOf="@+id/editTextNumber2" />
43
```

Codul prezentat mai sus este butonul pentru care se va face operatia de adunare. In continuare, va trebui sa implementati restul butoanelor pentru celelalte operatii, iar la final, interfata va arata cam asa:

In activitatea principala (**MainActivity.java**) vom crea o metoda **onCreate**, prin care se stabileste continutul activitatii, adica se leaga de activitatea de layout definit in fisierul **activity_main.xml**.

```java
package com.example.calculator;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;


2 usages
public class MainActivity extends AppCompatActivity {

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

Tot in acest fisier vor definite si implementate metode pentru fiecare operatie matematica specifica: **Add, Subtract, Multiply, Divide.** Aceste metode vor fi aplelate atunci cand utilizatorul apasa pe butoanele corespunzatoare, vom vedea mai tarziu cum. Mai jos este prezentata metoda pentru adunare a doua numere, iar pentru celelalte metode se va face implementarea de catre voi, asemanator cu aceasta.

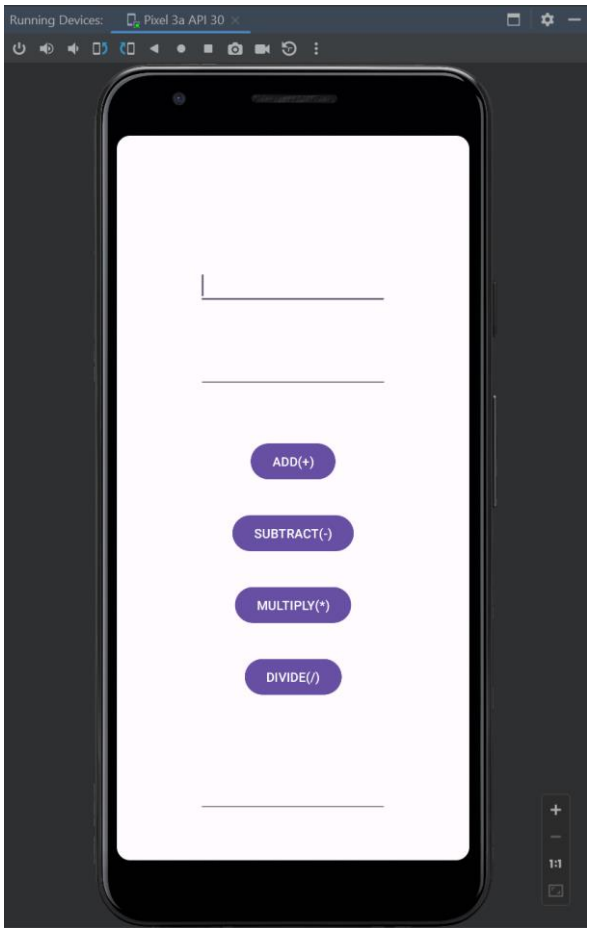
```java
    public void Add(View v){
        EditText et1 = findViewById(R.id.editTextNumber);
        EditText et2 = findViewById(R.id.editTextNumber2);
        EditText et3 = findViewById(R.id.editTextNumber3);

        double n1 = Integer.parseInt(et1.getText().toString());
        double n2 = Integer.parseInt(et2.getText().toString());
        double result = n1 + n2;

        et3.setText("Total Value " + result);
    }
```

Fiecare dintre aceste metode preia valorile introduse de utilizator din cele doua casete de text, realizeaza operatia matematica si afiseaza rezultatul intr-o a treia caseta de text.

Dupa ce s-au facut toate aceste implemetari, aplicatia finala va arata cam asa, in functie de ce parametrii ati furnizat pentru partea de layout.



3. Tic Tac Toe

Cea de-a treia aplicatie pe care dorim sa o facem este cea de a realiza un joc care pe care majoritatea persoanelor il stiu, „X si 0”. Pentru inceput, vom urmarii pasii prezentati la prima aplicatie pentru crearea fisierelor de care avem nevoie.

Mai apoi, vom deschide fisierul de layout(**activity_main.xml**)si stergem continutul acestuia. In acest fisier se va defini cum vor fi aranjate si afisate elementele interfetei utilizatorului in aplicatie. Codul va fi structurat in metode clare pentru fiecare operatie, facilitand intelegerea acestuia.

RelativeLayout este un tip de layout care permite plasarea si alinierea usoara a elementelor intr-un mod flexibil, fiind utilizat pentru a defini pozitia si relatiile intre diferite elemente ale interfetei.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:layout_margin="10dp"
    android:background="@drawable/blue"
    >
```

Urmatorul pas este afisarea numelui aplicatiei, iar pentru acesta sunt setate dimensiunea, culoarea si locul in care sa fie afisat.

```
<TextView
    android:id="@+id/texttitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Tic Tac Toe"
    android:layout_centerHorizontal="true"
    android:textSize="36sp"
    android:textStyle="bold"
    android:textColor="#cc1234"
    android:fontFamily="cursive"
    />
```

Se defineste un nou **RelativeLayout** in care se organizeaza elementele pentru afisarea scorurilor jucatorilor, iar textele pentru scoruri si punctele sunt stilizate cu fontul „akaya_telivigala”.

```
<RelativeLayout
    android:id="@+id/relative_layout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_below="@+id/texttitle"
    android:layout_marginBottom="20dp">

    <TextView
        android:id="@+id/text_player1"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:fontFamily="@font/akaya_telivigala"
        android:text="Player 1 Score"
        android:textAlignment="center"
        android:textColor="@color/black"
        android:textSize="26dp"
        android:textStyle="bold" />
```

Asemanator cu codul furnizat mai sus, se vor defini si alte TextView-uri, pentru player2, scorul pentru player1 si scorul pentru player2.

LinearLayout va aseza butoanele intru-un layout vertical, organizandu-le sub zona de scoruri.

```
82      <LinearLayout
83          android:id="@+id/layout_linear"
84          android:layout_width="wrap_content"
85          android:layout_height="wrap_content"
86          android:orientation="vertical"
87          android:layout_centerHorizontal="true"
88          android:layout_below="@+id/relative_layout"
89          >
```

In acesta se vor defini 3 LinearLayout care vor contine cele 3 butoane pentru a selecta casuta unde veti dori sa puneti X sau 0, in functie de randul cui este sa selecteze celula din joc. Butoanele sunt organizate ca o matrice de 3x3 in layout-urile orizontale definite. Mai jos aveti structura pe care trebuie sa o urmariti pentru a realiza lucrurile prezentate.

```
91          <LinearLayout
92              android:layout_width="wrap_content"
93              android:layout_height="wrap_content"
94              android:orientation="horizontal"
95              >
96              <Button
97                  android:id="@+id/btn1"
98                  android:layout_width="75dp"
99                  android:layout_height="75dp"
100                 android:textSize="26dp"
101                 android:layout_marginHorizontal="5dp"
102                 android:layout_marginBottom="1.25dp"/>
```

Ultimul pas este sa definiti un **TextView** care va afisa mesajele de stare sub matricea de butoane, avand aceleasi atribute ca cel prezentat mai sus, dar cu o alta functionalitate. Mai mult, se vor defini si 2 butoane care vor permite utilizatorului sa inceapa un nou joc sau sa reseteze jocul curent, stilizand textul cu dimensiune si font.

In fisierul **MainActivity.java** se va declara clasa cu acelasi nume si va implementa logica pentru jocul „Tic Tac Toe”. Aceasta gestioneaza interfata de utilizator, verifica

castigatorul, actualizeaza scorurile si ofera functionalitati pentru resetarea si reluarea jocului.

Se vor declara urmatoarele atribute:

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener{

    6 usages
    boolean playerOneActive;
    2 usages
    private TextView player1, player2, player;
    13 usages
    private Button[] buttons = new Button[9];
    2 usages
    private Button reset, playagain;
    8 usages
    int[] gameState = {2,2,2,2,2,2,2,2,2};
    1 usage
    int[][] winningPositions = {{0,1,2}, {3,4,5}, {6,7,8}, {0,3,6}, {1,4,7},
            {2,5,8}, {0,4,8}, {2,4,6}};
    4 usages
    int rounds;
    4 usages
    private int player1Cnt, player2Cnt;
```

Metoda **onCreate(Bundle savedInstanceState)** va initializa elementele interfetei de utilizator, variabilele si evenimentele cu ajutorul denumirilor din **activity_main.xml**. Asociati butoanele cu ajutorul setOnClickListener utilizand **this**.

Urmeaza implementarea metodei **onClick(View view)** cu ajutorul careia se va actualiza starea celulelor si se va verifica castigatorul(daca acesta exista). Totodata trebuie actualizat scorul si afisat un mesaj sugestiv(**de stare**).

```
if(!((Button)view).getText().toString().equals(""))
    return;
else if(checkWinner())
    return;
String bID = view.getResources().getResourceEntryName(view.getId());
int gameStatePointer = Integer.parseInt(bID.substring(bID.length()-1, bID.length()));

if(playerOneActive)
{
    ((Button)view).setText("X");
    ((Button)view).setTextColor(Color.parseColor("#ffc34a"));
    gameState[gameStatePointer] = 0;
}
else
{
    ((Button)view).setText("O");
    ((Button)view).setTextColor(Color.parseColor("#70fc3a"));
    gameState[gameStatePointer] = 1;
}
rounds++;
```

Metoda **checkWinner()** verifica daca exista un castigator comparand starea curenta a celulelor cu posibilele combinatii castigatoare, returnand adevarat(**true**) daca se gaseste vreo potrivire.

```
private boolean checkWinner() {
    boolean winnerRes = false;
    for(int[] winningPositions: winningPositions)
    {
        if(gameState[winningPositions[0]] == gameState[winningPositions[1]] &&
                gameState[winningPositions[1]] == gameState[winningPositions[2]] &&
                gameState[winningPositions[0]] == 1)
        {
            winnerRes = true;
        }
    }
    return winnerRes;
}
```

Metoda **playAgain()** pur si simplu reseteaza starea jocului, inclusiv celulele, numarul de runde si castigatorul activ, afisand un mesaj de stare sugestiv.

Ultima metoda care urmeaza a fi implementata este metoda **updatePlayerScore()**, metoda cu un nume destul de sugestiv, care va actualiza scorurile jucatorilor in elementele `**TextView**`.

```
private void updatePlayerScore() {
    player1.setText(Integer.toString(player1Cnt));
    player2.setText(Integer.toString(player2Cnt));
}
```

Aceasta aplicatie ofera o introducere detaliata in implementarea jocului **TicTacToe** pe platforma **Android**, evidentiind aspecte esentiale ale programarii in limbajul **Java**.

Aceste aplicatii va ajuta sa va familiarizati cu mediul de lucru in **Android**, evidențiind aspecte cheie ale programării în limbajul **Java**. Astfel, studenții au acum posibilitatea de a aplica și extinde aceste cunoștințe în dezvoltarea altor aplicații interactive adaptate dispozitivelor mobile.