Android aplikacija za vertikalni skok

Klement, Goran

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

Sveučilišni studij

ANDROID APLIKACIJA ZA VERTIKALNI SKOK

Završni rad

Goran Klement

Osijek, 2021.

SADRŽAJ

1. UVOD

Vertikalni skok sastavni je dio kolektivnih sportova s loptom. U nekim sportovima kao što su košarka i odbojka vertikalni skok smatra se najpotrebnijom sposobnosti igrača. Vertikalni skok ovisi o dva faktora, o RFD (engl. *rate of force development*) te o reaktivnosti. Oba faktora su jako bitna kada je u pitanju povećanje visine vertikalnog skoka. [1]

RFD označava stopu za razvoj sile, odnosno predstavlja koliko brzo mišići mogu proizvesti silu kako bi skok bio veći. Tijekom košarkaške ili odbojkaške igre igrači nemaju dovoljno vremena kako bi se sagnuli nisko u čučanj i na taj način generirali silu kako bi skok bio viši. Razlog toga je dinamika igre ili utjecaj protivnika, stoga je vrlo bitno imati razvijen RFD kako bi u što kraćem vremenu mogli proizvesti veću odraznu silu. Vježbe koje su povezane s okretnošću nam omogućuju povećanje RFD-a. Takve vježbe spajaju se s vježbama snage i vježbama pilometrije u koje ubrajamo horizontalne i vertikalne poskoke. [1]

Drugi faktor, reaktivnost, poboljšava se dubinskim skokovima. Veća reaktivnost omogućava bolju apsorpciju sile tijekom doskoka, čime se smanjuje vjerojatnost ozljede koje se učestalo događaju. Osim povećanja vertikalnog skoka, trening reaktivnosti ima za svrhu i jačanje ligamenata i zglobova. [1]

Tema ovog završnog rada je izrada Android aplikacije koja sadrži vježbe pogodne za razvoj reaktivnosti te RFD-a. Samim time pogodne su i za povećanje vertikalnog skoka te mogućnost praćenja napretka. Aplikacija je jednostavna i instinktivna za korištenje. Tijekom izrade aplikacije korišten je Android Studio, službeno razvojno okruženje unutar kojeg se kreiraju Android aplikacije pomoću Java programskog jezika i XML-a (engl. *eXtensible Markup Language*). Za pohranu podataka te ugradnju mogućnosti praćenja napretka korišten je Firebase koji pruža specifične protokole za komunikaciju između klijenta i poslužitelja.

1.1. Zadatak završnog rada

Zadatak završnog rada je kreirati Android aplikaciju koja bi uključivala animacije vježbi raspoređenih po težini. Uz to, potrebno je ugraditi mogućnost praćenja napretka i napraviti odgovarajući prikaz korisniku.

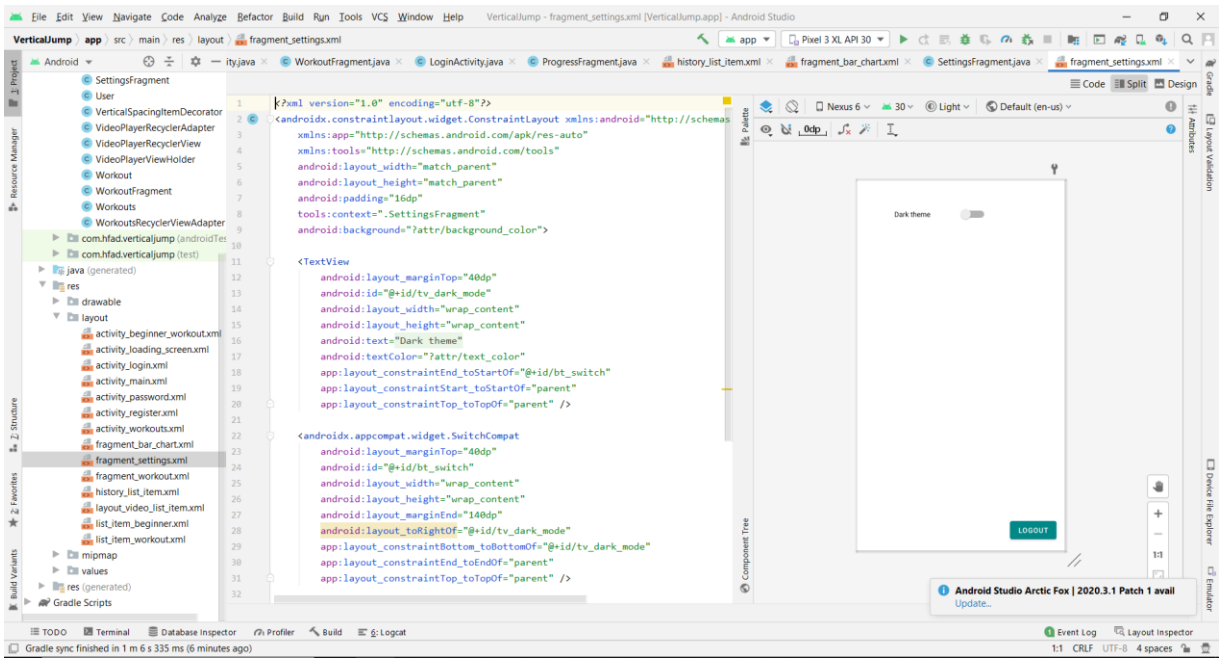
2. TEORIJSKA PODLOGA

U ovom poglavlju opisat će se korištene tehnologije pri izradi ove aplikacije s ciljem razumijevanja i lakšeg praćenja. Korištene tehnologije su Android Studio, operacijski sustav Android, programski jezik Java, XML, Firebase usluge te Adobe Photoshop.

2.1. Android Studio

Android Studio je prema [2] službeno integrirano razvojno okruženje (IDE, engl. *integrated development environment*) za operacijski sustav Android koji pripada Googleu. Prva stabilna verzija objavljena je u prosincu 2014. godine. Svrha Android Studioa je olakšati i unaprijediti razvoj aplikacija za Android uređaje. Isprva je službeni jezik za razvoj aplikacija bio Java programski jezik, no od 07. svibnja 2019. godine Kotlin je zamijenio Javu kao preferirani jezik.

Funkcionalnosti koje Android Studio omogućava su primjena promjena bez ponovnog pokretanja aplikacije, Gradle zasnovana podrška, ProGuard integracija, brzi emulator, mogućnost refaktoriranja koda i kratki popravci, bogati uređivač korisničkog sučelja koji omogućava korisnicima da kreiraju izgled aplikacije povlačenjem i ispuštanjem elemenata. Osim mobilnih uređaja ugrađena je i podrška i za Android Wear aplikacije, a jednostavno povezivanje aplikacije s Firebase oblakom i Android emulator olakšavaju posao osobama koje koriste Android Studio. Aplikacije mogu biti objavljene na Googleovom Play Storeu ukoliko poštuju pravila vezana uz sadržaj. [2] Na slici 2.1. prikazan je izgled razvojnog okruženja Android Studioa.



Sl. 2.1. *Izgled razvojnog okruženja Android Studioa*

2.2. Firebase baza podataka u stvarnom vremenu

Firebase je Googleova usluga za razvoj web i mobilnih aplikacija. Pruža korisnicima skup alata i usluga koji obavlja velik dio zadataka koje uobičajeno obavljaju programeri. Integracija aplikacije s Firebaseom jednostavna je pošto se sve može obaviti u nekoliko klikova. Jedan od alata koje pruža je i Firebase baza podataka u stvarnom vremenu koja omogućava korisnicima pohranu i sinkronizaciju podataka između korisnika u stvarnom vremenu. Podaci se spremaju u JSON (engl. *JavaScript Object Notation*) formatu. Firebase baza podataka u stvarnom vremenu je smještena na oblaku što znači da nema nedostataka održavanja servera. Ta funkcionalnost olakšava pristup podacima bilo s mobilnih ili web uređaja. Kada se podaci u bazi podataka u stvarnom vremenu promijene, promjene se spremaju na oblak i istovremeno se svi povezani uređaji obavještavaju u roku nekoliko milisekundi. [3]

Baza podataka u stvarnom vremenu je također pogodna i za korištenje bez povezanosti na mrežu. Ukoliko korisnik izgubi internetsku vezu, koristi se lokalna predmemorija uređaja za spremanje promjena. Kada se korisnik ponovno spoji na internet, lokalni podaci se automatski sinkroniziraju i prikazuju u aplikaciji. [3]

Govoreći o sigurnosti, vlasnik može odrediti kojim podacima korisnici mogu pristupiti i kako bi podaci trebali biti strukturirani. U nekoliko linija koda može se osigurati da korisnici imaju pristup samo svojim podacima pri čemu Firebase sam obavlja autentifikaciju i osigurava ispravan rad aplikacije. [3]

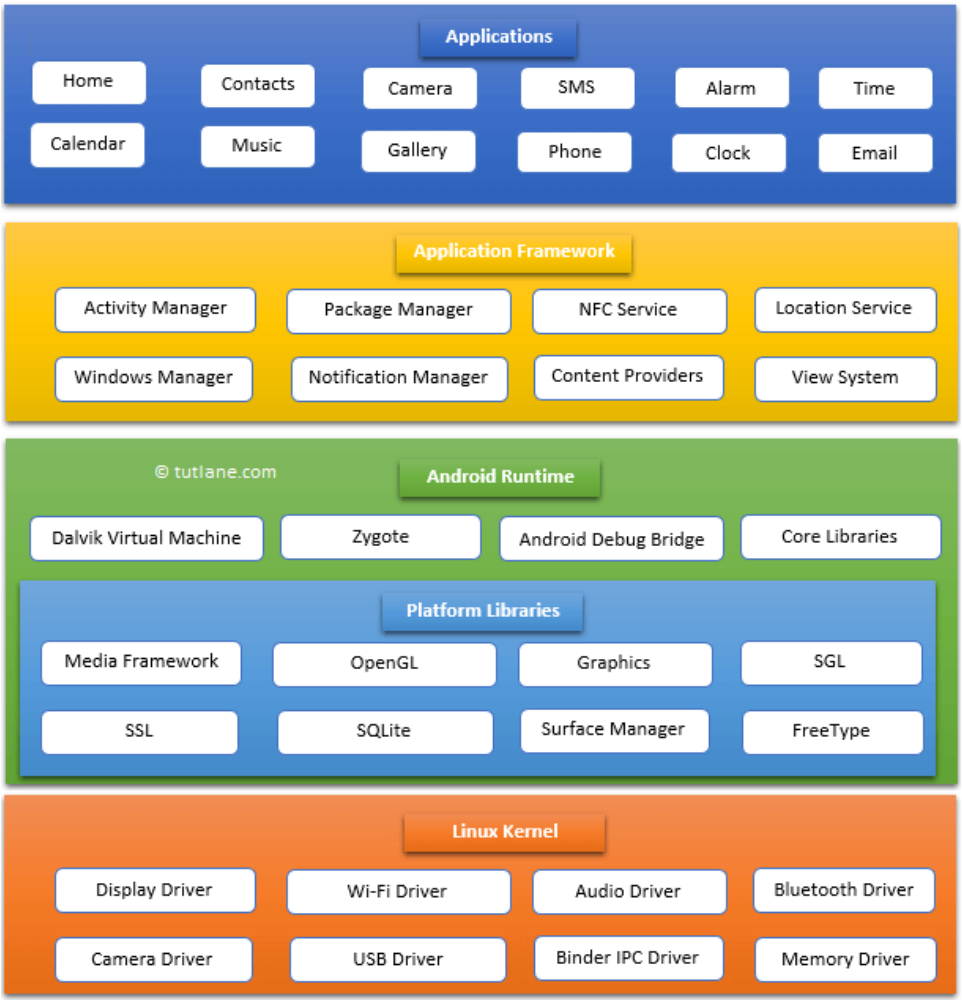
2.3. Operacijski sustav Android

Operacijski sustav Android je mobilni operacijski sustav baziran na Linux operacijskom sustavu koji je primarno kreiran za mobilne uređaje i tablete. Prvi put je javnosti predstavljen 2007. godine dok su se prvi uređaji koji su koristili Android pojavili 2008. godine. Android je trenutno najkorišteniji operacijski sustav za mobilne uređaje s oko 3 milijarde aktivnih korisnika. Neke od značajki Android operacijskog sustava su jednostavno i intuitivno korisničko sučelje, podrška za više dodirnih mjesta istovremeno, podrška za više jezika, povezivost putem Bluetootha, Wi-Fija, videopozivi, optimizirana grafika, GPS, upravljanja pomoću glasovnih naredbi i tako dalje. [4]

Govoreći o arhitekturi Android operacijskog sustava, prema [5] ona se dijeli na četiri sloja:

1. Linux jezgra koja je donji sloj arhitekture. Svrha joj je upravljanje upravljačkim uređajima kao što su upravljački uređaj za kameru, Bluetooth, memoriju i zaslon.
2. Android datoteke koje omogućuju podršku poput C/C++ datoteka i Java-zasnovanih datoteka.
3. aplikacijski okvir koji omogućuje usluge pomoću kojih se mogu kreirati klase. Uključuje usluge poput telefonskih usluga i lokacijskih usluga.
4. aplikacijski sloj uključuje nativne i ostale aplikacije poput glazbe, galerija, mobilnih igara.

   Građa arhitekture prikazana je detaljnije na slici 2.2.



Sl. 2.2. *Arhitektura Android operacijskog sustava* [5]

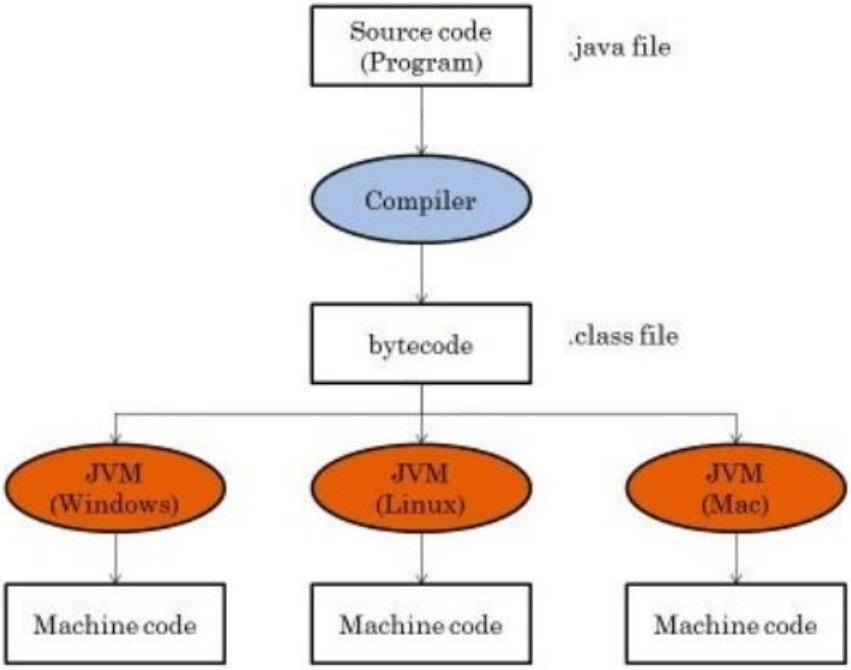
Upravo zbog toga što je Android Googleov proizvod, Android uređaji podržavaju sve Googleove usluge kao što su video usluge, email platforme i pohrana na oblaku. Temeljen je na

jezgri Linux te je izvorni kod javno dostupan (engl. *open source*) što znači da je besplatan za korištenje te ga svatko može koristiti. Google je također razvio i Android Auto za automobile, Wear OS za pametne satove te Android TV kojeg koriste pametni televizori. [6]

Trenutno je najnovija verzija Android 11. Prije Androida 10 nazivi verzija su bili nazivi slatkiša, tako da su dosadašnje poznatije verzije bile: Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo, Pie. Najveći konkurent mu je iOS, Appleov mobilni operacijski sustav. [4]

2.4. Programski jezik Java

Java je objektno orijentirani programski jezik objavljen 1995. godine. Razvili su ga James Gosling i ostali inženjeri iz tvrtke Sun Mycrosystems. Trenutno je jedan od najkorištenijih jezika s oko 10 milijuna korisnika. Inovacija koju je Java unijela je da se programi pisani u Javi pomoću JVM (engl. *Java Virtual Machine*) mogu izvoditi na svim platformama koje podržavaju Javu. To je moguće jer se kod prevodi u Java bytecode (inače se kod prevodi direktno u strojni jezik). Java bytecode je niz instrukcija koje JVM može pokrenuti bez obzira na arhitekturu računala. To je moguće jer prevoditelj kreira strojni kod u obliku .class datoteke.[7] Detaljniji prikaz nalazi se na slici 2.3.



Sl. 2.3. *Slijed pretvorbe .java datoteka u strojni kod* [8]

Java se koristi u slučajevima gdje je najbitnija brzina razvoja, a zahvaljujući hermetički zatvorenom okolišu programi se razvijaju s manje pogrešaka. [7]

Sintaksno je Java programski jezik sličan C++ programskom jeziku, dakle postoje klase i objekti tih klasa. Razlog tome je što je razvoj Jave inspiriran C jezikom, no pruža bolju sigurnost i pouzdanost. Java koristi automatski sakupljač smeća koji upravlja životnim ciklusom objekata a samim time i memorijom. Programer je taj koji odlučuje kada se objekt u Javi stvara, a Java čisti memoriju od objekata koji se više ne koriste. [7]

Neke od primjena Jave su: razvoj Android aplikacija, analiza podataka, razvoj računalnih aplikacija, znanstvene primjene, web serveri. Java ima podršku za jMonkeyEngine, moćan 3D-Engine koji ima veliku primjenu u razvoju 3D igara.

2.4.1.1. RecyclerView

RecyclerView je komponenta koja se koristi za učinkoviti prikaz velikog broja objekata u obliku liste. Nasljednik je ListViewa i GrindViewa. Najveća prednost mu je što može prikazivati podatke čiji se elementi mijenjaju u stvarnom vremenu. [9]

Kao što to i samo ime implicira, RecyclerView reciklira elemente. To radi na način da kada se zaslon pomakne te element liste izađe iz zaslona, RecyclerView ne uništava taj pogled nego ga ponovno koristi za novi element koji je ušao u zaslon. Ponovno korištenje poboljšava performanse, smanjuje potrošnju baterije i poboljšava odziv aplikacije. [9]

Ključni elementi koje omogućuju ispravno funkcioniranje RecyclerViewa su:

1. RecyclerView koji se umeće u korisničko sučelje, u njemu se prikazuju podaci
2. onCreateViewHolder metoda koja kreira i reciklira ViewHolder
3. onBindViewHolder metoda koja povezuje podatke koji se nalaze u adapteru i ViewHolder

2.4.1.2. ViewPager

ViewPager komponenta omogućuje pomicanje zaslona lijevo ili desno da bi se otvorio novi sadržaj. Pošto se ViewPager kombinira s fragmentima, sadržaj se učitava u fragmente, a da bi to bilo moguće treba postojati adapter za ViewPager. ViewPager objekti imaju ugrađene geste za pomicanje zaslona te su animacije pri pomicanju automatski ugrađene.

2.4.1.3. Fragment

Fragment je klasa koja je dodatni sloj između aktivnosti i korisničkog sučelja. Fragment ima svoj vlastiti izgled sučelja i životni ciklus te može rukovati vlastitim ulaznim događajima. Životni ciklus fragmenta usko je povezan sa životnim ciklusom aktivnosti što znači da mora biti ugošćen od strane aktivnosti ili drugog fragmenta. U jednu aktivnost moguće je ugraditi više fragmenata. [10]

Isto kao i aktivnost, fragment mora imati svoju datoteku korisničkog sučelja i Java klasu. Fragmente upotrebljavamo pomoću FragmentManager klase koja omogućava dodavanje, uklanjanje i izmjenu fragmenata unutar aplikacije.

2.4.1.4. TabLayout

TabLayout omogućuje prikaz tabova. Tabovi se dodaju pomoću newTab() funkcije, a pomoću setIcon() ili setText() funkcija može se definirati izgled TabLayouta. Potpunu funkcionalnost TabLayout dobije kada se upotrebljava zajedno s ViewPagerom, jer se time omogućuje pomicanje među ponuđenim tabovima. [11]

2.5. XML

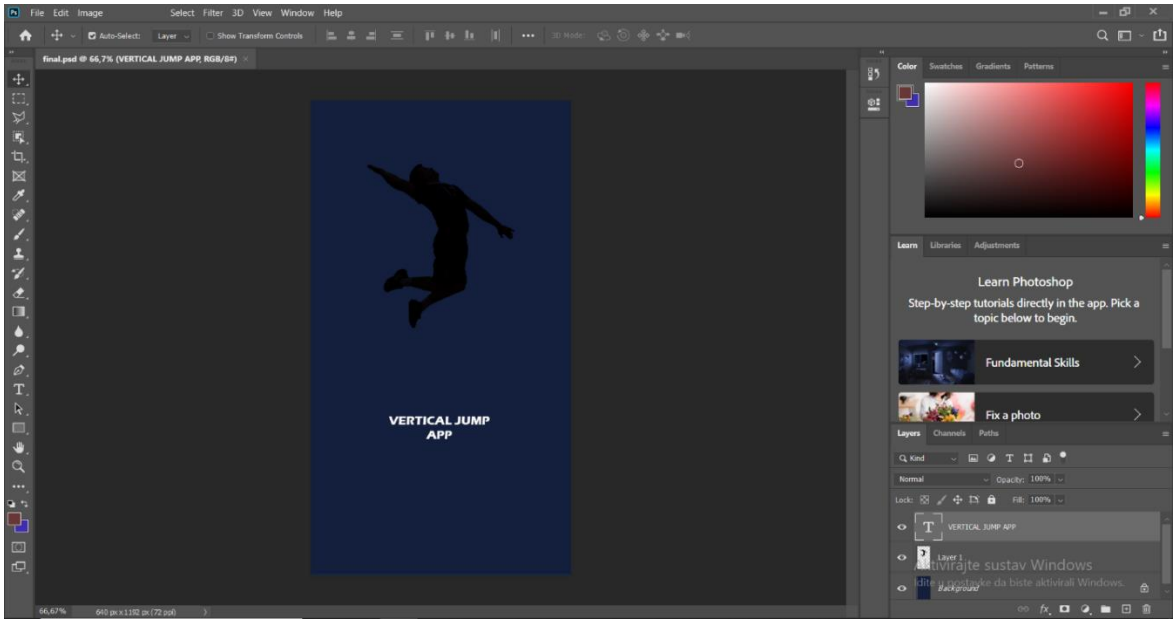
XML (engl. *eXtensible Markup Language*) je opisni jezik koji se koristi za definiranje dizajna aplikacije. Kod Android aplikacija koristi se za definiranje izgleda fragmenata i aktivnosti, a koristi se na način da se elementi kreiraju pomoću oznaka. Oznake su dio pravila koje se koriste za označavanje podataka da bi tekst bio čitljiv tekst editorima i lako razumljiv ljudima. XML se najčešće sprema kao tekstualna datoteka, a sastoji se od oznake i sadržaja.XML je potpuno neovisan o računalnoj platformi na kojoj se nalazi i operacijskom sustavu. [12]

Danas je XML vrlo raširen jezik koji se primjenjuje za razdvajanje podataka od prezentacije, za pohranu i razmjenu podataka i povećanje dostupnosti podataka. Njegova prednost nad ostalim tehnologijama su neovisnost, prenosivost i prilagodljivost korisničkim potrebama. Standardiziran je jezik, a o definiranju XML-a brine W3C (engl. *World Wide Web Consortium*). W3C održava WWW standarde, a u veljači 1998. godine objavio je prvu verziju preporuka za XML. [12]

2.6. Adobe Photoshop

Adobe Photoshop je grafički program za uređivanje koji je objavila tvrtka Adobe Systems. Prva verzija postala je dostupna 1990. godine. Trenutno je vodeći program za uređivanje i obradu slika.

Ima brojne primjene, među najpoznatijima su izrezivanje fotografija, mijenjanje rezolucije slike, uređivanje boja, uklanjanje zamagljivanja i crvenih očiju, mogućnost podjele slike u slojeve radi jednostavnijeg uređivanja. Upravo zbog podjele slika u slojeve korisnici najčešće odabiru taj program jer se vrlo precizno može urediti fotografija, a uklanjanjem sloja se može vidjeti učinak uređivanja ili čak i poništiti učinjeno. Pri izradi ovog rada Adobe Photoshop je korišten za uređivanje početne slike koja se pojavljuje tijekom učitavanja aplikacije. Izgled Adobe Photoshop aplikacije prikazan je na slici 2.4.



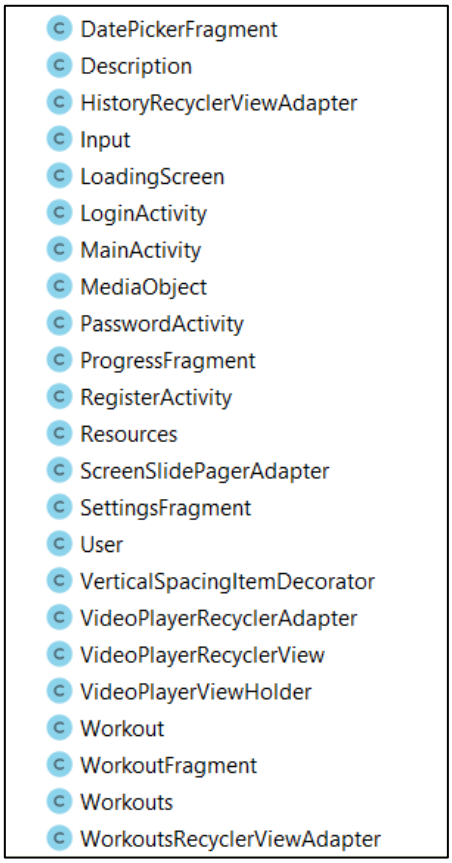
Sl. 2.4. *Izgled aplikacije Adobe Photoshop*

3. IZRADA ANDROID APLIKACIJE ZA VERTIKALNI SKOK

Za izradu ove aplikacije korišten je Android Studio. Aplikacija je napravljena u programskom jeziku Java. U ovom dijelu bit će pojašnjeni najbitniji dijelovi aplikacije te će biti prikazan dio koda aplikacije.

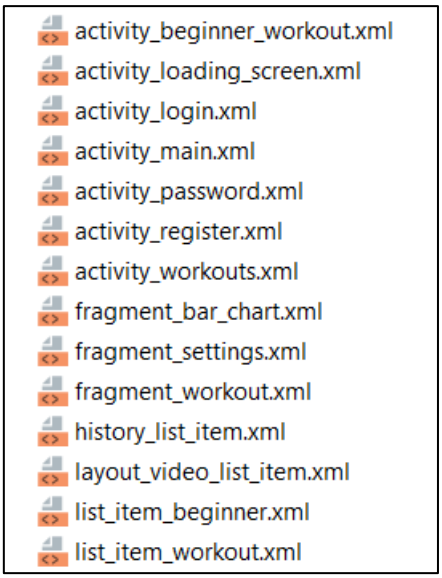
3.1. Uvod u aplikaciju

Svaka Android aplikacija sastavljena je od tri glavne datoteke: „manifest“, „java“ i „res“. Manifest datoteka sadrži metapodatke o aplikaciji što uključuje naziv paketa, nazive aktivnosti, dopuštenja, verziju podrške za Android. U java datoteci nalazi se kod napisan u Java programskom jeziku, a res mapa sadrži razne resurse poput stringova, slika i XML datoteka korisničkog sučelja. Popis svih korištenih Java datoteka prikazan je na slici 3.1.



Sl. 3.1. *Korištene Java datoteke*

Na slici 3.2. prikazane su XML datoteke korisničkog sučelja.



Sl. 3.2. *Korištene XML datoteke korisničkog sučelja*

Još jedan bitan dio aplikacije je Gradle skripta. U njoj se nalazi popis korištenih vanjskih knjižica čija je svrha uključiti u projekt implementacije elemenata poput RecyclerViewa i Firebasea čija je svrha već pojašnjena, Glide koji se koristi za učitavanje slika u ImageView, Material koji služi za uređivanje elemenata korisničkog sučelja, ExoPlayer koji omogućuje prikaz videozapisa unutar FrameLayouta. Popis svih korištenih implementacija prikazan je na slici 3.3.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'com.google.android.material:material:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation "androidx.recyclerview:recyclerview:1.2.0"
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'com.google.firebase:firebase-database:20.0.0'
    implementation 'com.github.bumptech.glide:glide:4.11.0'
    implementation 'com.google.firebase:firebase-auth:21.0.1'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.11.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'
    implementation "com.google.android.exoplayer:exoplayer:2.8.4"
    implementation "com.github.bumptech.glide:glide:4.9.0"
    annotationProcessor "com.github.bumptech.glide:compiler:4.9.0"

}
```

Sl. 3.3. *Gradle skripta*

3.2. Prijava korisnika

Kad se pokrene aplikacija, pokreće se se *LoadingScreen* aktivnost koja učitava početnu sliku na 2000 milisekundi. Izgled slike bit će prikazan naknadno u dijelu koji prikazuje izgled same aplikacije. Nakon što prođe 2000 milisekundi pomoću Intenta pokreće se *LoginActivity* koji omogućuje korisniku prijavu u aplikaciju preko emaila i lozinke koje je sam odredio. Cijeli sustav prijave odvija se preko Firebase autentifikacije koja osigurava da svaki korisnik dobije svoje podatke te da lozinka i email budu zaštićeni. Firebase sam obavlja posao provjere korisnika u sustavu na način da usporedi uneseni email i lozinku s podacima koji se već nalaze u bazi podataka.

Ukoliko su tijekom prijave email ili lozinka netočni odnosno ne postoje u bazi podataka, izbacuje se poruka da je prijava neuspješna, što je prikazano na slici 3.4.

```
mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            startActivity(new Intent( packageContext: LoginActivity.this, MainActivity.class));
    }
        else{

            Toast.makeText( context: LoginActivity.this,  text: "Failed to login!", Toast.LENGTH_LONG).show();
        }
    }
});
```

Sl. 3.4. *Metoda za provjeru uspješnosti autentifikacije*

3.3. Registracija korisnika i zaboravljena lozinka

Ukoliko korisnik još nije registriran, na *LoginActivityju* može pritiskom na „Register“ labelu otvoriti RegisterActivity koji prikazuje obrazac za registraciju. Budući korisnik treba unijeti željeni email i svoju lozinku te pritiskom na gumb prihvatiti registraciju.

Uvjeti za registraciju novog korisnika su:

1. Polje za email ne smije biti prazno
2. Email mora sadržavati znak „@“ i naziv usluge koju koristi čime se osigurava valjanost emaila
3. Polje za lozinku ne smije biti prazno
4. Lozinka mora imati minimalno 6 znakova

U slučaju da korisnik ne ispuni zadane uvjete prikazuje mu se upozorenje što nije valjano i iz kojeg razloga. Također fokus se vraća na ono što nije ispravno kako bi korisniku bilo jasnije što nije u redu. To se postiže funkcijom sa slike 3.5.

```
private void registerUser() {
    String email= et_email.getText().toString().trim();
    String password= et_password.getText().toString().trim();

    if(email.isEmpty()) {
        et_email.setError("Email is required!");
        et_email.requestFocus();
        return;
    }

        if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
             et_email.setError("Please enter valid email");
             et_email.requestFocus();
             return;
        }

        if(password.isEmpty()){

            et_password.setError("Password is required!");
            et_password.requestFocus();
            return;
        }

        if(password.length() < 6){
            et_password.setError("At least 6 characters is required!");
            et_password.requestFocus();
            return;

        }
```

Sl. 3.5. *Provjera valjanosti podataka za registraciju*

Ukoliko su u polja uneseni ispravni email i lozinka, kreira se novi korisnik koji se zapisuje u bazu podataka pod svojim identifikacijskim brojem koji je jedinstven u cijeloj bazi podataka. Novi korisnik je nakon toga u mogućnosti koristiti aplikaciju. Metoda koja provjerava uspješnost registracije prikazana je na slici 3.6.

Ukoliko već registrirani korisnik zaboravi lozinku, pritiskom na „*Forgot password*“ labelu otvara se *PasswordActivity* koji formom liči na obrazac za unos emaila. Sustav pomoću metode sa slike 3.7. šalje korisniku email čiji je sadržaj link koji otvara obrazac za unos nove lozinke. Nova lozinka se potom sprema u sustav te se korisnik ubuduće prijavljuje preko nove lozinke.

```
mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()){
                    User user = new User(email);
                    FirebaseDatabase.getInstance().getReference( path: "Users")
                            .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                            .setValue(user).addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                            if(task.isSuccessful()){
                                Toast.makeText( context: RegisterActivity.this,  text: "User has been registered successfully", Toast.LENGTH_LONG).show();
                                //povratak na login
                            }
                                else{
                                    Toast.makeText( context: RegisterActivity.this,  text: "Failed to register new user!", Toast.LENGTH_LONG).show();

                                }

                        }
                    });
                } else {
                    Toast.makeText( context: RegisterActivity.this,  text: "Failed to register new user!", Toast.LENGTH_LONG).show();

                }
            }
        });
```

Sl. 3.6. *Metoda za registraciju korisnika*

```
mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            startActivity(new Intent( packageContext: LoginActivity.this, MainActivity.class));
    }
        else{

            Toast.makeText( context: LoginActivity.this,  text: "Failed to login!", Toast.LENGTH_LONG).show();
        }
    }
});
```

Sl. 3.7. *Funkcija za slanje emaila za oporavak lozinke*

3.4. Fragmenti

Nakon uspješne prijave u aplikaciju, pokreće se *MainActivity* koji u sebi ima ViewPager s 3 fragmenta (*WorkoutFragment*, *ProgressFragment* i *SettingsFragment*). Uz ViewPager koristi se i TabLayout za intuitivnije korištenje aplikacije. Kod *MainActivitya* prikazan je na slici 3.8.

```
public class MainActivity extends AppCompatActivity {
    private ViewPager mViewPager;
    private TabLayout mTabLayout;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initViews();
        setUpPager();

    }
    private void initViews() {

        mViewPager = findViewById(R.id.viewPager);
        mTabLayout = findViewById(R.id.tabLayout);
    }
    private void setUpPager() {
        List<Fragment> fragmentList = new ArrayList<~>();
        fragmentList.add(WorkoutFragment.newInstance());
        fragmentList.add(ProgressFragment.newInstance());
        fragmentList.add(SettingsFragment.newInstance());
        PagerAdapter adapter= new ScreenSlidePagerAdapter(getSupportFragmentManager(), fragmentList);
        mViewPager.setAdapter(adapter);
        mTabLayout.setupWithViewPager(mViewPager);
        mTabLayout.getTabAt( index: 0).setIcon(R.drawable.ic_baseline_fitness_center_24);
        mTabLayout.getTabAt( index: 1).setIcon(R.drawable.ic_baseline_timeline_24);
        mTabLayout.getTabAt( index: 2).setIcon(R.drawable.ic_baseline_settings_24);

    }

}
```

Sl. 3.8. *MainActivity*

Kao što se vidi iz koda, dodavanje novih fragmenata obavlja se na jednostavan način. Nova instanca fragmenta dodaje se u listu fragmenata, a ta lista prosljeđuje se PagerAdapteru. Fragmenti se u aplikaciji pojavljuju redoslijedom kojim su dodani, a za svaki fragment dodan je tab za ljepši izgled i lakše korištenje. Svakom tabu dana je ikona koja odgovara onome što fragment prikazuje. Fragmenti se mogu mijenjati pomicanjem prsta lijevo ili desno ili klikom na sliku taba.

3.5. Lista vježbi

Na početnom zaslonu aplikacije pojavljuje se WorkoutFragment koji u sebi sadrži RecyclerView koji sadrži 3 elementa. Svaki od elemenata sastoji se od jednog *ImageViewa* koji prikazuje sliku medalje koja označava težinu vježbi (lagano, srednje ili napredno) i jednog *TextViewa* koji tekstualno govori težinu vježbi. Korisniku se klikom na elemente otvara liste vježbi ovisno o težini koju je odabrao. Svi tekstovi i slike umetnute su u Firebase bazu podataka kako bi aplikacija bila optimizirana. Učitavanje podataka iz baze podataka prikazano je na slici 3.9.

```
private void getDataFromFirebase() {
    Query query = root.child("workout");
    query.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                Workout workout = dataSnapshot.getValue(Workout.class);
                workoutList.add(workout);
            }
            adapter.notifyDataSetChanged();
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```

Sl. 3.9. *Metoda za dohvaćanje podataka iz baze*

Baza podataka funkcionira na način da se u bazu podataka umetnu podaci koji kao atribute imaju iste atribute kao i podatkovna klasa. Podatkovna klasa je klasa koja samo sadrži atribute, konstruktore i gettere i settere. Ti podaci spremljeni su u jedan nadređeni čvor, a preko reference na bazu podataka dohvatimo taj nadređeni čvor. Firebase potom učitava svu djecu čvora kao objekte te ih sprema u *ArrayList*. Na taj način se podaci iz online baze podataka prenose u program. U ovom slučaju lista objekata predaje se RecyclerViewAdapteru koji potom nastanjuje elemente RecyclerViewa elementima liste. Nastanjivanje se odvija u onBindViewHolder metodi, a kod je prikazan na slici 3.10.

```
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
    holder.tv_stage.setText(workouts.get(position).getText());


    Glide.with(mContext)
            .load(workouts.get(position).getMedal())
            .into(holder.iv_medal);

    Glide.with(mContext)
            .load(workouts.get(position).getNext())
            .into(holder.iv_next);
}
```

Sl. 3.10. *Metoda za naseljavanje elemenata RecyclerViewa*

WorkoutFragment klasa implementira *OnWorkoutListener* sučelje koje ima fukncij*u onWorkoutClick*. Ovisno o rednom broju liste na koji je korisnik kliknuo prikazuje se lista vježbi na način da se Intentu kao dodatan parametar doda redni broj. Sve to prikazano je na slikama 3.11. i 3.12.

```
@Override
public void onWorkoutClick(int position) {
    Intent intent= new Intent( WorkoutFragment.this.getActivity(), Workouts.class);
    intent.putExtra( name: "position", position);
    startActivity(intent);
}
```

Sl. 3.11. *Prepisana funkcija OnWorkoutListener sučelja*

```
public interface  OnWorkoutListener{
    void onWorkoutClick(int position);
}
```

Sl. 3.12. *OnWorkoutListener sučelje*

Nakon odabira težine otvara se nova aktivnost koja prikazuje video prikaz vježbi i broj ponavljanja unutar RecyclerViewa. Svi videozapisi pohranjeni su pomoću još jedne usluge Firebasea koja se zove Firebase storage. Firebase storage omogućava online pohranu videozapisa ili fotografija pri čemu se jednostavno može doći do URL-a (engl. *Uniform Resource Locator*) pohranjene datoteke. URL je potreban za dohvaćanje videozapisa iz baze podataka u aplikaciju,

pri čemu se *FrameLayout* iz XML datoteke popunjava videozapisom. Dio koda XML datoteke prikazan je na slici 3.13.

```
<TextView
    android:id="@+id/title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginEnd="10dp"
    android:layout_marginBottom="15dp"
    android:padding="10dp"
    android:textColor="#000"
    android:textSize="20sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toTopOf="@id/media_container"
    />


<FrameLayout
    android:layout_width="560dp"
    android:layout_height="300dp"
    android:layout_marginBottom="10dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/title"
    android:background="#000"
    android:scaleType="center"
    android:id="@+id/media_container">
```

Sl. 3.13. *Dio koda XML datoteke korisničkog sučelja za dodavanje videozapisa*

3.6. Praćenje rezultata

U aplikaciju je ugrađena i mogućnost praćenja napretka. Za potrebu toga kreiran je *ProgressFragment* koji se sastoji od dijela za prikaz dosadašnjih rezultata i dijela u kojemu se odabire novi rezultat. Korisnik može odabrati datum, unijeti visinu svog vertikalnog skoka te klikom na gumb pohraniti svoj rezultat u bazu podataka. Pošto se za pohranu koristi baza podataka u stvarnom vremenu, čim korisnik unese podatke o iznosu vertikalnog skoka odmah mu se taj rezultat dodaje u listu i prikazuje na ekranu kao zadnja unesena vrijednost. Na taj način jednostavno se može pratiti napredak.

Za implementaciju navedenog potrebno je koristiti *DatePicker*. *DatePicker* je komponenta koja omogućava jednostavan odabir datuma (slika 3.14.). Klikom na gumb na ekranu se prikaže kalendar sa širokim rasponom odabira datuma, tako da korisnici mogu odabrati proizvoljan datum.

```
btn_pick= getView().findViewById(R.id.btn_pick);
btn_pick.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        DialogFragment datePicker= new DatePickerFragment();
        datePicker.setTargetFragment( fragment: ProgressFragment.this, requestCode: 0);
        datePicker.show(getFragmentManager(), tag: "date picker");

    }
});
```

Sl. 3.14. *Metoda za prikaz DatePickera*

Korisnik odabire željeni datum te se on nakon formatiranja prikazuje u *TextViewu* kako bi korisnik mogao provjeriti još jednom je li uneseni datum ispravan.

Formatiranje se obavlja pomoću *format()* metode. Upravo zbog ovog dijela u aplikaciju je bilo potrebno ugraditi mogućnost registracije i prijave kako bi svaki korisnik dobio svoje podatke vezane za visinu i datum vertikalnog skoka. U slučaju da ta mogućnost nije bila ugrađena, svi podaci spremali bi se na isto mjesto te bi svaki korisnik koji otvori aplikaciju vidio sve do tada unesene rezultate svih korisnika zajedno. Na slici 3.15. prikazan je dio formatiranja stringa.

```
@Override
public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
    Calendar c= Calendar.getInstance();
    c.set(Calendar.YEAR, year);
    c.set(Calendar.MONTH, month);
    c.set(Calendar.DAY_OF_MONTH, dayOfMonth);
    String currentDateString= DateFormat.getDateInstance().format(c.getTime());
    tv_date.setText(currentDateString);

}
```

Sl. 3.15. *Formatiranje stringa u onDateSet metodi*

3.7. Postavke i odjava

Svrha posljednjeg, SettingsFragmenta, je omogućiti korisniku odabir načina rada i pružiti mogućnost odjave iz aplikacije. Sam fragment prilično je jednostavan te se sastoji od jednog gumba na čiji pritisak se korisnik odjavljuje iz aplikacije, što znači da se opet pri paljenju aplikacije mora ulogirati pomoću lozinke i emaila. Drugi element koji čini fragment je *SwitchCompat* odnosno sklopka koja ima 2 stanja, upaljeno i ugašeno. Komponenta je instinktivna za korištenje, a svrha joj je mijenjati način rada aplikacije između tamnog i svijetlog. Tamni način rada kao pozadinsku boju koristi crnu boju, a tekst je prikazan bijelom bojom. Kod svijetlog načina rada je obratno, pozadina je bijele boje dok je tekst crne. Da bi takva funkcionalnost bila moguća potrebno je u svaku aktivnost ili fragment unutar *onCreateView()* metode pročitati stanje sklopke i ovisno o tome odabrati temu aplikacije. To je prikazano na slici 3.16.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    if(AppCompatDelegate.getDefaultNightMode() == AppCompatDelegate.MODE_NIGHT_YES) {
        getActivity().setTheme(R.style.Theme_Dark);
    } if(AppCompatDelegate.getDefaultNightMode()==AppCompatDelegate.MODE_NIGHT_NO) {
        getActivity().setTheme(R.style.Theme_Light);
    }
    return inflater.inflate(R.layout.fragment_bar_chart, container, attachToRoot: false);
}
```

Sl. 3.16. *Odabir načina rada unutar onCreateView metode*

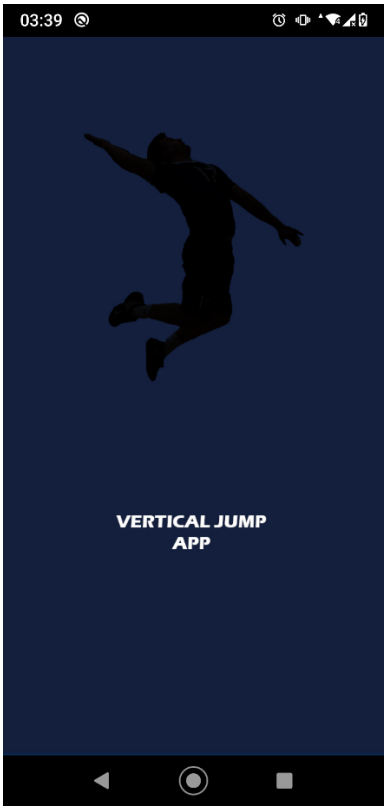
To je jedini fragment u aplikaciji koji ne koristi RecyclerView niti Firebase bazu podataka.

4. IZGLED I KORIŠTENJE APLIKACIJE

U ovom poglavlju opisat će se korištenje aplikacije te će biti prikazane slike zaslona s mobilnog uređaja na kojemu je aplikacija pokrenuta.

4.1. Početna slika

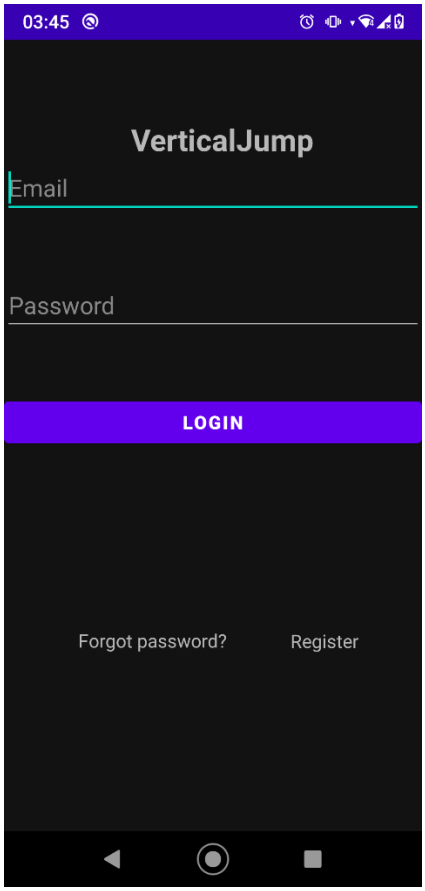
Pomoću programa Adobe Photoshop kreirana je slika prikazana na slici 4.1. Ta slika ima kao ulogu poboljšati korisničko sučelje te prikazati naziv aplikacije. U trenutku kad se aplikacija pokrene 2000 milisekundi je prikazana navedena fotografija, a nakon isteka tog vremena otvara se dio vezan uz prijavu odnosno registraciju u sustav.



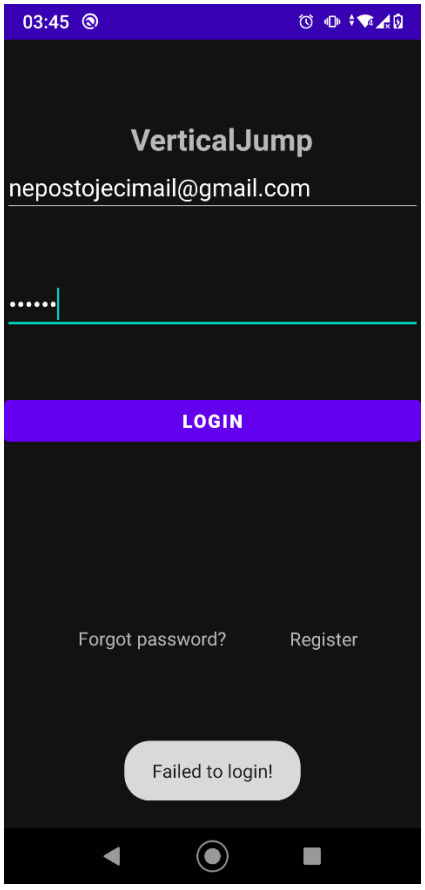
Sl. 4.1. *Uvodna slika*

4.2. Prijava u aplikaciju

Zaslon aktivnosti za prijavu prikazan je na slici 4.2. Aplikacija ima omogućene sve potrebne aktivnosti vezane uz prijavu u sustav. Unutar 2 *EditTexta* unose se email i lozinka, pomoću nagovještaja je objašnjeno koji se podatak gdje upisuje. Ukoliko je unos emaila i lozinke u redu, klikom na login gumb otvara se *MainActivity*, a ukoliko je unos netočan odnosno nepostojeći u bazi podataka, korisnik će dobiti poruku kao na slici 4.3. gdje unesena email adresa ne postoji u sustavu.



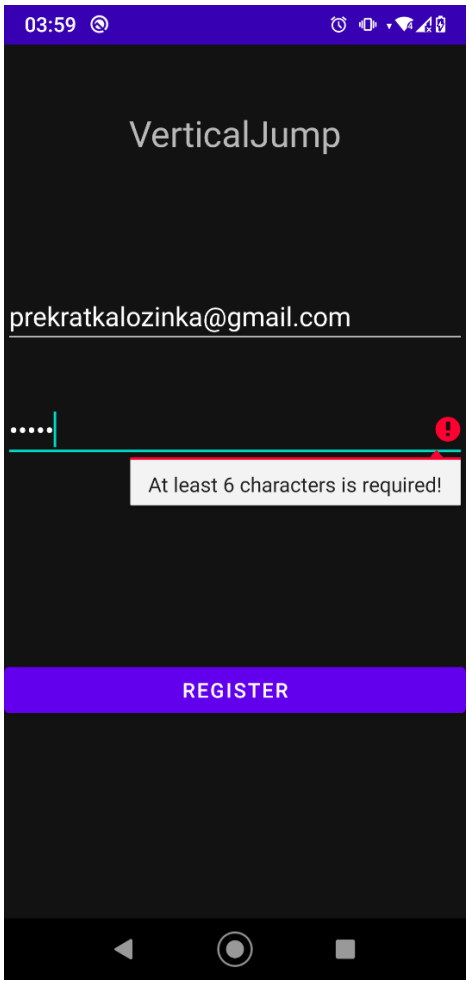
Sl. 4.2. *Izgled LoginActivityja*



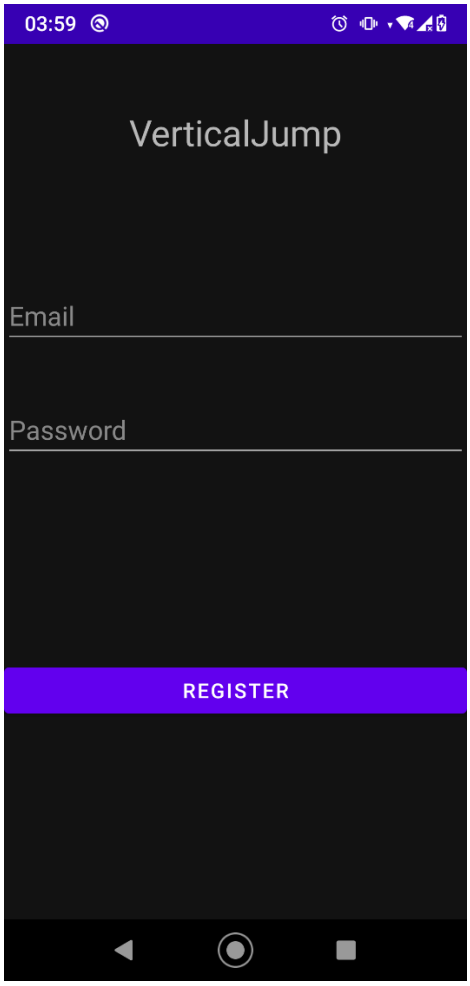
Sl. 4.3. *Toast poruka zbog nepostojećeg korisnika*

4.3. Registracija

Prikaz aktivnosti za registraciju prikazan je na slici 4.4. Aplikacija ne dopušta registraciju bez poštivanja normi vezanih uz email ili uz dužinu lozinke.



Sl. 4.4. *Greška kod unosa lozinke*

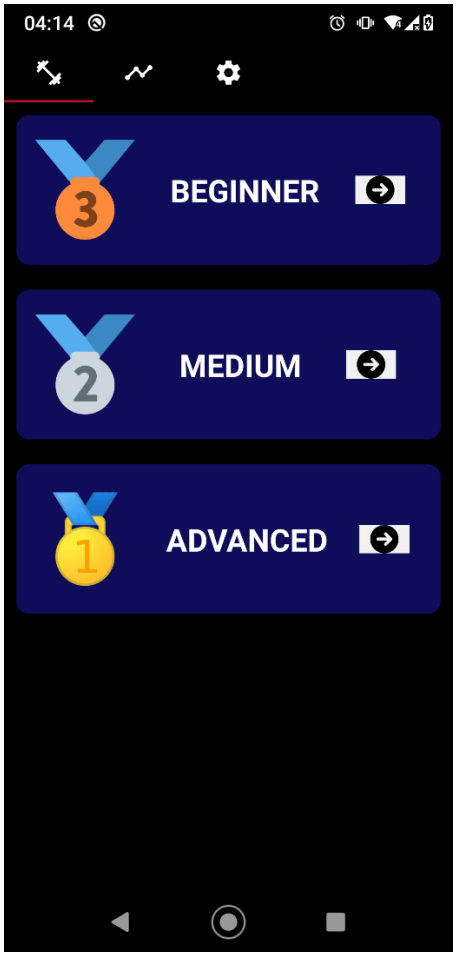


Sl. 4.5. *Prikaz izgleda RegisterActivityja*

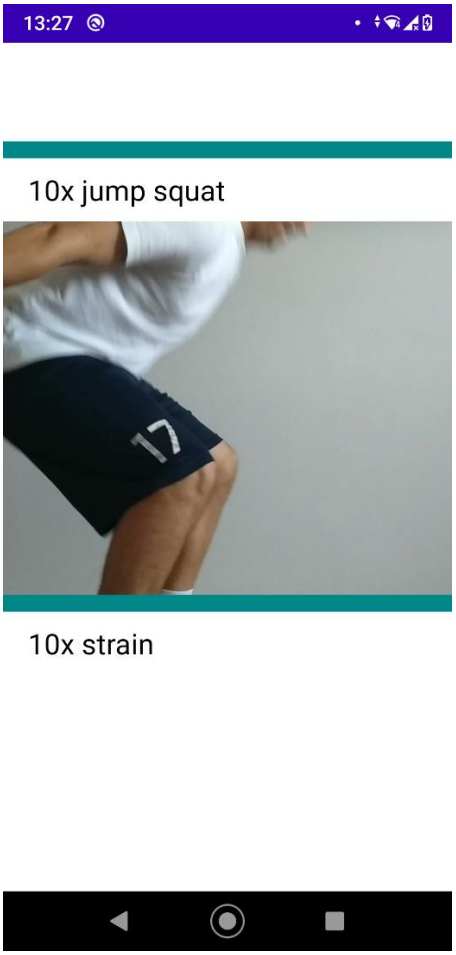
4.4. WorkoutFragment i vježbe

Fragment koji sadrži mogućnost odabira između 3 razine, raspoređeno po težini nazvan je WorkoutFragment. Elementi unutar RecyclerViewa stavljeni su u CardView kojem je dana blaga elevacija da bi se postigao 3D dojam. Klikom na jedan od elemenata u listi otvara se nova aktivnost koja sadrži listu vježbi. Nažalost, teško je postići da su svi videozapisi istovremeno učitani tako da se videozapisi učitavaju jedan po jedan ovisno o pomicanju gore-dolje po zaslonu mobilnog uređaja.

Izgled fragmenta prikazan je na slici 4.6., a jedna od vježbi prikazana je na slici 4.7.



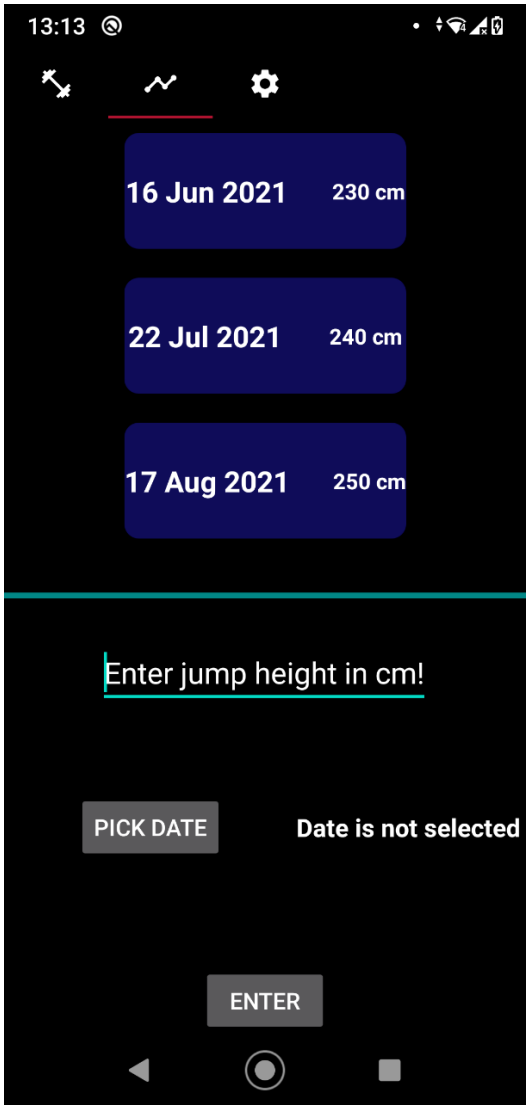
Sl. 4.6. *Prikaz WorkoutFragmenta*



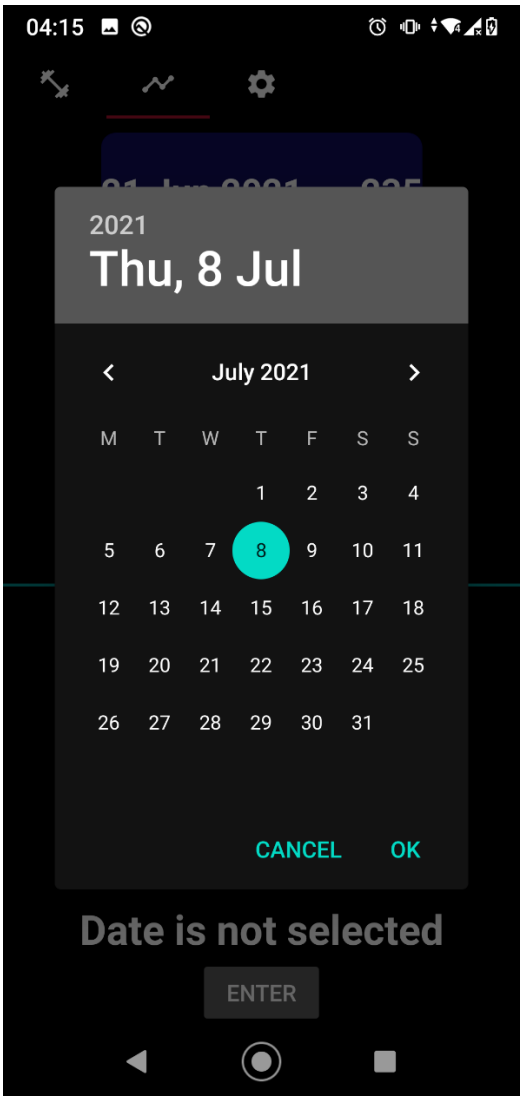
Sl. 4.7. *Prikaz videozapisa nakon odabira*

4.5. ProgressFragment

U ProgressFragmentu prikazani su RecyclerView u kojem su prikazani do sada pohranjeni rezultati vezani uz visinu vertikalnog skoka te dio koji omogućuje unos novog rezultata u listu. Za unos rezultata potrebno je pohraniti datum skoka i doseg skoka u centimetrima. Datum skoka unosi se na jednostavan način da se klikom na „Pick date“ gumb otvori prozor DatePickera koji je prikazan na slici 4.9., a iznos skoka korisnik sam mora izmjeriti i unijeti u aplikaciju. Na slici 4.8. prikazan je izgled cijelog fragmenta.



Sl. 4.8. *Prikaz ProgressFragmenta*

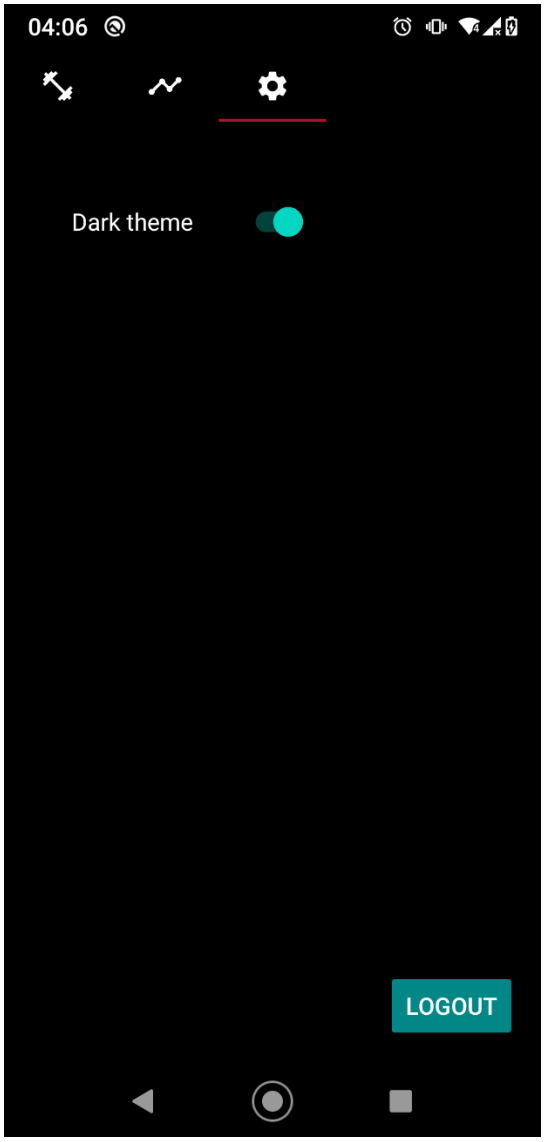


Sl. 4.9. *Prikaz DatePickera*

4.6. SettingsFragment

Posljednji fragment u nizu je SettingsFragment koji ima samo dvije funkcionalnosti, a to su promjena teme aplikacije i odjava korisnika. U ovom dijelu prikazat će se izgled tog fragmenta kada je sklopka uključena (tamni način rada) i kada je sklopka isključena (svijetli način rada).

Sl. 4.10. *Isključen tamni način rada*



Sl. 4.11. *Uključen tamni način rada*

5. ZAKLJUČAK

Sve je veća primjena mobilnih uređaja u svakodnevnim aktivnostima, a fitness aplikacije jedan su dio primjene. Ova aplikacija bavi se temom treninga i analize vertikalnog skoka odnosno omogućava korisniku da postigne bolje rezultate i prati navedene rezultate. Zahvaljujući podjeli niza vježbi po težinama svaki korisnik može odabrati svoju razinu ovisno o fizičkoj spremi. Vježbe su također koncipirane i kako bi smanjile mogućnost ozljede zglobova koje su nažalost česte u sportu. Kao dodatna mogućnost aplikacije ugrađena je sklopka pomoću koje korisnik može odabrati tamni ili svijetli način rada aplikacije. To je kreirano iz razloga što za ljudsko oko nije dobro da boja ekrana nije usklađena s okolinom u kojoj se korisnik nalazi, tako da se preporuča koristiti tamni način rada u uvjetima s malo svjetla dakle najčešće po noći, a svijetli način rada bolje je koristiti u dnevnim uvjetima.

Sve navedene značajke aplikacije postignute su uporabom Andorid Studioa te Java programskog jezika. Govoreći o uporabljenim programskim rješenjima korištene su značajke poput RecyclerViewa koji omogućava učitavanje i uporabu velikih količina podataka na niskim memorijskim zahtjevima, ViewPager komponenta koja uz pomoć fragmenata omogućuje bolje korisničko sučelje zahvaljujući ugrađenim gestama za pomicanje zaslona, TabLayout komponenta koja je neizostavni dio većine aplikacija jer pomaže korisniku pri snalaženju tijekom korištenja aplikacije. Za dizajn aplikacije korišten je XML opisni jezik.

Svakako da i dalje postoji mogućnost da se aplikacija unaprijedi, jedna od mogućnosti bi bila ugradnja grafičkog prikaza napretka korisnika, na primjer pomoću grafikona. Osim toga moguća bi bila i ugradnja samostalnog mjerenja i unosa visine vertikalnog skoka pomoću mobilnog uređaja.

Kreiranje Android aplikacije za vertikalni skok je bilo dosta izazovno jer je bilo potrebno uključiti mnogo značajki koje Android Studio omogućuje. Bilo je potrebno temeljito istražiti Java programski jezik i njegovu primjenu u programiranju Android aplikacija. U radu su korištene i usluge Firebasea što je olakšalo rad s korisnicima i podacima. Cilj ove aplikacije je bio pojednostaviti prikaz korisniku pošto većina ostalih aplikacija ima mnogo mogućnosti koje su nepotrebne i slabo korištene, a samim time se teško snaći pri korištenju tih aplikacija.

LITERATURA

[1] Vertikalni skok [online], dostupno na: http://fitness.bluegym.hr/vertikalni-skok/ [pristupljeno 27. lipnja 2021.]

[2] Android Studio [online], dostupno na: https://en.wikipedia.org/wiki/Android_Studio

[pristupljeno 27. lipnja 2021.]

[3] Firebase Realtime Database [online], dostupno na: https://firebase.google.com/docs/database [pristupljeno 27. lipnja 2021.]

[4] What is Android? [online], dostupno na: https://www.tutorialspoint.com/android/android_overview.htm [pristupljeno 27.lipnja 2021.]

[5] Android architecture [online], dostupno na: https://www.tutlane.com/tutorial/android/android-architecture [pristupljeno 20. kolovoza 2021.]

[6] Android (operacijski sustav) [online], dostupno na: https://hr.wikipedia.org/wiki/Android_(operacijski_sustav) [pristupljeno 27. lipnja 2021.]

[7] Java (programski jezik) [online], dostupno na: https://hr.wikipedia.org/wiki/Java_(programski_jezik) [pristupljeno 27. lipnja 2021.]

[8] What is Java bytecode? [online], dostupno na: https://www.javatpoint.com/java-bytecode [pristupljeno 20. kolovoza 2021.]

[9] Create dynamic lists with RecyclerView [online], dostupno na: https://developer.android.com/guide/topics/ui/layout/recyclerview [pristupljeno 27. lipnja 2021.]

[10] Merlin materijal, laboratorijska vježba 7 iz kolegija Osnove razvoja web i mobilnih aplikacija, dostupno na: https://moodle.srce.hr/2020-2021/mod/folder/view.php?id=1535654 [pristupljeno 28.lipnja 2021.]

[11] TabLayout [online], dostupno na: https://developer.android.com/reference/com/google/android/material/tabs/TabLayout [pristupljeno 28. lipnja 2021.]

[12] XML [online], dostupno na: https://hr.wikipedia.org/wiki/XML [pristupljeno 28. lipnja 2021.]

[13] Adobe Photoshop [online], dostupno na: https://hr.wikipedia.org/wiki/Adobe_Photoshop [pristupljeno 28. lipnja 2021.]

SAŽETAK

U ovom radu napravljena je Android aplikacija za vertikalni skok. Glavne zadaće aplikacije bile su kreirati Android aplikaciju koja bi uključivala animacije vježbi raspoređenih po težini i uz to i mogućnost praćenja napretka te napraviti odgovarajući prikaz korisniku. Vježbe su prikazane videozapisima, a praćenje napretka se odvija tako da korisnik pomoću DatePicker komponente odabere željeni datum i unese domet svoga vertikalnog skoka. Za funkcionalni dio aplikacije korišten je Java programski jezik,a dizajn je kreiran pomoću XML opisnog jezika. Sama aplikacija izrađena je u programu Android Studio. Zbog potrebe praćenja napretka ugrađena je mogućnost registriranja korisnika pomoću Firebase autentifikacije. Ključni algoritmi korišteni u aplikaciji detaljno su pojašnjeni te je dan njihov isječak koda.

Ključne riječi: aplikacija, vertikalni skok, Android, praćenje napretka, vježbe

SUMMARY

Android application for vertical jump

In this paper, an Android application for vertical jump was created. The main tasks of the application were to create an Android application that would include animations of exercises arranged by difficulty and ability to track progress and make an appropriate display to the user. The exercises are shown in videos, and the progress is monitored by the user using the DatePicker component to select the desired date and enter the range of his vertical jump. The Java programming language was used for the functional part of the application, and the design was created using the XML descriptive language. The application itself was created in Android Studio. Due to the need of tracking progress, the ability to register users using Firebase authentication is built-in. The key algorithms used in the application are explained in detail and their code snippet is given.

Keywords: application, vertical jump, Android, progress tracking, exercises

ŽIVOTOPIS

Goran Klement rođen je 05.08.1998. godine u mjestu Slatina, Republika Hrvatska. Osnovnu školu pohađao je u Osnovnoj školi Eugena Kumičića. Nakon osnovne škole pohađao je opću gimnaziju u Slatini, koju završava 2017. godine. Iste godine upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, smjer računarstvo, koji trenutno pohađa.

PRILOZI

Na priloženom CD-u nalaze se:

- Završni rad u .docx i .pdf formatu
- Z1P obrazac
- Izjavu o originalnosti
- Ključni dijelovi koda