# Mobilna aplikacija za generiranje slučajnih brojeva

**Rinčić, Leon**



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Sveučilišni studij**

# MOBILNA APLIKACIJA ZA GENERIRANJE SLUČAJNIH BROJEVA

**Završni rad**

**Leon Rinčić**

**Osijek, 2022.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

**Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju**

**Osijek, 16.09.2022.**

**Odboru za završne i diplomske ispite**

# Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

| | |
|---|---|
| **Ime i prezime Pristupnika:** | Leon Rinčić |
| **Studij, smjer:** | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. Pristupnika, godina upisa:** | R4269, 26.07.2018. |
| **OIB Pristupnika:** | 02160065608 |
| **Mentor:** | Izv. prof. dr. sc. Mirko Köhler |
| **Sumentor:** | , |
| **Sumentor iz tvrtke:** | |
| **Naslov završnog rada:** | Mobilna aplikacija za generiranje slučajnih brojeva |
| **Znanstvena grana rada:** | **Programsko inženjerstvo (zn. polje računarstvo)** |
| **Zadatak završnog rad:** | Tema je zauzeta. Zadatak završnog rada je napraviti mobilnu aplikaciju za generiranje slučajnih brojeva. Potrebno je istražiti područje slučajnih brojeva i njihovo nastajanje. Pomoću senzorskih podataka na mobilnom uređaju, aplikacija generira zadanu dužinu slučajnog broja. |
| **Prijedlog ocjene završnog rada:** | Vrlo dobar (4) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 2 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda<br>Jasnoća pismenog izražavanja: 2 bod/boda<br>Razina samostalnosti: 2 razina |
| **Datum prijedloga ocjene od strane mentora:** | 16.09.2022. |
| **Datum potvrde ocjene od strane Odbora:** | 21.09.2022. |

| | |
|---|---|
| Potvrda mentora o predaji konačne verzije rada: | Mentor elektronički potpisao predaju konačne verzije. |
| | Datum: |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 22.09.2022.**

| | |
|---|---|
| **Ime i prezime studenta:** | Leon Rinčić |
| **Studij**: | Preddiplomski sveučilišni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | R4269, 26.07.2018. |
| **Turnitin podudaranje [%]:** | 8 |

Ovom izjavom izjavljujem da je rad pod nazivom **: Mobilna aplikacija za generiranje slučajnih brojeva**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Mirko Köhler

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koj i mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

# SADRŽAJ

**1. UVOD ..... 1**

**1.1. Zadatak završnog rada ..... 1**

**2. PREGLED PODRUČJA ..... 2**

**2.1. „Istinski“ generatori slučajnih brojeva ..... 2**

2.1.1. Sklopovski temeljeni izvori ..... 2

2.1.2. Izvori temeljeni na programu ..... 2

**2.2. Pseudo slučajni generatori brojeva ..... 3**

**2.3. Usporedba s već postojećim rješenjima ..... 3**

2.3.1. Random Number Generator Plus ..... 3

2.3.2. Random UX ..... 4

2.3.3. Pretty Random – Random Number ..... 5

**3. KORIŠTENI ALATI I TEHNOLOGIJE ..... 6**

**3.1. Android Studio ..... 6**

**3.2. Android OS ..... 7**

**3.3. Java ..... 7**

**3.4. Žiroskop ..... 7**

**3.5. Senzor tlaka zraka ..... 8**

**3.6. Akcelerometar ..... 8**

**3.7. Geo lokacija ..... 9**

**3.8. SHA-512 ..... 9**

**4. PROGRAMSKO RJEŠENJE I PRIMJENA ..... 10**

**4.1. Stvaranje aplikacije ..... 10**

**4.2. Rad aplikacije ..... 17**

**5. REZULTATI ..... 19**

**5.1. Rezultati generiranja slučajnih brojeva ..... 19**

**6. ZAKLJUČAK ..... 22**

**LITERATURA ..... 23**

**SAŽETAK ..... 25**

**ABSTRACT** ..... **26**

**ŽIVOTOPIS** ..... **27**

# 1. UVOD

Ideja slučajnih brojeva nije nova jer se nasumični brojevi koriste tisućama godina. Cilj nasumičnih brojeva, od lutrija u antičkom Babilonu, preko ruleta u Monte Carlu pa sve do igara na kockicama u Las Vegasu, jest ostaviti krajnji ishod slučajnosti. Izostavivši kockanje, slučajnost se također koristi u znanosti, statistici, kriptografiji. Za generiranje mnogo nasumičnih brojeva koristeći metode mehaničke prirode potrebno je mnogo vremena i truda. Iz tog razloga smo dizajnirali mnogo moćnije alate.

Postoje različiti načini za stvaranje nasumičnih brojeva na računalu, ali ne postižu svi željenu istinsku slučajnost. U većini slučajeva, to ih čini beskorisnim za kriptografiju. Postoje i pravilno konstruirani slučajni generatori brojeva koji daju istinske slučajne brojeve i mogu se koristiti u kriptografiji.

Područje računarstva dobro je upoznato s funkcijom slučajnih brojeva. Zaštita osjetljivih podataka jedan je od najpopularnijih slučajeva korištenja nasumičnosti. Za potrebe kriptografskih primjena koje osiguravaju visoku razinu sigurnosti, potrebni su kvalitetni generatori istinskih nasumičnih brojeva. Slučajnost se može koristiti i za minimiziranje dugih pretraga, ometanje simetrija, skrivanje informacija i mjerenje informacija.

Tri dijela čine glavni dio ovog završnog rada, a to su: mogućnosti stvaranja slučajnog broja, korišteni alati i tehnologije te programsko rješenje. Prvi dio sastoji se od dijela u kojem se opisuje kako ljudi sada mogu doći do nasumičnog broja, u drugom dijelu opisane su tehnologije i alati koji su korišteni za stvaranje aplikacije, a u trećem dijelu objašnjeno je programsko rješenje.

## 1.1. Zadatak završnog rada

Cilj završnog rada je istražiti područje slučajnih brojeva te istražiti senzore koji se nalaze u mobitelu. Na temelju toga izraditi aplikaciju koja će generirati istinski slučajni broj. Aplikacija će koristiti senzore mobitela i ljudski utjecaj kako bi se generirao jedinstveni slučajni broj. Trenutno vrijeme će također imati utjecaj.

# 2. PREGLED PODRUČJA

Generatori slučajnih brojeva dijele se na dvije velike skupine. Te dvije skupine su „istinski“ generatori slučajnih brojeva i pseudo slučajni generatori brojeva. Istinski generatori slučajnih brojeva bazirani su na prirodnim pojavama ili fizikalnim procesima. Njihova nepredvidljivost potvrđena je zakonima fizike. Unutar računala, oni se dobivaju primjenom sklopovskih ili programskih rješenja. Pseudo slučajni nizovi brojeva, s druge strane, generirani su pomoću predodređene rekurzivne formule na temelju unosa entropije u početno stanje poznato kao seed ili key.

## 2.1. „Istinski“ generatori slučajnih brojeva

Ova grupa generatora slučajnih brojeva uključuje nedeterminističke izvore slučajnih brojeva, poput onih koje inspiriraju prirodna događanja (nasumični broj neovisan je o svome prethodniku u nizu). Izvedba ovih generatora je delikatna, skupa, a njihova uporaba je ograničena. Oni su u cijelosti ili uglavnom temeljeni na sklopovlju. Osim toga, stvaranje brojeva moglo bi biti sporo, a samo sklopovlje bi se moglo pokvariti. Stoga, simulacijski softveri češće koristi te generatore. Za određivanje „istinitih” nasumičnih brojeva mogu se koristiti dvije skupine izvora:

### 2.1.1. Sklopovski temeljeni izvori

- intervali između emisija tijekom radioaktivnog raspada
- šum poluvodičke diode
- nestabilnost frekvencije oscilatora
- očitanje mikrofona ili očitanja s kamere uređaja

U ovim je izvorima često neophodno dalje obrađivati prikupljene podatke. Dobiveni niz često je izvan ravnoteže jer je pojava nekih brojeva vjerojatnija, a moguća je i povezanost brojeva u nizu što znači da bi se pojavom određenog broja niza povećala i vjerojatnost da idući broj bude neki točno određeni. To bi značilo da je generator zapravo periferni uređaj jer se ti izvori moraju vanjskim putem povezati s računalom.

### 2.1.2. Izvori temeljeni na programu

- sistemski sat
- interval između dva pritiska tipki na tipkovnici ili mišu
- sadržaj ulaznog/izlaznog spremnika
- specifične varijable operacijskog sustava

Iako se u početku čini da ovaj skup izvora može pružiti nepredvidivi niz brojeva, to nije uvijek slučaj. Jednostavno je „predvidjeti“ ponašanje sistemskog sata. Uz to, korisnička interakcija prati određenu pravilnost koja se može preslikati na niz generiranih brojeva.

Moguće je stvoriti dobar generator slučajnih brojeva pametnom kombinacijom nekoliko različitih izvora slučajnih brojeva.

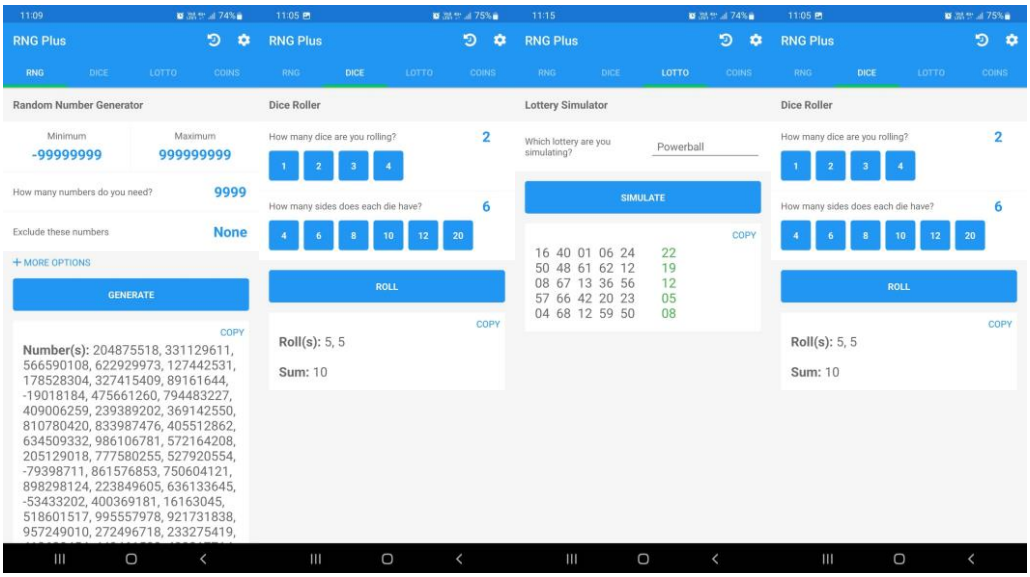
## 2.2. Pseudo slučajni generatori brojeva

Ovakav tip generatora je zapravo nekakva funkcija koja na temelju ulaznih vrijednosti daje niz nasumičnih pseudo slučajnih brojeva. Početni uvjet (seed) mora biti istinski slučajan. On se dobiva generatorom istinski slučajnih brojeva. Nakon što su početni uvjeti određeni, svi brojevi niza mogu se predvidjeti što nas dovodi determinističke karakteristike, pa dobivene brojeve nazivamo pseudo slučajnima. Ukoliko je početni uvjet poznat, cijeli niz brojeva može se reproducirati što znači da ako treba ponovna reprodukcija niza, dovoljno je sačuvati početni uvjet. Deterministične metode mogu se skriti pažljivim odabirom generatora i početnih uvjeta, a rezultat toga je da generirani niz poprima mnoge karakteristike slučajnih brojeva. Generatori pseudo slučajnih brojeva su brži, lakše ostvarivi i prenosivi. Potrebno je, međutim, pri odabiru generatora, dobro paziti bilo da se upotrebljava u simulacijske svrhe ili u kriptografske.

## 2.3. Usporedba s već postojećim rješenjima

### 2.3.1. Random Number Generator Plus

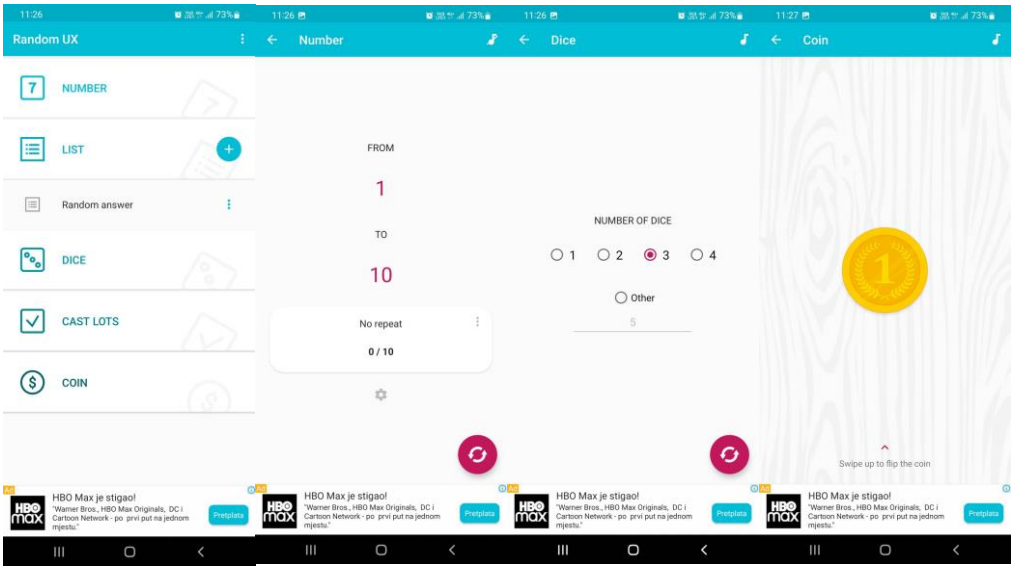
Random Number Generator Plus je Android aplikacija dostupna za mobilne uređaje. Proizvela ju je tvrtka pod nazivom RandomAppsInc. Ova aplikacija generira maksimalno 9999 brojeva u rasponu od [-99999999, 99999999]. Aplikacija također ima ugrađen generator slučajnih brojeva napravljen na principu kockica, binga i bacanja kovanice. Sličnost s aplikacijom koja se izrađuje ovim završnim radom je ta da ova aplikacija generira slučajne brojeve no ti brojevi su pseudo slučajni.




Sl. 2.1. Prikazi aplikacije Random Number Generator Plus

### 2.3.2. Random UX

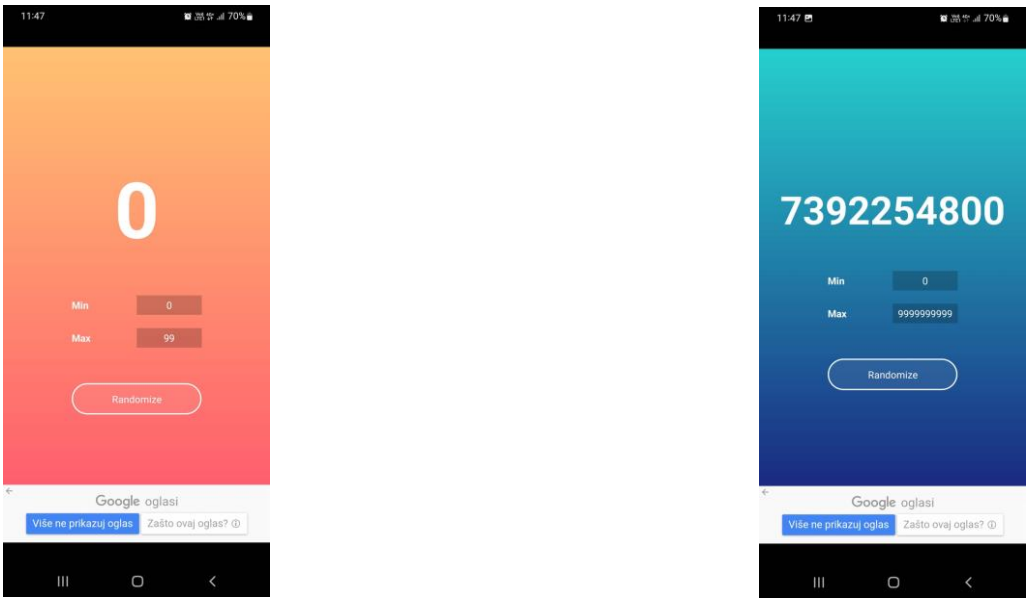
Random UX je mobilna aplikacija za Android uređaje. Proizvedena je od strane tvrtke UXApps Ltd. Mogućnosti aplikacije su generiranje nasumičnih brojeva, nasumični odabir listi, „bacanje“ kockice i novčića. Sličnosti su također generiranje slučajnih brojeva, ali i ti brojevi su pseudo slučajni.




Sl. 2.2. Prikazi aplikacije Random UX

### 2.3.3. Pretty Random – Random Number

Pretty Random – Random Number jednostavna je mobilna aplikacija za Android OS. Aplikacija sadrži jedan prikaz i u njoj je moguće generirati pseudo slučajni broj. Ova aplikacija i aplikacija koje se izrađuje vrlo su slične, a razlika je da se u ovoj aplikaciji pseudo slučajni broj, a u aplikacijom koja se opisuje u ovom radu stvara se istinski slučajni broj.




Sl. 2.3. Prikaz aplikacije Pretty Random – Random Number

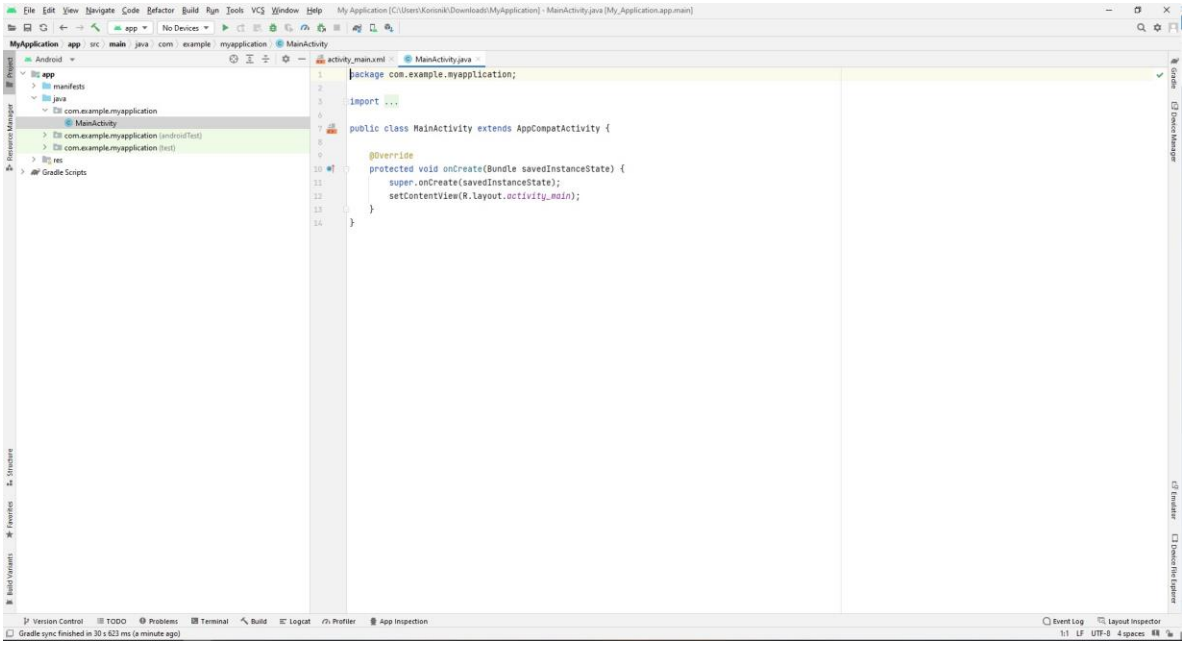
# 3. KORIŠTENI ALATI I TEHNOLOGIJE

Ovaj dio završnog rada opisuje alate i tehnologije korištene za stvaranje aplikacije.

## 3.1. Android Studio

Službena integrirana razvojna okolina (IDE) za razvoj Android aplikacija naziva se Android studio i temeljena je na IntelliJ IDEA, integrirano razvojno okruženje za Java softver i uključuje alate za uređivanje i razvoj kodova. Za podršku razvoju aplikacija unutar operativnog sustava Android, Android Studio koristi Gradle-ov sustav za izgradnju, emulator, kodne predloške i integraciju GitHuba. Svaki projekt u studiju Android ima jedan ili više modaliteta s izvornim kodom i datotekama resursa. Ti modaliteti uključuju module aplikacija Android, module biblioteke i module Google App Enginea.

Program je prvotno predstavljen na Google I/O u svibnju 2013. godine, a prva stabilna verzija stavljena je na raspolaganje u prosincu 2014. godine. Dostupan je na Mac, Windows i Linux operacijskim sustavima. Kao glavni IDE za razvoj Android aplikacija, preuzeo je ulogu Eclipse Android Development Toolsa (ADT). Android Studio i Software Development Kit moguće je preuzeti direktno sa Googleove stranice.




Sl. 3.1. Android studio

## 3.2. Android OS

Android OS je operacijski sustav temeljen na Linuxu i većinom se koristi na pametnim telefonima i tabletima. Operacijski sustav izgrađen na Linux kernelu, GUI, internet pretraživač i aplikacije krajnjih korisnika koje se mogu preuzeti sadržane su u Android platformi. Iako je krenuo kao operacijski sustav za pametne telefone i tablete, Google je odlučio proširiti ga na automobile, televizore i pametne satove. Android je trenutno najrasprostranjeniji operacijski sustav za pametne telefone i broji gotovo 2 milijarde aktivnih korisnika. Izvorni kod otvorenog je tipa.


Sl. 3.2. Android logo

## 3.3. Java

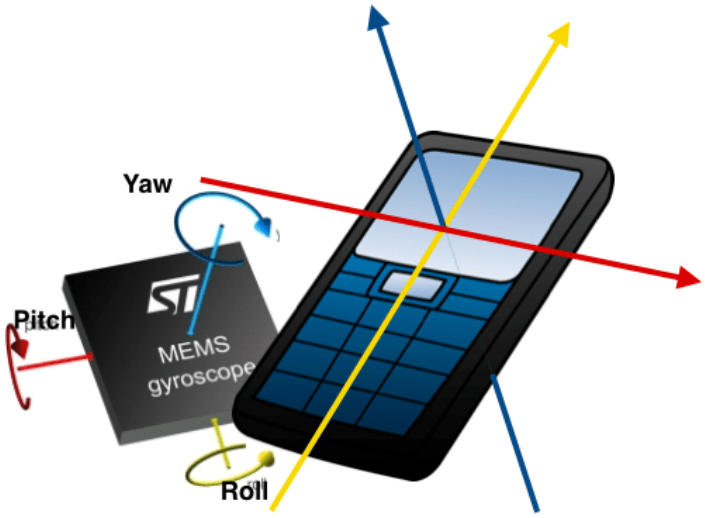
Java je jedan od najpopularnijih objektno orijentiranih programskih jezika. Vrlo je slična programskim jezicima C i C++, no za razliku od programskog jezika C, u Javi se koristi automatski skupljač smeća koji uvelike olakšava rukovanje memorijom. Dizajnirao ga je James Gosling i tvrtka Sun Microsystems. Prvi put je objavljen 1995. kao dio Java platforme. Java je bila glavni Googleov razvojni jezik Android aplikacija do 2019. godine kada ju zamjenjuje Kotlin jer je Google tim smatra kako će to učiniti razvoj Android aplikacija lakšim i zanimljivijim.


Sl. 3.3. Java logo

## 3.4. Žiroskop

Žiroskop je uređaj koji se upotrebljava za određivanje ili održavanje orijentacije i kutne brzine. Slično tomu, u mobitelu se nalazi žiro senzor koji bilježi kutnu brzinu vrtnje i promjene u orijentaciji mobitela. Koriste se pri radu s različitim aplikacijama i softwareom. Jedan od primjera

je u aplikaciji Google karte, tijekom korištenja navigacije, može se uočiti kako se pokazivač koji predstavlja uređaj okreće dok se osoba koja drži uređaj kreće. Pomoću žiroskopa određuje se orijentacija mobitela i tako se pokazivač okreće. Prisutnost žiro senzora također nekada može predstavljati i problem. Jedan od primjera je zakretanje zaslona koji se zakreće zahvaljujući žiroskopu no taj problem lako se otkloni tako što se ugasi senzor. Na slici 2.4. može se vidjeti prikazane 3 glavne osi senzora u pametnom telefonu.




Sl. 3.4. Osi žiroskopa u pametnom telefonu

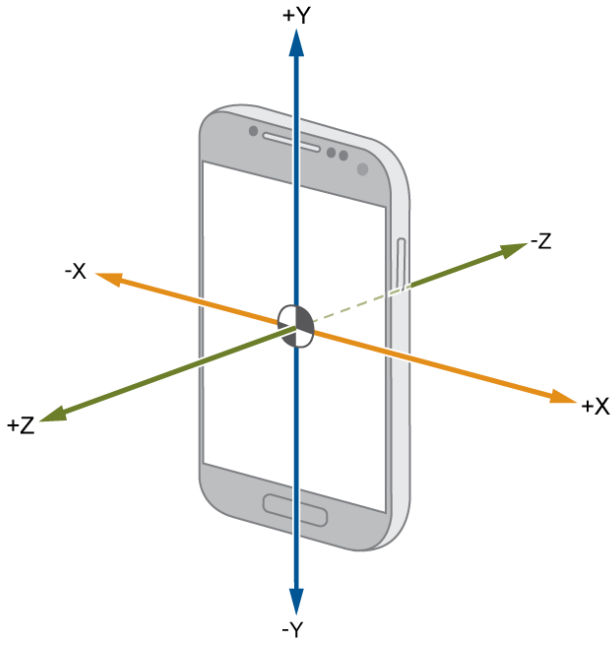
## 3.5. Senzor tlaka zraka

Dodavanjem senzora tlaka zraka, pametni telefoni su podigli svoje lokacijske usluge na višu razinu. Korisnici mogu dobiti puno precizniju lokaciju i iz tog razloga geolokacijske usluge rade preciznije. U Samsungovim i drugim Android uređajima, koristeći senzor pritiska tlaka, može se odrediti trenutnu nadmorsku visinu. Još jedna od upotreba senzora tlaka, ako koristimo vakuumsku komoru, može biti da se odredi postoji li stopostotni vakuum unutar komore.

## 3.6. Akcelerometar

Akcelerometar unutar pametnog telefona je senzor koji daje prostorni položaj objekta na koji je ugrađen. Funkcionira koristeći 3 osi kao što je prikazano na slici 2.4. Akcelerometar mjeri brzinu ubrzanja koja se primjenjuje na uređaj u odnosu na sve 3 osi. Ubrzanje na svakoj od osi izraženo

je u $m/s^2$. Gravitacijska sila uvijek je uključena u mjerenju ubrzanja. Razlika između žiroskopa i akcelerometra je ta da žiroskop može osjetiti rotaciju dok akcelerometar to ne može.




Sl. 3.5. Prikaz osi akcelerometra

## 3.7. Geo lokacija

Geo lokacija je geografska lokacija uređaja povezanog na internet. Ovaj izvor podataka daje informacije koristeći GPS sustav i internet. Geografska visina i širina vrijednosti su koje su izražene u stupnjevima.

## 3.8. SHA-512

SHA-512 ili Secure Hash Algorithm 512 je algoritam raspršivanja koji se koristi kako bi se podatak bilo koje dužine pretvorio u podatke fiksne dužine. Izlazna vrijednost hash algoritma je uvijek fiksna i neovisna o ulaznoj vrijednosti. I najmanja promjena na ulaznoj vrijednosti algoritma u potpunosti mijenja izlaznu vrijednost. Primjer kako izgleda hash broja 10: 3c11e4f316c956a27655902dc1a19b925b8887d59eff791eea63edc8a05454ec594d5eb0f40ae151d f87acd6e101761ecc5bb0d3b829bf3a85f5432493b22f37. Ovaj hash zapravo je heksadekadski broj. SHA se ovdje koristi kako bi dobili 128 znamenkasti nasumični broj.

# 4. PROGRAMSKO RJEŠENJE I PRIMJENA

U ovom poglavlju prikazuje se stvaranje aplikacije i način na koji ona funkcionira.

## 4.1. Stvaranje aplikacije

Obavezni dio izrade Android mobilne aplikacije je AndroidManifest.xml datoteka. U toj datoteci sadržane su bitne informacije o aplikaciji koje alati koriste za izgradnju Androida. Sve dozvole koje aplikacija mora imati za pristupanje zaštićenim dijelovima sustava, naziv aplikacije, deklaracija komponenti. Za potrebe ove aplikacije, bilo je potrebno dodati dozvole za pristup lokaciji uređaja i dozvolu za pristup internetu. Obje dozvole koriste se u svrhu dobivanja geo lokacije uređaja. Na slici 4.1. prikazane su dozvole.

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

Sl. 4.1. Dozvole za pristup lokaciji i internetu

U podacima o aplikaciji na slici 4.2. nalaze se deklaracija imena i ikona aplikacije.

```
<application
    android:allowBackup="true"
    android:dataExtractionRules="@xml/data_extraction_rules"
    android:fullBackupContent="@xml/backup_rules"
    android:icon="@mipmap/ic_launcher"
    android:label="RandomNumberGenerator"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/Theme.RandomNumberGenerator"
    tools:targetApi="31">
```

Sl. 4.2. Podaci o aplikaciji

Ova aplikacija sadrži samo jednu komponentu odnosno samo jedan zaslon. Kada bi aplikacija sadržavala još neki zaslon oni bi također bili prikazani ovdje.

```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

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

Sl. 4.3. Podaci o komponentama

Aktivnost je glavna komponenta Android aplikacije. Ova aplikacija ima samo jedan zaslon pa ima i samo jednu aktivnost. Svaka aplikacija mora imati glavni zaslon koji se većinom naziva MainActivity. To je početni zaslon aplikacije. Izgled zaslona opisan je opisnim jezikom XML. XML je jezik za označavanje podataka i vrlo je sličan HTML jeziku.

activity_main.xml

```
<?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:visibility="visible"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="255dp"
        android:layout_height="58dp"
        android:layout_marginStart="24dp"
        android:layout_marginTop="168dp"
        android:text="@string/current_user_location"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="User Location" />

    <TextView
        android:id="@+id/textView3"
        android:layout_width="250dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="28dp"
        android:text="@string/gyroscope"
        android:textSize="24sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="Gyroscope" />
```

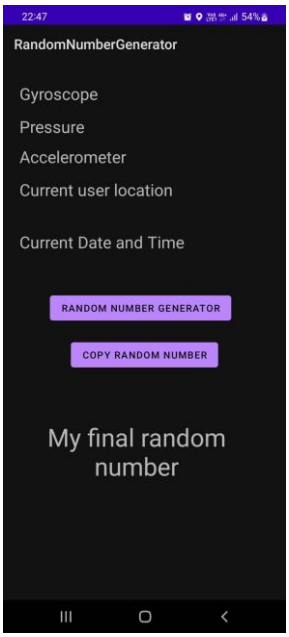
Sl. 4.4. Opisivanje početnog zaslona koristeći XML kod

Svaka aktivnost ima neku interakciju s korisnikom pa zbog toga klasa aktivnost vodi računa o stvaranju prozora u koji se postavlja UI.

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

Sl. 4.5. Veza između zaslona i aktivnosti

Na slici 4.6. prikazan je početni zaslon aplikacije koji sadrži TextViewove i buttone.




Sl. 4.6. Početni zaslon aplikacije

TextView je element korisničkog sučelja koji korisniku prikazuje tekst. Ovdje su potrebni TextView-i kako bi mogli prikazati vrijednosti žiroskopa, pritiska zraka, akcelerometra, trenutne lokacije korisnika, trenutno vrijeme i konačni generirani slučajni broj.

```xml
<TextView
    android:id="@+id/textView"
    android:layout_width="255dp"
    android:layout_height="58dp"
    android:layout_marginStart="24dp"
    android:layout_marginTop="168dp"
    android:text="@string/current_user_location"
    android:textSize="24sp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:text="User Location" />
```

Sl. 4.7. Primjer XML-a za TextView

Na slici 4.7. možemo vidjeti XML kod za svaki od TextViewa zaduženih za prikazivanje podataka na temelju kojih se generira slučajni broj.

TextView za konačni slučajni broj nešto je drugačiji i može se vidjeti na slici 4.8. Razlika u odnosu na gornji XML je ta da je ovdje centrirani i povećani tekst te se dodaje maksimalna dužina zadanog broja koja iznosi 128 znamenki.

```
<TextView
    android:id="@+id/textView7"
    android:layout_width="300dp"
    android:layout_height="match_parent"
    android:scrollbars="vertical"
    android:layout_marginStart="44dp"
    android:layout_marginTop="520dp"
    android:text="@string/my_final_random_number"
    android:textAlignment="center"
    android:textSize="36sp"
    android:maxLength="128"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:text="Final random number" />
```

Sl.4.8. XML za TextView konačnog broja

Button je element korisničkog sučelja koje korisnik može dodirnuti ili kliknuti da bi izveo akciju. Aplikacija ima gumb za generiranje slučajnog broja i gumb za kopiranje istog. Na slici 4.9. nalazi se XML kod oba gumba.

```
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="68dp"
    android:layout_marginTop="56dp"
    android:contentDescription="@string/random_number_generatror"
    android:onClick="RandomNumberButton"
    android:text="@string/random_number_generatror"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView6"
    tools:text="Generate random number" />

<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:text="Copy Random Number"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/button" />
```

Sl.4.9. XML kod gumbova

Slika 4.10. prikazuje kod kojim iniciramo senzore uređaja. Senzori koji se iniciraju su žiroskop, senzor tlaka zraka i akcelerometar. Pomoću upravljača senzora je potrebno stvoriti senzore koje će se koristiti.

```
SensorManager sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

Sensor sensorGyroscope = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
Sensor sensorPressure = sensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE);
Sensor sensorAccelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Sl. 4.10. Stvaranje senzora

Nakon toga potrebno je pozvati SensorEventListener za svaki od ova 3 senzora. SensorEventListener koristi se za primanje obavijesti od SensorManagera kada postoje novi senzorski podaci. Definiranjem listnera automatski se stvore dvije nove metode, OnSensorChanged i OnAccuracyChanged, ali ovdje se koristi samo OnSensorChanged. Kroz tu metodu potrebno je u lokalne varijable spremiti vrijednosti očitanja senzora. Za žiroskop i akcelerometar zbrajamo vrijednosti sve tri osi kako bi dobili jednu vrijednost. Lokalne varijable koristit će se kasnije u kodu.

```
SensorEventListener sensorEventListenerGyroscope = new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        doubleGyroscope = sensorEvent.values[0] + sensorEvent.values[1] + sensorEvent.values[2];
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {
    }
};
SensorEventListener sensorEventListenerPressure = new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        doublePressure = sensorEvent.values[0];
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {
    }
};

SensorEventListener sensorEventListenerAccelerometer = new SensorEventListener() {
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {


        doubleAccelerometer = sensorEvent.values[0] + sensorEvent.values[1] + sensorEvent.values[2];

    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }
};
```

Sl. 4.11. Dohvaćanje očitanja senzora

Sljedeći korak je registrirati svaki listener za određeni senzor na određenoj frekvenciji uzorkovanja koja je u ovom slučaju zadana preko defaultnog delaya i iznosi 200 000 mikrosekundi.

```
sensorManager.registerListener(sensorEventListenerGyroscope, sensorGyroscope, SensorManager.SENSOR_DELAY_NORMAL);
sensorManager.registerListener(sensorEventListenerPressure, sensorPressure, SensorManager.SENSOR_DELAY_NORMAL);
sensorManager.registerListener(sensorEventListenerAccelerometer, sensorAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
```

Sl. 4.12. Registriranje senzora

Prije nego što krene implementiranje geo lokacije, potrebno je dodati Google play services API u projekt kako bi imali pristup Google servisima. To se učini tako da se u build gradle dodaje dependency kao što je prikazano na slici 4.13. Uz to, potrebno je i dodati dopuštenja sa slike 4.1. u AndroidManifest datoteku.

```
implementation 'com.google.android.gms:play-services-location:20.0.0'
```

Sl. 4.13. Dodavanje dependencya

```
private boolean isGPSEnabled() {
    LocationManager locationManager = null;
    boolean isEnabled = false;

    if (locationManager == null) {
        locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    }

    isEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
    return isEnabled;

}
```

Sl. 4.14. Metoda kojom se provjerava je li uključena lokacija uređaja

Na slici 4.14. prikazana je jednostavna metoda koja vraća vrijednost istinu ili laž. Služi za provjeru je li uključena lokacija uređaja.

```
private void turnOnGPS() {


    LocationSettingsRequest.Builder builder = new LocationSettingsRequest.Builder()
            .addLocationRequest(locationRequest);
    builder.setAlwaysShow(true);

    Task<LocationSettingsResponse> result = LocationServices.getSettingsClient(getApplicationContext())
            .checkLocationSettings(builder.build());

    result.addOnCompleteListener(new OnCompleteListener<LocationSettingsResponse>() {
        @Override
        public void onComplete(@NonNull Task<LocationSettingsResponse> task) {

            try {
                LocationSettingsResponse response = task.getResult(ApiException.class);
                Toast.makeText( context: MainActivity.this,  text: "GPS is already tured on", Toast.LENGTH_SHORT).show();

            } catch (ApiException e) {

                switch (e.getStatusCode()) {
                    case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:

                        try {
                            ResolvableApiException resolvableApiException = (ResolvableApiException) e;
                            resolvableApiException.startResolutionForResult( activity: MainActivity.this,  requestCode: 2);
                        } catch (IntentSender.SendIntentException ex) {
                            ex.printStackTrace();
                        }
                        break;

                    case LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:
                        //Device does not have location
                        break;
                }
            }
        }
    });
```

Sl. 4.15. Metoda koja se koristi za paljenje lokacije uređaja

```
private void getCurrentLocation() {


    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (ActivityCompat.checkSelfPermission( context: MainActivity.this, Manifest.permission.ACCESS_FINE_LOCATION) == PackageManager.PERMISSION_GRANTED) {

            if (isGPSEnabled()) {

                LocationServices.getFusedLocationProviderClient( activity: MainActivity.this)
                        .requestLocationUpdates(locationRequest, (LocationCallback) onLocationResult(locationResult) -> {
                                super.onLocationResult(locationResult);

                                LocationServices.getFusedLocationProviderClient( activity: MainActivity.this)
                                        .removeLocationUpdates( callback: this);

                                if (locationResult != null && locationResult.getLocations().size() >0){

                                    int index = locationResult.getLocations().size() - 1;
                                    doubleLatitude = locationResult.getLocations().get(index).getLatitude();

                                    doubleLongitude = locationResult.getLocations().get(index).getLongitude();
                                }
                            }, Looper.getMainLooper());

            } else {
                turnOnGPS();
            }

        } else {
            requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION},  requestCode: 1);
        }
    }
}
```

Sl. 4.16. Metoda za dohvaćanje trenutne lokacije

Slika 4.15. prikazuje metodu za uključivanje lokacije uređaja. Kada se implementira i ta metoda, dostupno je sve što je potrebno za dohvaćanje lokacije. Unutar metode GetCurrentLocation, koja je prikazana na slici 4.16. prvo se provjerava ima li aplikacija dopuštenje da pristupi lokaciji uređaja. Ukoliko ima, nastavlja se dalje na provjeru je li lokacija uključena. Ukoliko je i taj uvjet zadovoljen, dohvaća se lokacija uređaja. Geografska širina i dužina spremaju se u globalne varijable koje će biti korištene za generiranje slučajnog broja.

## 4.2. Rad aplikacije

```
public void RandomNumberButton(View view){

    getCurrentLocation();

    double doubleCurrentDatetime = Calendar.getInstance().getTimeInMillis();

    textViewGyroscope.setText(Double.toString(doubleGyroscope));
    textViewPressure.setText(Double.toString(doublePressure));
    textViewAccelerometer.setText(Double.toString(doubleAccelerometer));
    textViewCurrentDateTime.setText(Double.toString(doubleCurrentDatetime));
    textViewUserLocation.setText("Latitude: "+ doubleLatitude + "\n" + "Longitude: "+ doubleLongitude);

    double doubleFinalRandomNumber = ((1000000*doubleGyroscope) * doublePressure * (10000*doubleAccelerometer) * doubleLatitude * doubleLongitude + doubleCurrentDatetime);

    if(doubleFinalRandomNumber<1)
        doubleFinalRandomNumber*=-1;

    String Sha=String.valueOf(doubleFinalRandomNumber);
    String HexValueOfSha=Sha512.encryptThisString(Sha);
    BigInteger DecValueOfSha = new BigInteger(HexValueOfSha,  radix: 16);

    textViewFinalRandomNumber.setText(String.valueOf(DecValueOfSha));
}
}
```

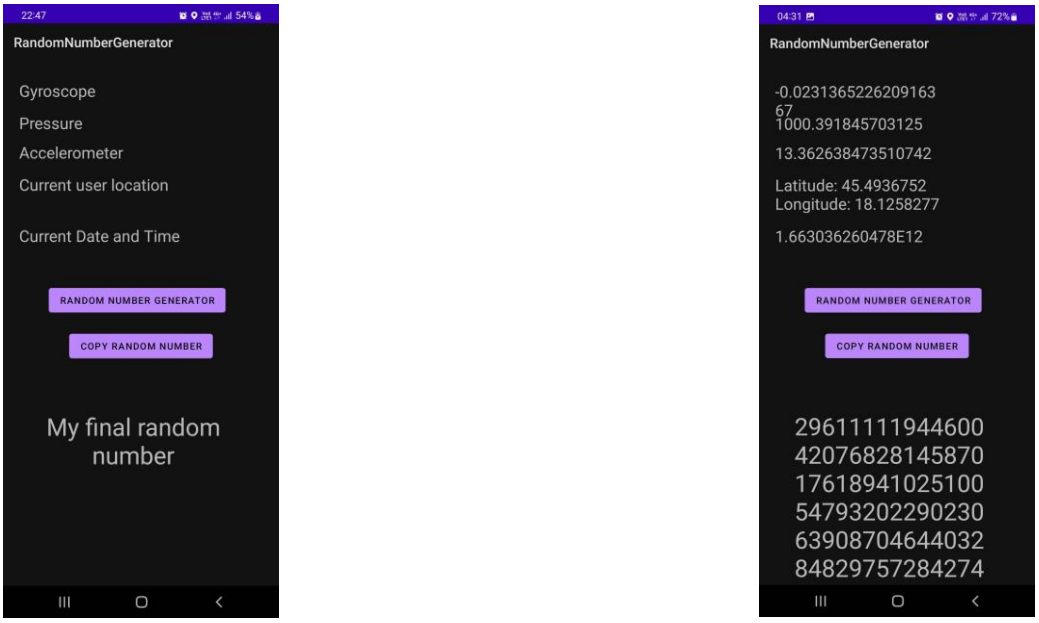
Sl. 4.17. Generiranje slučajnog broja

Slika 4.17. prikazuje OnClickListener koji se koristi za generiranje slučajnog broja. Prilikom klika na gumb GenerateRandomNumber, prvo se pozove metoda za dobivanje trenutačne lokacije. U globalne varijable doubleLatitude i doubleLongitude spremaju se vrijednosti geografske širine i dužine. Bilježi se trenutno vrijeme u milisekundama. Zatim se definiraju vrijednosti TextViewova. Unutar jednadžbe doubleFinalRandomNumber možemo uočiti kako se koriste sve dobivene varijable. Žiroskopska varijabla i varijabla akcelerometra proširene su kako bi imale veći utjecaj na konačni slučajni broj. Dobiveni broj nema 128 znamenki koliko je potrebno i iz tog razloga dobiveni broj stavimo SHA-512 algoritam koji generira heksadekadski broj. Taj heksadekadski broj se zatim ponovno pretvara u dekadski broj i tako se dobiva konačni istinski slučajni broj koji je nenegativan i sastoji se od 128 znamenki.

Slika 4.18. prikazuje implementaciju algoritma SHA-512 koristeći MessageDigest klasu.

```
public class Sha512 {

    public static String encryptThisString(String input)
{
    try {

        MessageDigest md = MessageDigest.getInstance("SHA-512");

        byte[] messageDigest = md.digest(input.getBytes());

        BigInteger no = new BigInteger( signum: 1, messageDigest);

        String hashText = no.toString( radix: 16);

        while (hashText.length() < 32) {
            hashText = "0" + hashText;
        }

        return hashText;
    }

    catch (NoSuchAlgorithmException e) {
        throw new RuntimeException(e);
    }
}
}
```

Sl. 4.18. SHA-512 algoritam




Sl. 4.19. Aplikacija prije i poslije generiranja broja

# 5. REZULTATI

U ovom poglavlju prikazani su rezultati generiranja slučajnih brojeva.

## 5.1. Rezultati generiranja slučajnih brojeva

Tablica 5.1. Rezultati generiranja aplikacije

| | Vrijeme | Rezultat | Krajnji rezultat – slučajni broj |
|---|---|---|---|
| 1. | 15. ruj 2022. 13:56:52 | 7076071499301812 | 99365590593128084097786498871918 74082228408650993809564747434434 06564550839643144176718076316015 76513543238115790051516185489074 |
| 2. | 15. ruj 2022. 14:00:34 | 333681234051944260 | 22746295681543575441604523318710 11152447069376176573937048816094 20767350370587251719928271171054 14908767743481983693795499365845 |
| 3. | 15. ruj 2022. 14:03:15 | 14336613248668980 | 15290940321669506079075748618989 17614285033534797760892854162439 37184873017645533847093241225298 11062715634534184752760730715366 |
| 4. | 15. ruj 2022. 14:05:57 | 8458842960105542 | 11426262005782942608022745894507 55689949441453391613872977027108 85883891182460211976342480207253 76549163265629996865253950614424 |
| 5. | 15. ruj 2022. 14:07:21 | 6589205374596559.9 | 52183274277199901181277494119435 00333492565670687336465019950631 07386073544938958268601826131728 22473224644654888735787240571603 |
| 6. | 15. ruj 2022. 14:09:03 | 3983229425278840.5 | 47896890430286787314432267016790 13693156860797073870792967178063 16195116078821042934180345975115 51882995959313894662931919678531 |
| 7. | 15. ruj 2022. 14:10:16 | 1085041090403963.1 | 95916238502487860750957930303339 53747752979385766479154109915387 |

| | | | |
|---|---|---|---|
| | | | 274192848852085198220151629066602750910535539042290437320508208 2 |
| 8. | 15. ruj 2022. 14:11:49 | 3522058312963468 | 460982937039332535062833891003123761590162334859454004542047419375644605208312881192136397069569438466187247248475922692611037 11 |
| 9. | 15. ruj 2022. 14:12:55 | 765011469711619.5 | 181163796672057005454362399958212638054628056570375522292894366097435120579278689239090835172686633349234101919635749119832891302 |
| 10. | 15. ruj 2022. 14:13:55 | 58138407438257008 | 359242471563247975419862824415550329772586876069792404494095678755813310459526278276109793915393257241362436322383023922653056 0 |
| 11. | 15. ruj 2022. 14:15:44 | 53826309493954664 | 122665405435344206528311704482848066755269191089124348846392032530048371562951988574235092507267521511923375033078243775282399 30 |
| 12. | 15. ruj 2022. 14:16:54 | 1409029963388928.8 | 116084469337327708850603655390804756057605924932022570664719933036214659652366373165474265860023775843357479534145559581269336 74 |
| 13. | 15. ruj 2022. 14:17:48 | 17596130206408264 | 445552389692377960462926264922532772277746142428590181118909124891301297762535213527943984579645039755282830701731672880439131 51 |
| 14. | 15. ruj 2022. 14:19:48 | 48868615274924904 | 672278094486261922106834236611472472495737541815787234919770748245159201854140132228117049381360136069041518187246469733021840 40 |
| 15. | 15. ruj 2022. 14:20:46 | 103387330642609536 | 244567910726642221941482677640047399953578433790790969039094617 7 |

| | | | |
|---|---|---|---|
| | | | 818000514798681104000644619957017621222568926467733349551548597 |

Unutar tablice 5.1. nalaze se rezultati generiranja slučajnog broja. U prvom stupcu nalazi se vrijeme nastajanja slučajnog broja. Unutar drugog stupca prikazani su rezultati koji se dobiju iz jednadžbe prikazane slikom 4.17. tako što se uvrste očitanja senzora. Treći stupac je konačni rezultat koji je tražen. Taj rezultat se dobije nakon operacija sa SHA-512. Kao što se može vidjeti iz dobivenih rezultata, svi dobiveni brojevi su nasumični i nisu povezani jedni s drugima što znači da aplikacija funkcionira.

# 6. ZAKLJUČAK

Ideja ovog završnog rada bila je stvoriti aplikaciju koja će generirati slučajne brojeve. Generirani brojevi trebali su biti nepredvidljivi što znači da je generator trebao biti istinski. Generator je realiziran koristeći senzore mobilnog uređaja.

Tijekom stvaranja aplikacije nastao je problem kod završnog koraka. Aplikacija je trebala vraćati 128-znamenkasti broj. Uzimajući samo očitanja senzora bilo je nemoguće dobiti toliki broj jer vrijednosti očitanja senzora nisu dovoljno velika. Iz tog razloga morao se koristiti SHA algoritam.

Programsko je rješenje ostvareno no to ne znači da se aplikacija ne može poboljšati. Cilj ovog završnog rada ostvaren je koristeći 4 senzora, ali mobilni uređaji sadrže mnogo senzora, dodavajući još neki od njih, slučajni brojevi bili bi još slučajniji.

# LITERATURA

[1] S. Jurić, „Generatori pseudo-slučajnih brojeva“, FER, Zagreb, 2001., dostupno na: http://sigurnost.zemris.fer.hr/random/2001_juric/#CSPRNG [12.9.2022.]

[2] D. Beaver, N. So, “Global, unpredictable bit generation without broadcast ”, in Advances in Cryptology, Eurocrypt’93.

[3] A. Stefanov, N. Gisin, O. Guinnard, L. Guinnard, H. Zbinden, “Optical quantum randoom number generator”, in Journal of Modern Optics, Volume 47, Issue4, 2000, pp. 595-598.

[4] Random UX, https://play.google.com/store/apps/details?id=ru.uxapps.random [12.9.2022.]

[5] Pretty Random, https://play.google.com/store/apps/details?id=com.foxbytes.prettyrandom [12.9.2022.]

[6] RNG Plus,

https://play.google.com/store/apps/details?id=com.randomappsinc.randomnumbergeneratorplus [12.9.2022.]

[7] Meet Android Studio, https://developer.android.com/studio/intro [12.9.2022.]

[8] E. Mixon, AndroidOS, https://www.techtarget.com/searchmobilecomputing/definition/Android-OS [12.9.2022.]

[9] Android (operating system), https://en.wikipedia.org/wiki/Android_(operating_system) [12.9.2022.]

[10] Java (programming language), https://en.wikipedia.org/wiki/Java_(programming_language) [12.9.2022.]

[11] How does gyroscope sensor work in your smartphone?, https://www.techaheadcorp.com/knowledge-center/how-gyroscope-sensor-work-in-smartphone/ [12.9.2022.]

[12] Pressure sensing in smarphones provides advanced location info and more, https://allsensors.com/blog/pressure-sensing-in-smartphones-provides-advanced-location-info-and-more/ [12.9.2022.]

[13] Accelerometer, https://www.mathworks.com/help/supportpkg/android/ref/accelerometer.html [12.9.2022.]

[14] SHA-512 Hashing Algorithm Overview, https://komodoplatform.com/en/academy/sha-512/ [12.9.2022.]

[15] XML, https://hr.wikipedia.org/wiki/XML [12.9.2022.]

# SAŽETAK

Ovaj završni rad opisuje stvaranje aplikacije za generiranje slučajnih brojeva. Namijenjena je za korisnike Android pametnih uređaja. Vrsta slučajnih brojeva koja se generira koristeći ovu aplikaciju su istinski slučajni brojevi. Istinska slučajnost postiže se tako da se prilikom pritiska na gumb za generiranje brojeva zabilježe trenutni podaci zadanih senzora i ubace u jednadžbu. Rezultat jednadžbe ubacuje se u sigurni algoritam SHA-512 koji vraća hash čija je vrijednost heksadekadska. Ta heksadekadska vrijednost se pretvara nazad u decimalnu i vraća korisniku kao finalni slučajni broj.

**Ključne riječi:** generator slučajnih brojeva, geolokacija, istinski generator slučajnih brojeva, sha512

# ABSTRACT

**Mobile application for generating random numbers**

This undergraduate thesis describes the creation of a random number generation application. It is intended for users of Android smart devices. The type of random numbers generated by this application are “true” random numbers. The “true” randomness is achieved by pressing the number generation button to capture the instantaneous data of the default sensors and inserting them into the equation. The result of the equation shall be inserted into a secure SHA-512 algorithm that returns hash with hexadecadine value. This hexadecaden value is converted back to decimal and returned to the user as a final random number.

**Keywords**: random number generator, geolocation, true random number generator, sha512

# ŽIVOTOPIS

Leon Rinčić rođen je 11. lipnja 1999. godine u Osijeku. Nakon pohađanja Osnovne škole kralja Tomislava, upisuje smjer tehničar za elektroniku u Srednjoj školi Isidora Kršnjavog Našice. Srednjoškolsko obrazovanje završava 2018. godine te iste godine nastavlja školovanje na Fakultetu Elektrotehnike, Računarstva i Informacijskih Tehnologija Osijek, smjer računarstvo. Razvoj mobilnih aplikacija smatra izrazito zanimljivim i to u jezicima Java i Kotlin.

_____
Potpis autora