Android aplikacija za procjenu rezultata sistematskog pregleda

Fekete, Juraj

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

Stručni preddiplomski studij

ANDROID APLIKACIJA ZA PROCJENU REZULTATA SISTEMATSKOG PREGLEDA

Završni rad

Juraj Fekete

Osijek, 2019.

SADRŽAJ

1. UVOD

Android aplikacija za procjenu rezultata sistematskog pregleda pod nazivom HpCheck radi na principu mobilnog formulara koji uspoređuje unesene parametre s onima koji su propisani u Hrvatskom zdravstvu. Parametri su dobiveni od medicinskog laboratorija za medicinsku dijagnostiku u Osijeku. Glavna svrha aplikacije je da uvelike olakša i ubrza postupak sistematskog pregleda i to na način da preskoči odlazak obiteljskom liječniku i gubljenje vremena u čekaonicama tako da svaka osoba može jednostavno provjeriti svoje rezultate na mobitelu. Aplikacija služi samo za tumačenje rezultata sistematskog pregleda, a liječnik izvršava fizički pregled i prikuplja parametre za izvještaj. HpCheck je jednostavan program zato što se njegova glavna svrha može ostvariti kroz svega dva ili tri klika.

U drugom poglavlju su prikazana neka od sličnih rješenja koja se mogu pronaći na servisu *Google Play*. Ta rješenja spadaju u kategoriju „zdravlje“ kao i aplikacija ovog rada ali se bave proučavanjem i uspoređivanjem drugačijih parametara. U drugom poglavlju su detaljnije objašnjene funkcije i svrhe aplikacija uz pripadajuće isječke.

U trećem poglavlju je detaljnije objašnjen pristup vlastitom rješenju i ideja po kojoj su funkcije zamišljene i programirane. Bavi se parametrima, njihovim prikazom i dizajnom aplikacije.

U ovom poglavlju se detaljno prolazi kroz aktivnosti i funkcije aplikacije te se objašnjavaju uz pripadajuće odsječke kodova i slike na kojima se može vidjeti primjer za svaku od aktivnosti.

1.1. Zadatak rada

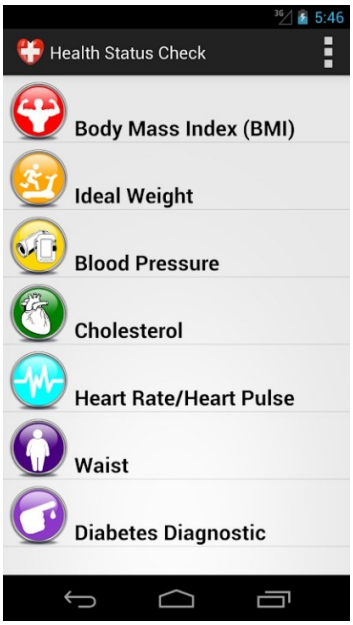
Zadatak ovog rada je pronaći najefikasnije rješenje kojim bi se omogućilo tumačenje sistematskih pregleda u što kraćem vremenu. Za pronalazak rješenja su korišteni parametri za sistematske preglede i napredno poznavanje programskog jezika Java. Zadatak je bio osmisliti funkcije koje bi korisnicima omogućile provjeru rezultata sistematskih pregleda i uvid u sve prethodne preglede uz uvjet da su prikazani samo njihovi pregledi a ne i pregledi drugih korisnika aplikacije. Zamišljeno je i da jedna od funkcija bude vodič kroz obližnje medicinske ustanove koje vrše sistematske preglede. Ideja tog vodiča je da bude karta koja pruža informacije o obližnjim ustanovama u odnosu na vašu trenutnu poziciju.

2. PREGLED POSTOJEĆIH RJEŠENJA

Već postoji puno napravljenih programa na temu zdravlja koji koriste različite metodologije i predstavljanje potrebnih rezultata. U nastavku će biti predstavljena dva programa, izdvojena zbog svoje jedinstvenosti i načina pristupa. Svaki program ima zajedničku karakteristiku, a to je da u konačnici korisnik dobiva informacije ili preporuku o trenutnom stanju zdravlja.

2.1. Health Status Check

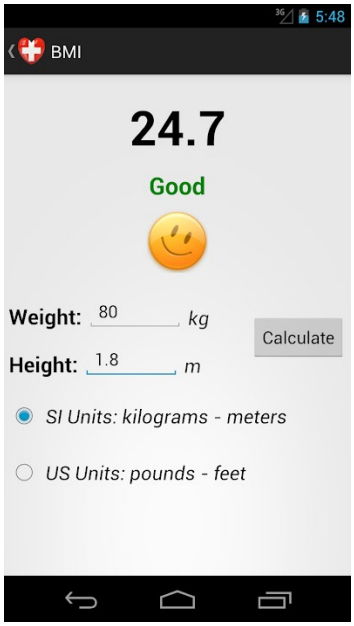
Health Status Check je aplikacija koja izračunava unesene vrijednosti i uspoređuje ih s unaprijed definiranim parametrima. Ova aplikacija se ponaša kao kalkulator i pomoću nje je moguće izračunati i provjeriti parametre za indeks tjelesne mase (engl. *Body Mass Index*), idealnu masu, tlak, razinu kolesterola, otkucaj srca, obujam struka i dijagnozu dijabetesa. Početni zaslon aplikacije je prikazan na slici 2.1.



[1]Sl. 2.1. *Prikaz početnog zaslona*

[1] Sl. 2.1. je preuzeta s https://play.google.com/store/apps/details?id=nnduy.healthcheck

Indeks tjelesne mase računa se na način da se unese korisnikova masa i visina te pritiskom na tipku izračunaj (engl. *Calculate*) aplikacija vrši izračun i po završetku ispisuje stanje uz pripadajuću sliku. Funkcija je prikazana na slici 2.2.



[2]**Sl. 2.2.** *Prikaz funkcije za računanje indeksa tjelesne mase*

Za računanje idealne mase potrebno je odabrati spol, tjelesni tip i unijeti visinu. Po završetku ispunjavanja formulara pritiskom na izračunaj dobiva se rezultat u kilogramima. Taj rezultat prikazuje korisnikovu idealnu težinu s obzirom na unesenu visinu, tjelesni tip i spol.

Računanje krvnog tlaka se izvršava nakon unosa razine sistoličkog i dijastoličkog tlaka. Nakon unesenih parametara i pritiska tipke izračunaj aplikacija ispisuje stanje tlaka koje ovisi o unesenim parametrima. Neki od mogućih rezultata su optimalan, normalan, visoko normalan i hipertenzija.

Razina kolesterola se izračunava unosom vrijednosti kolesterola i odabirom tipa kolesterola. Ponuđeni tipovi su ukupni kolesterol, LDL kolesterol i HDL kolesterol. Nakon što unosa kolesterola i tipa, pritiskom na tipku provjeri dobiva se izvještaj o stanju kolesterola koji može biti dobar, granica visokog rizika i visoki rizik.

Brzina otkucaja srca se izračunava na način da se unese brzina otkucaja srca u minuti i pritiskom na tipku provjeri dobiva se rezultat koji može biti sporo, normalno ili brzo.

[2] Sl. 2.2. je preuzeta s https://play.google.com/store/apps/details?id=nnduy.healthcheck

Za računanje obujma struka potrebno je širinu struka te odabrati spol. Aplikacija uzima unesene parametre i uspoređuje ih s onima koji su unaprijed definirani. Na temelju toga ima dva moguća rezultata, a to su: dobar ili rizik od pretilosti.

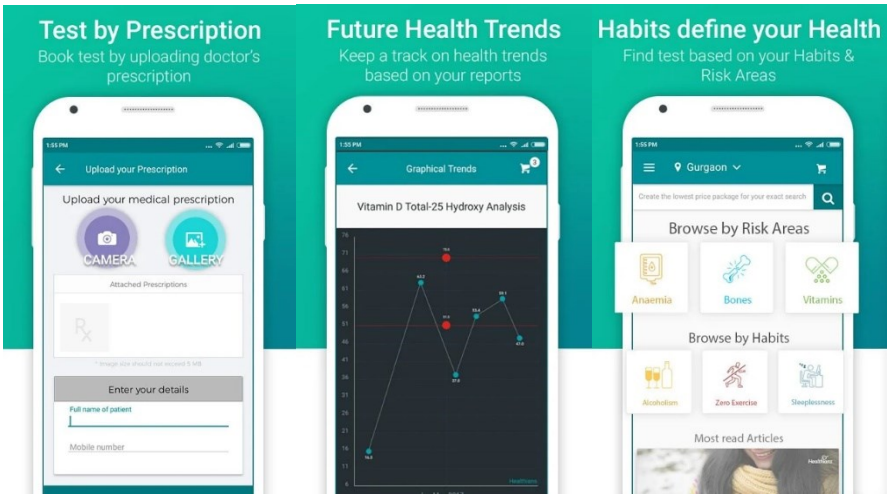
Za dijagnosticiranje dijabetesa odabire se tip testa i unosi se potreban parametar. Pritiskom na tipku dijagnoza (engl. *Diagnose*) dobiva se rezultat koji može biti: dobar ili dijabetes.

2.2. Healthians – Full Body Checkup & Blood Tests

Healthians je indijska služba za kućnu njegu i medicinu. Preko ove aplikacije nude svojim pretplaćenim korisnicima mogućnost zdravstvene pomoći i savjete koje bi drugdje dobili na drugačiji način. Aplikacija nudi liječničku podršku dok korisnik sjedi u svojoj kući ili dok je u pokretu. Sve se izvršava preko mobilnog uređaja.

Pomoću ove aplikacije ljudi mogu dobiti dijagnozu i potrebne savjete bez odlaska liječniku. Ukoliko je korisnička situacija ozbiljnija i inzistira na liječničkom pregledu, preko aplikacije dogovara se sastanak i liječnik dolazi u njegov dom te uzima sve potrebne uzorke i izvršava pregled.

Aplikacija je osmišljena da bude kao društvena mreža s korisnikovim doktorom i medicinskim timom. Na slici 2.3. su prikazane neke od funkcija aplikacije.



[3]Sl. 2.3. *Prikaz funkcija aplikacije Healthians*

[3] Sl. 2.3. je preuzeta s https://play.google.com/store/apps/details?id=com.healthians.main.healthians&hl=en

3. RAZVOJ VLASTITOG RJEŠENJA

Nakon analize drugih programa, te procjene njihovih metoda i relevantnosti rješenja, potrebno je po uzoru na ista osmisliti vlastito rješenje. Svi programi su zasnovani na pružanju povratnih informacija korisnicima koje se odnose na njihovo stanje zdravlja. Kako bi program mogao pružiti povratnu informaciju, treba usporediti unesene vrijednosti s unaprijed definiranim parametrima koji su kasnije prikazani.

3.1. Parametri

Parametri dobiveni iz laboratorija za medicinsku dijagnostiku su podijeljeni u nekoliko skupina. U ovoj aplikaciji se vrši provjera za osnovni tip sistematskog pregleda, što znači da se uzimaju samo osnovne mjere.

Na izvještaju sistematskog pregleda mjere su podijeljene u dvije grupe: hematologija i biokemija. U nastavku su prikazane slike na kojima se mogu vidjeti parametri i njihove vrijednosti u odnosu na spol i dob. Na slici 3.1. vidi se prikaz parametara za hematologiju dok je na slici 3.2. prikazana tablica parametara za biokemiju.

Analit/pretraga/indeks	Jedinica	Spol	Dob	Interval
Sedimentacija eritorcita	mm/3,6 ks	muški	20-50 g.	2 - 13
			>50 g.	3 - 23
		ženski	20 - 50 g.	4 - 24
			>50	5 - 28
Leukociti	x 10^9/L	muški	≥20 g.	3,4 - 9,7
		ženski	≥20 g	3,4 - 9,7
Eritrociti	x10^{12}/L	muški	≥20 g	4,34 - 5,72
		ženski	≥20 g	3,86 - 5,08
Hemoglobin	g/L	muški	≥20 g.	138 - 175
		ženski	≥20 g.	119 - 157
Hematokrit	L/L	muški	≥20 g.	0,415 - 0,530
		ženski	≥20 g.	0,356 - 0,460
MCH	pg	muški	≥20 g.	27,4 - 33,9
		ženski	≥20 g.	27,4 - 33,9
MCHC	g/L	muški	≥20 g.	320 - 345
		ženski	≥20 g.	320 - 345
MCV	f L	muški	≥20 g.	83 - 97,2
		ženski	≥20 g.	83 - 97,2
Trombociti	x10^9/L	muški	≥20 g.	158 - 424
		ženski	≥20 g.	158 - 424
RDW	%	muški	≥20 g.	9 - 15
		ženski	≥20 g.	9 - 15
Prosječni volumen trombocita	f L	muški	≥20 g.	6,8 - 10,4
		ženski	≥20 g.	6,8 - 10,4

Sl. 3.1. *Prikaz parametara hematologije*

Analit/pretraga/indeks	Jedinica	Spol	Dob	Interval
				kapilarna krv
Glukoza	mmol/L	muško,žensko	20 - 30 g.	3,3 - 5,2
		muško,žensko	>30 g.	3,5 - 5,6
				serum
		muško,žensko	20 - 30 g.	4,2 - 6
		muško,žensko	>30 g.	4,4 - 6,4
Urea	mmol/L	muško,žensko	≥20	2,8 - 8,3
Kreatinin	µmol/L	muški	≥20	79 - 125
		ženski	≥20	63 - 107
AST	U/L	muški	≥20	11 - 38
		ženski	≥20	8 - 30
ALT	U/L	muški	≥20	12 - 48
		ženski	≥20	10 - 36
GGT	U/L	muški	≥20	11 - 55
		ženski	≥20	9 - 35

Sl. 3.2. *Prikaz parametara biokemije*

3.2. Dizajn

Aplikacija koristi jednostavan dizajn te se sastoji od ukupno četiri aktivnosti (engl. *Activity*). Neke aktivnosti su sastavljene od fragmenata koji služe Za intuitivnije snalaženje korisnika tijekom korištenja aplikacije, estetsko poboljšanje i kako bi olakšali korištenje funkcija. Fragmenti su prikazani kao model straničnog pogleda (engl. *ViewPager*) koji u nekim dijelovima aplikacije ostavlja dojam listanja stranica knjige. Ta funkcija olakšava korištenje programa zato što je široko rasprostranjena te se koristi u većini današnjih modernih aplikacija i programa.

Implementacija fragmenata i straničnog pogleda te njihov poziv je prikazan u slijedećem odsječku koda 3.1. i 3.2. koji se nalazi u prilogu P.3.1.

```
     public class LoginFragment extends Fragment implements
View.OnClickListener {

     …

public LoginFragment() {
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        view = inflater.inflate(R.layout.login_fragment, container,
false);
     …

Login.setOnClickListener(this);

        return view;
    }
```

Odsječak programskog koda 3.1. *LoginFragment.java*

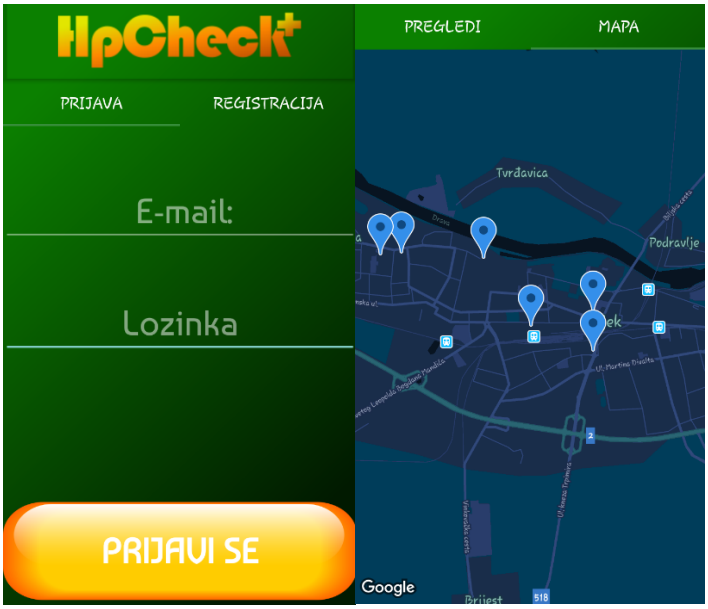
```
public class MainActivity extends AppCompatActivity {

…

ViewPagerAdapter adapter = new
ViewPagerAdapter(getSupportFragmentManager());
        adapter.AddFragment(new LoginFragment(), "Prijava");
        adapter.AddFragment(new RegisterFragment(),
"Registracija");
        viewPager.setAdapter(adapter);
        tablayout.setupWithViewPager(viewPager);
    }
```

Odsječak programskog koda 3.2. *MainActivity.java*

Dizajnerske komponente kao što su logo, pozadinska slika, pozadine tipki i ikone se nalaze u mapi *res* iz P.3.1. i putanja do njih je *hpcheck/app/src/main/res/drawable*. U aplikaciji se nalazi funkcija koja koristi usluge Google mapa i za nju je implementirana posebna pozadina koja je preuzeta s Google servisa, a putanja do datoteke u P.3.1. je *hpcheck/app/src/main/res/raw.* Slika 3.3. prikazuje izgled implementiranih pozadina, ikona i mape.



Sl. 3.3. *Prikaz implementacije dizajnerskih značajki*

U nastavku je prikazan odsječak koda 3.3 , 3.4. i 3.5. koji se odnosi na prethodnu sliku. Sav kod se nalazi u *layout* mapi P.3.1. a putanja do njega je *hpcheck/app/src/main/res/layout*.

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


<com.google.android.gms.maps.MapView
android:id="@+id/mapView"
android:layout_width="match_parent"
android:layout_height="match_parent">


</com.google.android.gms.maps.MapView>



</RelativeLayout>
```

Odsječak programskog koda 3.3. *map_fragment.xml*

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

    <android.support.design.widget.AppBarLayout
        android:id="@+id/appbarid"
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:background="@color/transparent"
        android:elevation="0dp"
        android:gravity="center">

        <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="10dp"
            android:src="@drawable/logo" />
    </android.support.design.widget.AppBarLayout>

    <android.support.design.widget.TabLayout
        android:id="@+id/tablayout_id"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/transparent"
        app:tabGravity="fill"
        app:tabIndicatorColor="@color/md_green_700"
        app:tabMode="fixed"

app:tabTextColor="@color/md_white_1000"></android.support.design.widget.
TabLayout>

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager_id"
        android:layout_width="match_parent"

android:layout_height="match_parent"></android.support.v4.view.ViewPager
>

</LinearLayout>
```

Odsječak programskog koda 3.4. *activity_main.xml*

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

    <android.support.design.widget.TextInputEditText
        android:id="@+id/etEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:fontFamily="@font/baumans"
        android:hint=" E-mail:"
        android:inputType="text"
        android:textColor="@color/md_white_1000"
        android:textAlignment="center"
        android:textSize="30sp" />

    <android.support.design.widget.TextInputEditText
        android:id="@+id/etPassword"
        android:textColor="@color/md_white_1000"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:fontFamily="@font/baumans"
        android:hint="Lozinka"
        android:inputType="textPassword"
        android:textAlignment="center"
        android:textSize="30sp" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

        <Button
            android:id="@+id/bLogin"
            android:layout_marginTop="150dp"
            android:layout_width="match_parent"
            android:layout_height="100dp"

            android:background="@drawable/button"
            android:fontFamily="@font/baumans"
            android:text="Prijavi se"
            android:textColor="@color/md_white_1000"
            android:textSize="30sp"
            android:textStyle="bold" />
</LinearLayout>
</LinearLayout>
```

Odsječak programskog koda 3.5. *login_fragment.xml*

Boje koje su korištene za kreiranje pozadine, fontove i ikone se nalaze u posebnoj datoteci u P.3.1. a putanja do njih je *hpheck/app/src/main/res/values/colors.xml*.

4. STRUKTURA APLIKACIJE

Aplikacije se pokreće preko virtualnih emulatora na korisnikovom računalu ili pomoću mobilnih uređaja koji koriste Android operativni sustav. U ovom poglavlju vide se funkcije i kod aplikacije. Na slici 4.1. je prikazan dijagram toka aplikacije kroz program i moguće ishode u odnosu na odabrane akcije.

Sl. 4.1. *Prikaz dijagrama toka aplikacije*

4.1. Početni izbornik

Prilikom pokretanja aplikacije se otvara početni zaslon koji je definiran u *MainActivity* sa svojim pripadajućim rasporedom (engl. *Layout*). *MainActivity* je sastavljen od dva fragmenta koji su prikazanu u obliku *ViewPager*-a. Fragmenti koji su prikazani su: *Prijava* i *Registracija*.

Prijava je prva prikazana i sastoji se od dva polja za unos teksta i jedne tipke za potvrdu. Korisnik unosi svoj e-pošta i zaporka te pritiskom na tipku „Prijavi se“ dolazi do provjere podataka i usporedbe s podatcima na servisu Firebase. Ukoliko uneseni podatci ne postoje u bazi korisnika, aplikacija će prikazati jedno od povratnih poruka s opisom problema. Fragment za prijave je prikazan u slijedećem odsječku koda 4.1.a) i 4.1.b).

```
public class LoginFragment extends Fragment implements
View.OnClickListener {
    private TextInputEditText Email, Password;
    private Button Login;
    private FirebaseAuth auth;
    View;
    public LoginFragment() {
    }
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        view = inflater.inflate(R.layout.login_fragment, container,
false);
        auth = FirebaseAuth.getInstance();
        Email = (TextInputEditText) view.findViewById(R.id.etEmail);
        Password = (TextInputEditText)
view.findViewById(R.id.etPassword);
        Login = (Button) view.findViewById(R.id.bLogin);
        Login.setOnClickListener(this);
        return view;
    }
```

Odsječak programskog koda 4.1. a) *LoginFragment.java*

```
@Override
    public void onClick(View view) {
        String email = Email.getText().toString().trim();
        final String password = Password.getText().toString().trim();
        if (TextUtils.isEmpty(email)) {
            Email.setError("Unesite e-mail adresu");
            return;
        }
        if (TextUtils.isEmpty(password)) {
            Password.setError("Unesite lozinku!");
            return;
        }
  auth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(getActivity(), new
OnCompleteListener<AuthResult>() {
                    …
    @Override
    public void onClick(View view) {
        String email = Email.getText().toString().trim();
        final String password = Password.getText().toString().trim();
        if (TextUtils.isEmpty(email)) {
            Email.setError("Unesite e-mail adresu");
            return;}
        if (TextUtils.isEmpty(password)) {
            Password.setError("Unesite lozinku!");
            return;}
        auth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(getActivity(), new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
task) {
                        if (!task.isSuccessful()) {
                            // there was an error
                            if (password.length() < 6) {
                                Password.setError("Kriva lozinka, molim
unesite točnu lozinku");
                            } else {
                                Toast.makeText(getActivity(), "Login
failed", Toast.LENGTH_LONG).show();
                            }} else {
                            Intent = new Intent(getActivity(),
ContentActivity.class);
                            startActivity(intent);
                            getActivity().finish();
                        }}});}}
```

Odsječak programskog koda 4.1. b) *LoginFragment.java*

Ukoliko korisnik ne posjeduje račun koji je povezan s ovom aplikacijom, a želi koristiti funkcije ove aplikacije potrebno je registrirati se unutar aplikacije. Funkcija registracije je definirana u fragmentu *RegisterFragmen*“ koji se nalazi u *MainAcitivity*. Registracija se sastoji od dva polja za popunjavanje i tipke za potvrdu. U polja se stavljaju e-pošta i zaporka koju smo

odabrali za ovaj račun. Potrebno je odabrati zaporku koja se sastoji od minimalno šest znakova. Cijeli kod za funkcije fragment registracije se nalazi u odsječku za kod 4.3. *RegisterFragment.java*, a kod za dizajn fragmenta se nalazi u odsječku koda 4.2. register_fragment.

```
<LinearLayout android:layout_height="match_parent"
android:layout_width="match_parent"
android:orientation="vertical"
android:gravity="center"
android:background="@color/transparent"
xmlns:android="http://schemas.android.com/apk/res/android">
<android.support.design.widget.TextInputEditText
android:id="@+id/tiEmail"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="50dp"
android:fontFamily="@font/baumans"
android:hint=" E-mail:"
android:inputType="text"
android:textColor="@color/md_white_1000"
android:textAlignment="center"
android:textSize="30sp" />
<android.support.design.widget.TextInputEditText
android:id="@+id/tiPassword"
android:textColor="@color/md_white_1000"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_marginTop="50dp"
android:fontFamily="@font/baumans"
android:hint="Lozinka"
android:inputType="textPassword"
android:textAlignment="center"
android:textSize="30sp" />
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content">
<Button
android:id="@+id/bSubmit"
android:layout_marginTop="150dp"
android:layout_width="match_parent"
android:layout_height="100dp"
android:background="@drawable/button"
android:fontFamily="@font/baumans"
android:text="Registriraj se"
android:textColor="@color/md_white_1000"
android:textSize="30sp"
android:textStyle="bold" />
</LinearLayout>
</LinearLayout>
```

Odsječak programskog koda 4.2. *register_fragment.xml*

```
public class RegisterFragment extends Fragment implements
View.OnClickListener {
    private TextInputEditText Email, Password;
    private Button Submit;
    private FirebaseAuth auth;
    private DatabaseReference databaseReference;
    View;
    public RegisterFragment() {
    }
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        databaseReference =
FirebaseDatabase.getInstance().getReference();
        view = inflater.inflate(R.layout.register_fragment, container,
false);
        Email = (TextInputEditText) view.findViewById(R.id.tiEmail);
        Password = (TextInputEditText)
view.findViewById(R.id.tiPassword);
        Submit = (Button) view.findViewById(R.id.bSubmit);
        Submit.setOnClickListener(this);
        auth = FirebaseAuth.getInstance();
        return view;
    }
@Override
    public void onClick(View view) {
        String email = Email.getText().toString().trim();
        String password = Password.getText().toString().trim();
        if (TextUtils.isEmpty(email)) {
            Email.setError("Unesite e-mail");
            return;
        }
        if (TextUtils.isEmpty(password)) {
            Password.setError("Unesite lozinku");
            return;
        }
        if (password.length() < 6) {
            Password.setError("Lozinka prekratka, unesite minimalno 6
znakova");
            return;
        }
auth.createUserWithEmailAndPassword(email, password)
                .addOnCompleteListener(getActivity(), new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
task) {
                        Toast.makeText(getActivity(), "Registracija
uspješna", Toast.LENGTH_LONG).show();
                        if (!task.isSuccessful()) {
                            Toast.makeText(getActivity(),
"Authentication failed." + task.getException(),
                                    Toast.LENGTH_LONG).show();
                        } else {
                            onAuthSuccess(task.getResult().getUser());
                        }}});}
```

```
private void onAuthSuccess(FirebaseUser user) {
        String username = usernameFromEmail(user.getEmail());
        writeNewUser(user.getUid(), username, user.getEmail());
        Toast.makeText(getActivity(), "Registration complete",
Toast.LENGTH_LONG).show();
    }
    private String usernameFromEmail(String email) {
        if (email.contains("@")) {
            return email.split("@")[0];
        } else {
            return email;
        }
    }
    private void writeNewUser(String userId, String name, String email)
{
        User user = new User(email);
        databaseReference.child("users").child(userId).setValue(user);
    }
}
```

Odsječak programskog koda 4.3. *RegisterFragment.java*

Pritiskom tipke „Registriraj se“ prikupljaju se elementi koje je korisnik unio u oba polja te se pohranjuju na Firebase servis u bazu podataka za autentikaciju. Po završetku procesa će aplikacija ispisati poruku u obliku *Toast* funkcije koja može biti „Registracija uspješna“ ili „Authentication failed“ ovisno o ishodu procesa.

Nakon uspješne registracije korisnik može koristiti svoje podatke kod prijave. Nakon unosa svih elemenata kod prijave izvršava se provjera i usporedba podataka s onima na Firebase servisu te ako podatci postoje prijava će biti uspješno završena i otvara se slijedeća aktivnost, a to je lista pregleda.

4.2. Lista pregleda

Nakon uspješne prijave pokreće se aktivnost *ContentActivity* koja je sastavljena od dva fragmenta a to su: *Pregledi* i *Mapa*. Prikaz koda koji se nalazi u P.3.1. vidi se u odsječku koda 4.4. *ContentActivity.java* a dizajn u odsječku koda 4.5.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:background="@drawable/background"
    android:layout_height="match_parent"
    android:layout_width="match_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <android.support.design.widget.TabLayout
        android:id="@+id/tablayout_id_content"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/transparent"
        app:tabGravity="fill"
        app:tabIndicatorColor="@color/md_green_700"
        app:tabMode="fixed"
        app:tabTextColor="@color/md_white_1000">
    </android.support.design.widget.TabLayout>
    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager_id_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent">
    </android.support.v4.view.ViewPager>
</LinearLayout>
```

Odsječak programskog koda 4.4. *content_activity.xml.*

```
public class ContentActivity extends AppCompatActivity {
    @BindView(R.id.tablayout_id_content)
    TabLayout tablayout_content;
    @BindView(R.id.viewpager_id_content)
    ViewPager viewPager_content;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.content_activity);
        ButterKnife.bind(this);


        ViewPagerAdapter adapter = new
ViewPagerAdapter(getSupportFragmentManager());
        adapter.AddFragment(new ListFragment(), "Pregledi");
        adapter.AddFragment(new MapsFragment(), "Mapa");
        viewPager_content.setAdapter(adapter);
        tablayout_content.setupWithViewPager(viewPager_content);
    }}
```

Odsječak programskog koda 4.5. *ContentActivity.java*

Prvo su prikazani podatci o pregledima koji su poredani u listi. Svaki element liste ima sažet opis o tom pregledu. Podatci o pregledu koji su prikazani na listi su: ime, datum pregleda, spol i dob. Pritiskom na element liste se pruža prošireni opis pregleda. Izgled *Activity*-a je prikazan na slici 4.2.

Sl. 4.2. *Liste pregleda*

U nastavku je prikazan kod 4.6. funkcije za pokretanje proširenog opisa.

```
@Override
                    public void onClick(View view) {
                        Intent intent = new
Intent(ListFragment.this.getContext(), ResultsActivity.class);
                        intent.putExtra("CheckResult", model);
                        startActivity(intent);
                    }
```

Odsječak programskog koda 4.6. *pokretanje proširenog opisa*

Na dnu aplikacije u desnom kutu se nalazi tipka kojom se započinje novi pregled, a funkcija je prikazana u kodu 4.7.

```
  StartCheck = (FloatingActionButton) view.findViewById(R.id.fabStart);
        StartCheck.setOnClickListener(new View.OnClickListener() {
  @Override
            public void onClick(View view) {
                startActivity(new Intent(getActivity(),
CheckActivity.class));
            }
```

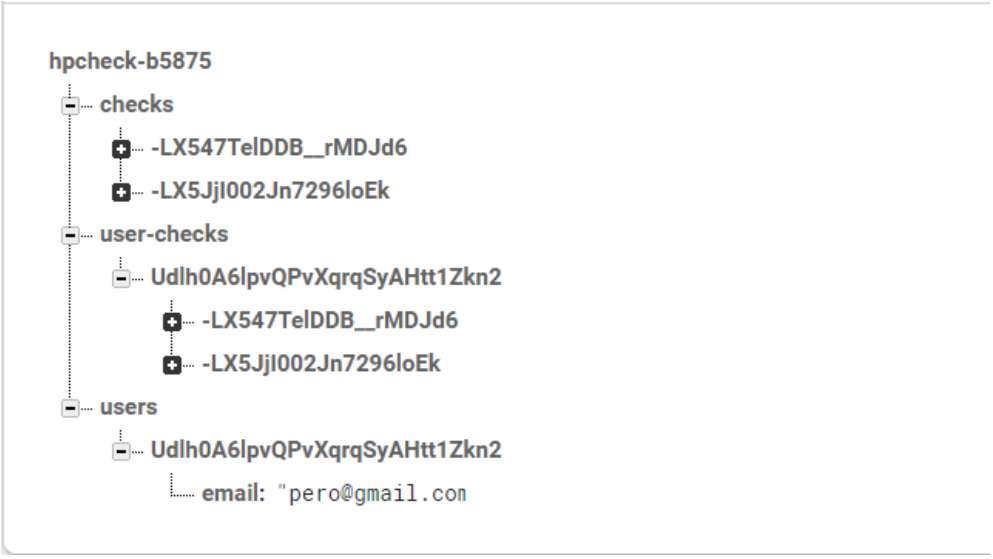
Odsječak programskog koda 4.7. *pokretanje novog pregleda*

U zaglavlju fragmenta se nalazi tipka koja omogućuje odjavu korisničkog računa i povratak na početnu aktivnost i prikazana je u odsječku koda 4.8.

```
  LogOut = (Button) view.findViewById(R.id.bLogOut);
        LogOut.setOnClickListener(new View.OnClickListener() {
                                        @Override
                                        public void onClick(View view) {
                                    FirebaseAuth.getInstance().signOut();
                                            startActivity(new
Intent(getActivity(), MainActivity.class));
                                        }
                                    }
```

Odsječak programskog koda 4.8. *funkcija odjave korisnika*

Podatci o pregledima koji se popunjavaju u elemente liste se prikupljaju s Firebase servisa i njihovo pozivanje je prikazano odsječkom koda 4.9 . Izgled podataka svakog pregleda koji se nalaze na Firebase servisu su prikazani slikom 4.3.



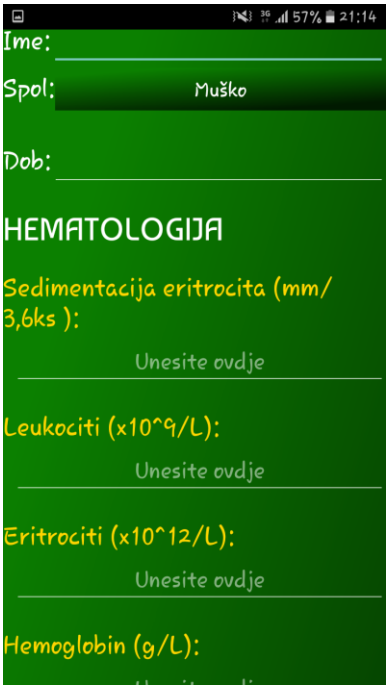
Sl. 4.3. *Liste pregleda u servisu Firebase*

```
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable
ViewGroup container, @Nullable Bundle savedInstanceState) {
        view = inflater.inflate(R.layout.list_fragment, container,
false);
…
        recyclerView = view.findViewById(R.id.rvList);
        databaseReference =
FirebaseDatabase.getInstance().getReference();
        setupUI();
        rvList = view.findViewById(R.id.rvList);
        return view;
    }
    private void setupUI() {
        mManager = new LinearLayoutManager(this.getContext());
        mManager.setReverseLayout(true);
        mManager.setStackFromEnd(true);
        recyclerView.setLayoutManager(mManager);
        Query tasksQuery = getQuery();
        FirebaseRecyclerOptions options = new
FirebaseRecyclerOptions.Builder<Check>()
                .setQuery(tasksQuery, Check.class)
                .build();
        checksAdapter = new FirebaseRecyclerAdapter<Check,
CheckViewholder>(options) {
            @Override
            public CheckViewholder onCreateViewHolder(ViewGroup
viewGroup, int i) {
                LayoutInflater inflater =
LayoutInflater.from(viewGroup.getContext());
                return new
CheckViewholder(inflater.inflate(R.layout.item_check_list, viewGroup,
false));}};
        recyclerView.setAdapter(checksAdapter);}
    @Override
    public void onStart() {
        super.onStart();
        if (checksAdapter != null) {
            checksAdapter.startListening();}}
    @Override
    public void onStop() {
        super.onStop();
        if (checksAdapter != null) {
            checksAdapter.stopListening();}}
    public Query getQuery() {
        return databaseReference.child("user-checks")
                .child(getUid());
    }
```

Odsječak programskog koda 4.9. *prikupljanje podataka s Firebase servisa*

4.3. Pregled

U prethodnom poglavlju je spomenuta mogućnost pokretanja pregleda pomoću tipke čija je funkcija prikazana u odsječku koda 4.7. Pritiskom tipke se otvara nova aktivnost pod nazivom *CheckActivity* u kojoj korisnik unosi svoje podatke i parametre koji su potrebni za obavljanje provjere sistematskog pregleda. Izgled aktivnosti je prikazana slikom 4.3.



Sl. 4.3. *Prikaz aktivnosti za unos podataka*

Aplikacija je namijenjena za sistematske preglede odraslih osoba pa je potrebno da korisnik kod unosa svoje dobi stavi broj veći od dvadeset. Ukoliko taj uvjet nije zadovoljen aplikacija će poslati jednu od povratnih informacija koja su prikazane u odsječku koda 4.10.

```
{
…
String dob = etAge.getText().toString();
        if (TextUtils.isEmpty(dob)){etAge.setError("Unos
obavezan");return;}
        if (Integer.valueOf(dob)<20){ etAge.setError("Osoba mora biti
starija od 20 godina");return;
}
```

Odsječak programskog koda 4.10. *Provjera korisnikove dobi*

Na dnu aktivnosti se nalazi tipka *Potvrdi* koja prikuplja sve parametre i podatke te ih sprema i razvrstava na *Firebase* servis. Ti podatci se kasnije prikazuju na listi pregleda koja je objašnjena u prethodnom poglavlju. Datum pregleda se ne unosi već po pritisku tipke *Potvrdi* aplikacija automatski prikupi parametre i trenutni datum te ih pohranjuje u bazu s ostalim podatcima. Prikupljanje podataka i spremanje u bazu je prikazano slijedećim odsječkom koda 4.11.

```
@Override
    public void onClick(View view) {
        final String userId = getUid();
        setEditingEnabled(true);
        Date currentTime = Calendar.getInstance().getTime();
        SimpleDateFormat dateFormat = new
SimpleDateFormat("dd.MM.yyyy");
        String datum = dateFormat.format(currentTime);
        String personName = etPersonName.getText().toString();
        String spol = spGender.getSelectedItem().toString();
        String dob = etAge.getText().toString();
        if (TextUtils.isEmpty(dob)){etAge.setError("Unos
obavezan");return;}
        if (Integer.valueOf(dob)<20){ etAge.setError("Osoba mora biti
starija od 20 godina");return; }
        String sedimantacijaEritrocita =
etSedimentcaijaErit.getText().toString();if
(TextUtils.isEmpty(sedimantacijaEritrocita)){etSedimentcaijaErit.setErro
r("Unos obavezan");return;}
        String leukociti = etLeukociti.getText().toString();if
(TextUtils.isEmpty(leukociti)){etLeukociti.setError("Unos
obavezan");return;}
   …
databaseReference.child("users").child(userId).addListenerForSingleValue
Event(
 new ValueEventListener() {
                    @Override
 public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                        User user = dataSnapshot.getValue(User.class);
                        if (user == null) {
                            Toast.makeText(CheckActivity.this,
                                    "Error: could not fetch user.",
                                    Toast.LENGTH_SHORT).show();
                        } else {
                            writeNewCheck(userId, user, check);
                            CheckActivity.this.finish();}}
@Override
                    public void onCancelled(@NonNull DatabaseError
databaseError) {
                        setEditingEnabled(true);
                    }});}
NonNull DataSnapshot dataSnapshot) {
                        User user = dataSnapshot.getValue(User.class);
                        if (user == null) {
                            Toast.makeText(CheckActivity.this,
                                    "Error: could not fetch user.",
                                    Toast.LENGTH_SHORT).show();}
else {
                            writeNewCheck(userId, user, check);
                            CheckActivity.this.finish();
                        }}
                    @Override
                    public void onCancelled(@NonNull DatabaseError
databaseError) {
                        setEditingEnabled(true);
                    }});}
```

Odsječak programskog koda 4.11. *Prikupljanje parametara i spremanje u bazu podataka*

4.4. Rezultati

Prikaz rezultata se dobiva pritiskom na element liste koja je objašnjena u poglavlju 4.2. Izgled aktivnosti rezultata je prikazana slikom 4.4.

Sl. 4.4. *Prikaz aktivnosti za unos podataka*

Rezultati su prikazani kao lista koja sadrži imena parametra i vrijednosti koje je korisnik unio prilikom aktivnosti pregleda. Iz slike 4.3. se vide dvije različite ikone koje se pojavljuju pokraj vrijednosti pojedinih parametara. Zelena ikona pokazuje da vrijednost unesenog parametra zadovoljava onu koju je propisao laboratorij za medicinsku dijagnostiku dok crvena znači da je vrijednost previsoka ili preniska. U nastavku je prikazan odsječak koda 4.12. koji opisuje prikaz parametara u aktivnosti i dodjeljivanje ikona.

```
private void setupUI(Check check) {
  tvSedimentacijaEritrocita.setText(check.getSedimantacijaEritrocita());
        if (check.isSedimentacijaEritrocitaOk()) {
        ivSedimentacijaEritrocita.setImageResource(R.drawable.positive);
        } else {
        ivSedimentacijaEritrocita.setImageResource(R.drawable.negative);
        }
        tvLeukociti.setText(check.getLeukociti());
        if (check.isLeukocitiOk()){
            ivLeukociti.setImageResource(R.drawable.positive);
        }else {
            ivLeukociti.setImageResource(R.drawable.negative);
        }
...   }
```

Odsječak programskog koda 4.12. *Prikazivanje rezultata pregleda*

Uz rezultate pregleda, prethodni kod prikazuje i funkcije koje vrše uspoređivanje unesenih parametara s onima koji su propisani od strane medicinskog laboratorija. Funkcije su: isSedimentacijaEritrocitaOk i isLeukocitiOk. Slijedeći odsječak koda 4.13. opisuje funkciju isSedimentacijaEritrocitaOk koja vraća rezultat u obliku TRUE ili FALSE. Taj rezultat ovisi o tome da li je uneseni parametar unutar propisanih granica s obzirom na dob i spol korisnika.

```
public Boolean isSedimentacijaEritrocitaOk() {
        Integer dob = Integer.valueOf(this.Dob);
        Integer sedimentacijaEritrocita =
Integer.valueOf(this.getSedimantacijaEritrocita());
        if (this.getSpol().equals("Muško")) {
            if (dob <= 50) {
                if (
                        sedimentacijaEritrocita >= 2
                        && sedimentacijaEritrocita <= 13
                ) {
                    return Boolean.TRUE;
                } else {
                    return Boolean.FALSE;
                }
            } else {
                if (
                        sedimentacijaEritrocita >= 3
                        && sedimentacijaEritrocita <= 23
                ) {
                    return Boolean.TRUE;
                } else {
                    return Boolean.FALSE;
                }
            }
        } else if(this.getSpol().equals("Žensko")) {
            if (dob <= 50) {
                if (
                        sedimentacijaEritrocita >= 4
                        && sedimentacijaEritrocita <= 24
                ) {
                    return Boolean.TRUE;
                } else {
                    return Boolean.FALSE;
                }
            } else {
                if (
                        sedimentacijaEritrocita >= 5
                        && sedimentacijaEritrocita <= 28
                ) {
                    return Boolean.TRUE;
                } else {
                    return Boolean.FALSE;
                }}
        } else {

            return Boolean.FALSE;
       }}
```

Odsječak programskog koda 4.13. *Uspoređivanje parametara*

5. ZAKLJUČAK

U prethodnim poglavljima su prikazani odsječci kodova i objašnjenja za sve funkcije aplikacije te dokaz funkcionalnosti kroz slike za svaku aktivnost. Rad je temeljen na brojnim idejama i aplikacijama koje se nalaze na servisu *Google Play* ali razlika je ta što se koriste parametri hrvatskog zdravstva i prikaz ustanova koje obavljaju sistematske preglede obuhvaća samo područje grada Osijeka. Postavljen je pristup programiranju i dizajnu koji bi rezultirao jednostavnom aplikacijom koja omogućava korisniku brži pregled u što manje klikova i vremena. Za izradu su korištene neke od postojećih rješenja koje se mogu pronaći na internetu a omogućavaju jednostavan pristup potrebnim informacija u aplikaciji. Za funkcionalnost pregleda i uspješno uspoređivanje parametara su korištene osnovne matematičke funkcije kao što su *veće od, veće ili jednako od* i *manje od* koje se nalaze u *if* bloku.

Postoje brojni koraci koji su ovu aplikaciju mogli učiniti boljom. Prilikom izvođenja sistematskog pregleda neki od parametara variraju pa bi se to trebalo uzeti u obzir te u aplikaciji postaviti dozvoljena prekoračenja za parametre koji su kritični. Ova aplikacija pruža pregled osnovnog tipa sistematskog pregleda i tu bi se moglo proširiti na napredniji tip koji bi obuhvaćao veći broj parametara. Mapa koja pruža informacije o ustanovama koje provode sistematske preglede bi se mogla proširiti da obuvača veće područje. Mogao bi se implementirati bolji i ljepši dizajn kao i popravci koji bi učinili aplikaciju optimiziranijom.

LITERATURA

[1] Google dokumentacija, s interneta, https://developer.android.com/guide

[2] Google dokumentacija, s interneta, https://developer.android.com/guide/components/activities/intro-activities

[3] Google dokumentacija, s interneta, https://developer.android.com/guide/components/fragments

[4] Google dokumentacija, s interneta, https://developer.android.com/reference/java/util/Map

[5] Youtube, s interneta, https://www.youtube.com/watch?v=WgXA697mAPA

[6] Youtube, s interneta, https://www.youtube.com/watch?v=fGtFfCKONhQ

[7] Youtube, s interneta, https://www.youtube.com/watch?v=ACK67xU1Y3s

[8] AndroidTM Application Development For Dummies, Published by *Wiley Publishing Inc*. 111 River Street Hoboken.

[9] Busy Coder's Guide to Android Development, Author: Mark L. Murphy, Originally published: 2009.

[10] AndroidHive-https://www.androidhive.info/2016/06/android-getting-started-firebase-simple-login-registration-auth/

[11] Klinička bolnica Merkur, Zavod za kliničku kemiju, Harmonizacija laboratorijskih nalaza u području opće medicinske biokemije, Zagreb, 2004. godine.

PRILOZI

P3.1. – Kod i programski podatci potrebni za izvršavanje aplikacije, nalazi se u digitalnom formatu priloženom u mapi HpCheck.

SAŽETAK

Tema rada je dizajn i razvoj funkcionalne mobilne aplikacije koja tumači rezultate sistematskog pregleda. Predstavljena su najsličnija rješenja te je definirano razvojno okruženje u kojem se razvija vlastito. Za razvoj programa se koristi Java programski jezik i Android studio koji se uz programiranje koristi i kao virtualni alat za testiranje aplikacije. Aplikacija koristi Firebase servis za spremanje i dohvaćanje podataka koji su potrebni za funkcioniranje određenih značajki. U sebi ima i funkciju autentikacije koja omogućava da svaki korisnik ima pristup samo svojim pregledima. Ukoliko korisnik ne posjeduje korisnički račun može se registrirati unutar same aplikacije. Prijavljeni korisnik ima i mogućnost obavljanja pregleda koji se po završetku sprema u bazu i kasnije prikazuje na listi pregleda. Dobiveni parametri su prilagođeni za preglede određenih dobnih skupina tako da korisnici mlađi od dvadeset godina nisu u mogućnosti koristiti funkciju pregleda. Pritiskom na jedan od pregleda s liste se pokreće nova aktivnost koja nudi opširnije informacije o tom pregledu uz ikone koje predstavljaju status za svaki parametar. Još jedna od funkcija aplikacije je prikaz mape obližnjih medicinskih ustanova koje obavljaju sistematske preglede.

Ključne riječi: Android, Java, Firebase, sistematski pregled, autentikacija, mapa

ABSTRACT

ANDROID APPLICATION FOR EVALUATION OF THE PHYSICAL EXAMINATION RESULTS

The theme of the paper is the design and development of a functional mobile application that interprets the results of a physical examination. The most similar solutions are presented and the development environment in which one develops is defined. Java programming language and Android studio are used for the development of the program, which in addition to programming is used as a virtual tool for testing the application. The application uses the Firebase service to store and retrieve the data required to function certain features. It also has an authentication feature that allows each user to access only their reviews. If the user does not have an account, he can register within the application itself. The logged in user also has the option of performing an examination, which is then saved to the database and later displayed in the review list. The resulting parameters are customized for examinations of specific age groups so that users under the age of twenty are not able to use the examination function. Clicking on one of the reviews in the list triggers a new activity that provides more detailed information about that review with icons representing the status for each parameter. Another feature of the app is to view a map of nearby medical facilities performing physical examinations.

Keywords: Android, Java, Firebase, physical examination, authentication, maps

ŽIVOTOPIS

Juraj Fekete rođen je 5.8.1993. u Osijeku. Osnovnu školu je pohađao u Višnjevcu. Za to vrijeme aktivno je trenirao nogomet. 2008. upisuje Tehničku školu i prirodoslovnu gimnaziju Ruđer Bošković. Za to vrijeme počinje svirati u tamburaškoj školi Batorek.. Nakon završetka srednje škole započinje stručni studij informatike na Elektrotehničkom fakultetu u Osijeku. U razdoblju travnja 2017. do siječnja 2018. pohađa stručno osposobljavanje za developera mobilnih aplikacija na Elektrotehničkom fakultetu u Osijeku. U razdoblju lipanj 2019. do kolovoz 2019. pohađa stručno osposobljavanje za 3D dizajnera na otvorenom učilištu Algebra.

Potpis: _____