

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 271

POKRETNİ KLİJENT ZA PRAĆENJE SENZORSKIH OČITANJA

Dario Mesić

Zagreb, lipanj 2021.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu \izvornik.

SADRŽAJ

| | |
|---|-----------|
| 1. Uvod | 1 |
| 2. Opis projektnog zadatka | 3 |
| 3. Arhitektura i dizajn sustava | 5 |
| 3.1. Baza podataka | 6 |
| 3.1.1. InfluxDB | 6 |
| 3.1.2. Firebase | 6 |
| 4. Implementacija | 8 |
| 4.1. Korištene tehnologije i alati | 8 |
| 4.1.1. Operacijski sustav Android | 8 |
| 4.1.2. Android Studio | 9 |
| 4.1.3. Programski jezik Java | 10 |
| 4.1.4. XML | 10 |
| 4.1.5. Rest API | 11 |
| 4.1.6. CSV | 12 |
| 4.2. Ispitivanje programskog rješenja | 13 |
| 4.3. Upute za puštanje u pogon | 16 |
| 4.3.1. Produkcijско okruženje | 16 |
| 4.3.2. Lokalno pokretanje za razvoj | 16 |
| 4.4. Mogući problemi | 17 |
| 5. Android aplikacija | 18 |
| 5.1. Stvaranje projekta | 18 |
| 5.2. Funkcionalnosti aplikacije | 20 |

| | |
|--|-----------|
| 5.3. Zahtjevi i karakteristike sustava | 26 |
| 5.4. Moguća poboljšanja | 27 |
| 5.4.1. Nadogradnja baze podataka | 27 |
| 5.4.2. Korisničko iskustvo | 27 |
| 5.4.3. Proširenje sustava | 27 |
| 6. Zaključak i budući rad | 28 |
| Literatura | 29 |
| Popis slika | 31 |

1. Uvod

U današnjem Internetu raste broj jednostavnih uređaja poput senzorskih čvorova koji odašilju različite podatke. Da bi se od tih podataka dobile konkretne informacije, potrebno ih je pohraniti, analizirati i pregledno prikazati korisnicima

Život bez Interneta je danas nezamisliv. Susrećemo se s njim u raznim pogledima svakodnevnog života, a određene tehnologije i uređaje rabimo i u najvažnijim kućanskim zadacima. Iot, odnosno Internet of things iliti Internet stvari označava upravo to povezivanje uređaja putem Interneta. Njegov smisao je omogućiti ljudima živjeti kvalitetnije, smanjiti troškove, olakšati stvari te raditi pametnije. Iot čine ljudi, infrastruktura, stvari, procesi i podaci, a svaki od njih ima veliku ulogu u cjelokupnoj mrežnoj infrastrukturi.

Koncept IoT-a odnosi se na povezanost različitih fizičkih objekata putem interneta, a prvi put ga je spomenuo britanski poduzetnik Kevin Ashton, iz Procter i Gamblea, kasnije zaposlenik MIT-ovog Auto-ID Centra, 1999. godine. IoT postaje veliki posao budućnosti te će zasigurno biti u fokusu četvrte industrijske revolucije odnosno sveopće digitalizacije koja podrazumijeva senzore u svim područjima, život u „cloudu“ i internetsku povezanost ne samo nas već i svega što nas okružuje.[8] Naravno da sve što je dobro ima i svoje rizike i opasnosti. S obzirom na to da je tehnologija danas toliko zastupljena, ključno je održati povjerljivost i integritet podataka. Sve više se susrećemo s protivnicima tehnologija koji navode kako im je sigurnost narušena te traže promjene. Iot se koristi i u nadzoru prometa, kontroli javnog prijevoza, sigurnosnim kamerama pa i ostalim službama te se ljudi jednostavno počinju osjećati "promatrano". Internet je evoluirao kroz četiri faze tako što je svaka bila zapravo nadograđena na onu prethodnu te je ostavila veliki trag u tadašnjem društvu.

Internet stvari sastoji se od više stotina milijunskih senzora koji proizvode svoje podatke u realnom vremenu te samostalno komuniciraju jedni s drugima i sa raznim aplikacijama.

S obzirom na to da senzori imaju mogućnost generiranja puno podataka, potrebno je na lijep i pregledan način prikazati sve te podatke korisniku. Najbolji način za to je izrada

aplikacije.

Praktični dio ovoga rada je aplikacija za Android, ali prije svega treba se dobro upoznati sa strukturom te svim funkcionalnostima aplikacije kako bi se mogle prikazati njegove mogućnosti.

Završni rad podijeljen je u pet međusobno povezanih cjelina u kojoj svaka sadrži najbitnije stvari u vezi aplikacije kako bi se što bolje obradila tema.

Drugi dio "Opis projektnog zadatka" nudi pregled svih funkcionalnosti zadatka. Aplikacija je podijeljena u nekoliko aktivnosti te je svaka ukratko opisana zajedno s mogućnostima koje korisnik može ostvariti u svakoj od aktivnosti.

U trećem djelu "Arhitektura i dizajn sustava" opisani su sustavi od kojih se aplikacija sastoji. Dakle tu je ukratko opisano sučelje koje se koristi za izradu aplikacije zajedno s bazama podataka koje su bile korištene za prikaz senzorskih podataka.

Četvrti dio "Implementacija" je podijeljen u više podnaslova u kojima se opisuju korištene tehnologije, ispituju programska rješenja, daju upute za puštanje aplikacije u pogon te su prikazani mogući problemi.

U petom dijelu "Android aplikacija" dan je uvid u sve aktivnosti aplikacije zajedno s njezinim funkcionalnostima. Također su opisane moguće nadogradnje u aplikaciji zajedno sa zahtjevima i karakteristikama sustava.

U poglavlju "Zaključak" dana je sistematizacija cjelokupnog rada, kao i spoznaje do kojih je došlo.

2. Opis projektnog zadatka

Cilj ovog projekta je razviti mobilnu aplikaciju koja će ponuditi svakom korisniku uvid u najnoviju vremensku prognozu, grafove očitavanja podataka sa senzora te mnoge druge opcionalnosti koje aplikacija pruža. Aplikacija će tako olakšati korisniku pristup vremenskoj prognozi te omogućiti im prikaz drugih senzorskih podataka po njihovoj želji.

Senzori su uređaji koji pretvaraju ulazni signal u električni analogni ili digitalni izlaz koji je čitljiv te tako omogućuje da se određeni fizički parametar, poput temperature ili vlažnosti, pretvori u signal. Senzori zapravo prevode opće neelektrične vrijednosti u električnu vrijednost.

FER-ov Laboratorij za Internet stvari postavio je nekolicinu senzora oko područja na fakultetu te je sada potrebno preuzeti podatke s tih senzora te prikazati ih lijepo grafički putem aplikacije. Na raspolaganju su nam brojni senzori, ali oni koji nas najviše zanimaju su ovi vezani uz vremensku prognozu. Iako danas postoji puno sličnih rješenja vezanih za prikaz senzorskih očitavanja ili prikaz vremenske prognoze, poanta ove aplikacije je da omogućiti korisniku pristup najnovijim podacima iz različitih spektara vremenske prognoze na najjednostavniji mogući način za korisnika. Podaci će se moći prikazati na različite načine te će korisnik imati uvid i u njihove grafičke vrijednosti kroz određeni period vremena.

Sensor dashboard

| Search for ... <input type="text"/> <input type="button" value="Search"/> | | | | <input type="button" value="+ Create sensor"/> | |
|---|--|------------------------------|-------------------------|--|--|
| Online | Name | Username | Last seen at | | |
| Offline | Stanko's test sensor | stanko_test | 2021-02-10 14:48:04 UTC | | |
| Offline | Parking sensor | parking | 2021-04-03 08:56:05 UTC | | |
| Offline | Meto-plus-1 | pavle | 2021-05-24 11:08:03 UTC | | |
| Online | Waspmote-agriculture-plug&play | iotlab-agri1 | 2021-05-28 18:05:03 UTC | | |
| Online | Meto-plus-2 | pavle2 | 2021-05-28 18:05:03 UTC | | |
| Online | IoTlab-Waspmote-temp_hum_lum_pres_hall | iotlab | 2021-05-28 18:05:03 UTC | | |

Slika 2.1: Sensor dashboard

Neki od sličnih Iot rješenja su i pametni satovi, pametna poljoprivreda, pametni frižider... **Pametni satovi** su vjerojatno jedan od najraširenijih proizvoda s kojima se danas susrećemo. Takav pametan sat zamijenit će hrpu gadgeta koji su nam neophodni. Lagan je za uporabu te može biti uz nas čitavo vrijeme. Osim što je odličan u sportu, olakšava komunikaciju u vožnji i pruža neovisnost od mobitela. Pametni satovi se razvijaju sve više s dolaskom drugačijih i naprednijih operativnih sustava. Gotovo svi proizvođači elektroničkih uređaja u ponudi imaju pametne satove tako da više nije namijenjen osobama dubljeg džepa.

Pametna poljoprivreda pruža lakša rješenja u poljoprivredi jer ima mogućnost obavljanja različitih zadataka, uštede vremena te povećanju produktivnosti. Na primjer, pametni uređaji mogu kontrolirati vlažnost tla i po potrebi uključiti navodnjavanje. Mogu kontrolirati stanje usjeva i u ovisnosti od vanjskih uvjeta odrediti idealno vrijeme za sadnju, obradu ili berbu različitih kultura istovremeno.

Osim ova dva primjera, sve više se susrećemo s pojmom **Pametni frižider**. Frižider se spaja na tablet uređaj te pomoću njega korisnik može pogledati što sve nedostaje u frižideru. Pametni frižider također ima opciju pamćenja recepata na osnovu namirnica koje korisnik ima na zalihi te praćenja datuma isteka roka trajanja određenih proizvoda. To su sve pametna rješenja s kojima se susrećemo iz dana u dan, a razvijaju se zahvaljujući Iot tehnologiji. [1]



Slika 2.2: Primjer Iot tehnologije

3. Arhitektura i dizajn sustava

Arhitektura programske podrške se sastoji od dva podsustava:

- **Mobilna aplikacija**
- **Baza podataka**

Mobilna aplikacija

Za izradu mobilne aplikacije - sučelja putem kojeg krajnji korisnici koriste sustav koristio se **Android studio**. Android studio je baziran na uporabi programskog jezika Jave ili Kotlin. Svaka aplikacija sastoji se od manifest datoteka nazvanih aktivnosti koje su pisani u XML. programskom jeziku. Android manifest sadrži ključne informacije o aplikaciji koje operacijski sustav mora imati prije pokretanja aplikacije. Aplikacija je spojena na opću knjižnicu android sustava koja uključuje osnovne pakete za izgradnju aplikacija. Korisnik ima mogućnost dodavanja paketa po volji iz različitih izvora, ali ih mora navesti u build pathu.

Osim manifesta, android aplikacija sadrži i folder java/ koji definira klase organizirane po projektima. Ovdje programer piše svoj kod koji će zatim primijeniti u svakoj aktivnosti (manifestu). Aplikacija sadrži i folder values u kojemu korisnik dodaje svoje stilove, boje.. Build gradle(project) služi za kompajliranje određenih modula te paketa.

Aplikacija se kompajlira putem emulatora koji je besplatan i jednostavan za testiranje Android aplikacije. Korisnik prije pokretanja može odabrati preko kojeg operativnog sustava te preko koje vrste zaslona želi pokrenuti vlastitu aplikaciju. Dizajn svoje stranice može se vidjeti i bez pokretanja aplikacije odabirom određenog manifesta te klikom na Design.

Svrha mobilne aplikacije je da poslužuje informacije te da daje pristup korisniku na intuitivan i oku ugodan načina.

3.1. Baza podataka

Za aplikaciju praćenja senzorskih očitavanja koristile su se dvije baze podataka. Jedna od njih, Firebase služila je za pohranu korisnika, dok je druga, InfluxDB, služila za pristup svim podacima koje su se koristile u aplikaciji. Razlog korištenja baze podataka je u brzom i jednostavnom pohranu, izmjeni i dohvaćanju podataka za daljnju obradu.

3.1.1. InfluxDB

InfluxData je platforma koja nudi različite vremenske serije s ciljem implementacije, promatranja, učenja i automatizacije svih vrsta sustava, aplikacija i poslovnih procesa u različitim radnim okruženjima. Glavne značajke baze su:

- sposobnost promatranja i automatizacije ključnih sustava, infrastrukture, aplikacija i poslovnih procesa
- analizirati i automatizirati senzore i uređaje u stvarnom vremenu, koji generiraju informacije za administrativne zadatke
- više opcija instrumentacije koje otkrivaju obrasce uporabe i odatle se mogu stvoriti nove poslovne mogućnosti

U završnom radu pristup bazi je bio onemogućen tako da se podatke uzimalo preko Rest API sučelja koje je opisano kasnije. [2]

3.1.2. Firebase

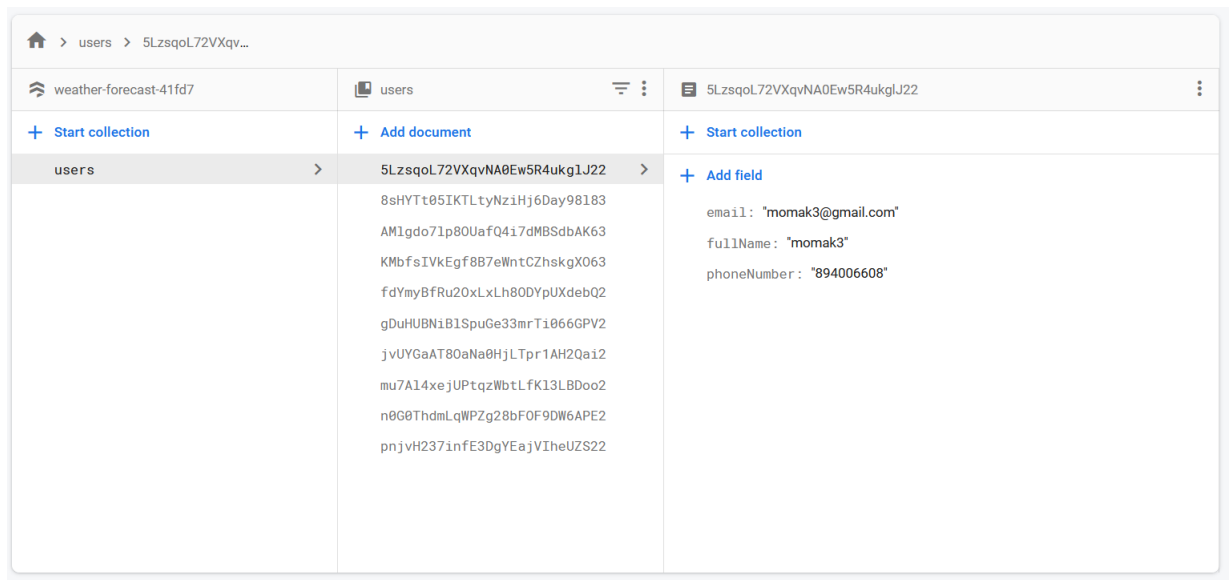
Kao što je već opisano, Firebase je korišten u cilju najbržeg i najlakšeg spremanja korisnika u sustav. Firebase je inače Googleova platforma koja sadrži mnoge mogućnosti koje programerima olakšavaju razvoj mobilnih ili web aplikacija.

Platforma Firebase je na svome početku bila baza podataka u stvarnom vremenu, kasnije ju je kupio Google i proširio te pretvorio u sveobuhvatnu platformu koja omogućuje i analizu podataka, odnosno na osnovi dobivenih podataka moguće je donijeti ispravnu odluku. Prilikom korištenja Firebase platforme potrebno je ubaciti Firebaseu svoj projekt.

Android Studio je olakšao pristup Firebase-u tako da je sam pristup platformi vrlo jednostavan te čak detaljno objašnjen na samom Android Studiju. S obzirom je trebalo spremi

korisnika te njegove osnovne podatke, u Firebaseu je korištena opcija *Authentication* koja korisniku omogućuje jednostavno spremanje korisnika u bazu.

Svi trenutni i postojeći zapisi korisnika mogu se vidjeti na službenoj stranici Firebasea. Korisnik Firebasea ima uvid u sve spremljene korisnike aplikacije te datum njihovog stvaranja, njihov zadnji login, email adresu te hashirani password. On također ima funkciju jednim klikom maknuti korisnika iz baze.



Slika 3.1: Firebase korisnici prijavljeni u sustav

4. Implementacija

Za implementaciju aplikacije koristili su se različiti razvojni alati s namjerom da se na najlakši način kreiraju određeni dijelovi aplikacije. Razvojni alati su inače skup programskih alata koji omogućavaju kreiranja aplikacija za određene skupine programa, programskog okruženja, hardvera, OS-s i sličnih platformi. Sastoje se od određene skupine gotovih programskih rješenja koji imaju mogućnost komunikacije sa specifičnim programom ili platformom. [3] Najpoznatiji razvojni alati za korištenje su GitHub, YouTube, Visual Studio ...

4.1. Korištene tehnologije i alati

Korištene tehnologije za izradu ove aplikacije objašnjene su i detaljno opisane u ovome poglavlju. Najprije je opisan rad sustava Android zajedno s korištenjem Android Sustava kao razvojnog okruženja, zatim korištenje programskog alata Java, uporaba XML-a te Rest API-a.

4.1.1. Operacijski sustav Android

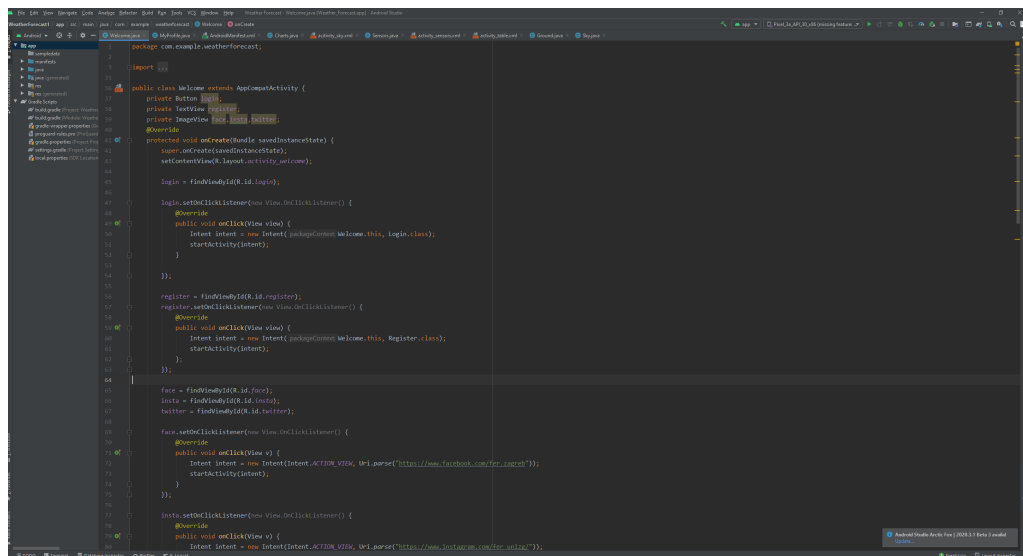
Razvojem mobilnih uređaja nastajala je sve veća potreba za stvaranje sustava koji bi dodatno olakšao korištenje takvih uređaja uz neke nove mogućnosti. Firma Android Inc. osnovanoj 2003. godine osnovna djelatnost bila je razvoj programske potpore za pametne mobilne uređaje. Ovaj operacijski sustav je modularan i prilagodljiv, te se osim u pametnim telefonima i tabletima koristi i u drugim naprednim uređajima, primjerice pametnim televizorima, pametnim satovima... Android je zasnovan na jezgri Linux 2.6 i napisan u programskom jeziku C/C++.[7]

4.1.2. Android Studio

Android Studio (SI.2.2) je službeno integrirano razvojno okruženje(engl. Integrated development environment, IDE) namijenjeno za izradu aplikacije koje koriste Android operacijski sustav. Izgrađen je na temelju JetBrains' IntelliJ IDEA programskoj podršci (softveru) specifično dizajniranoj za razvijanje Android aplikacija. Objavljen je na Google-ovoj konferenciji razvojnih inženjera.

Prva stabilna inačica objavljena je 2014. godine. Dostupan je i na drugim operacijskim sustavima. Neke od boljih značajki su specifično refaktoriranje i brzi popravci, čarobnjaci na temelju predložaka za stvaranje uobičajenih Android-ovih dizajna i komponenti, poseban emulator (Android Virtual Device) za pokretanje i uklanjanje pogrešaka u aplikacijama unutar Android Studija. Kako bi se lakše refaktorirao ili popravio kôd nude se informacije za ispravak unutar linije kôda koja se trenutno piše te izbacuje određene probleme.

Neki od problema ili potencijalnih opasnosti može biti upućivanje objekata na koje se odnosi odabrani objekt kako programer ne bi pogriješio u korištenju koncepta nasljeđivanja. Takav način upozorenja može izbaciti informaciju o povratnoj vrijednosti metode, lambda i izrazu operatera, opisnih vrijednosti. Također se unutar Android Studija nudi profiliranje performansi kako bi se moglo pratiti kako računalo reagira na izradu aplikacije.[10]



Slika 4.1: Android studio

4.1.3. Programski jezik Java

Java je objektno orijentirani programski jezik koji je razvila tvrtka zvana Sun Microsystems-a koju danas posjeduje korporacija Oracle. Razvoj je počeo 1991., kao dio projekta *Green*, a objavljen je u studenom 1995. U to vrijeme, glavni izbor programskog jezika je bio C++. No, s vremena na vrijeme, poteškoće s jezikom C++ su dovele do točke gdje je najjednostavnije bilo napraviti novu jezičnu platformu, čiji je rezultat bio Java koja se prikazala kao idealna za izradu sigurnih te jednostavnih aplikacija koje će biti lako dobavljene korisnicima. U odnosu na ostale programske jezike, Java nudi mogućnost izvođenja bez preinaka na svim operativnim sustavima za koje postoji JVM (Java Virtual Machine), dok drugi klasični programski jezici, poput C-a, koriste posebno prilagođivanje Operacijskom sustavu na kojem se izvode. Time i bogatim skupom klasa za rad s mrežnim komunikacijama u jednom trenutku je Java bila najbolji izbor za široku lepezu mogućih aplikacija.

Programski jezik Java je dizajniran s namjerom rješavanja raznih zadataka i izazova koji nastaju prilikom razvoja aplikacija u okolišu kakvog nam nude internet i široko rasprostranjene mreže. Java se smatra jednim od najpopularnijih programskih jezika jer procjene govore kako se njime služi i do 10 milijuna korisnika, a koristi se i koristi kao osnovni jezik za programiranje Googleovog sustava Android.[9]

4.1.4. XML

XML je kratica od “Extensible Markup Language”, što znači da je zapravo riječ o proširivom jeziku za označavanje. Njime zapisujemo dokumente i podatke u tekstualnom formatu koji je čitljiv za tekst editore kao što su npr. MS Word ili za razne programerske editore. Najčešće se sprema kao tekstualna datoteka koja se sastoji od sadržaja i oznaka.

Iako je prvenstveno namijenjen za programsku obradu, XML format je čitljiv i ljudima. Neke od mogućih upotreba XML-a su: razmjena, pohrana i povećanje dostupnost podataka. XML se često primjenjuje u definiranju matematičkih formula, lakši je za opis tvrdnji i poveznica s tvrdnjama, kao jezik za integraciju električkih sustava (CIM).[6]

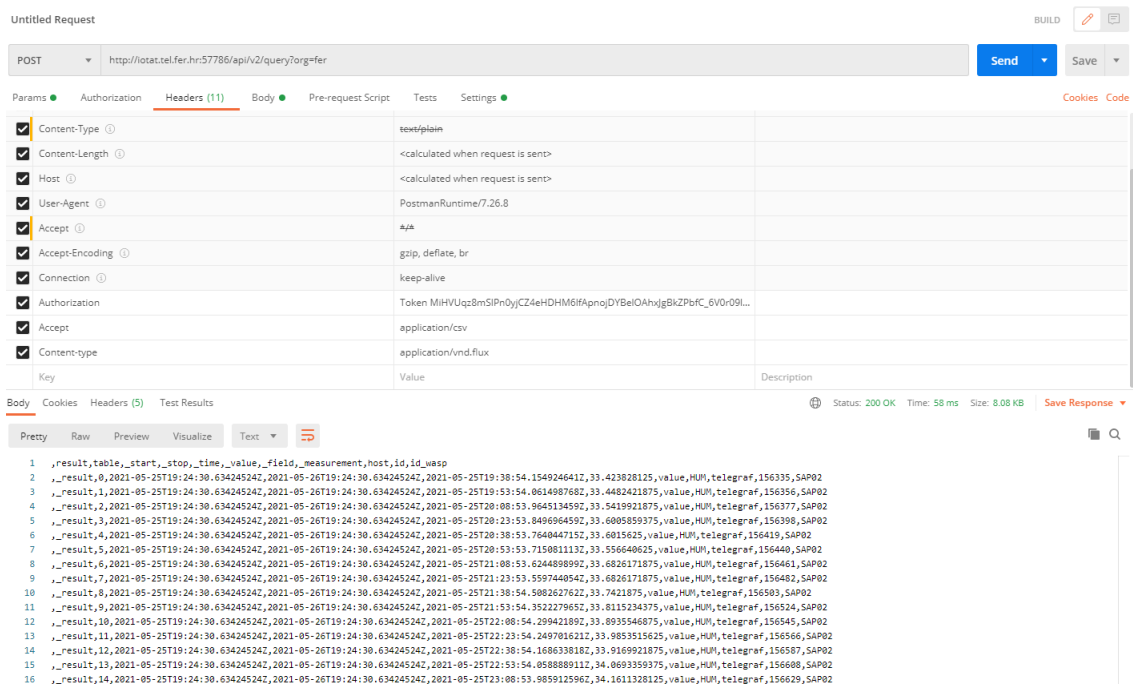


Slika 4.2: XML file sa stranice DHMZ

4.1.5. Rest API

Zadaća pozadinskog sustava je povezati bazu podataka u kojoj su spremljeni svi podaci sustava s klijentskim aplikacijama koje trebaju čitati i upravljati tim podacima uz poštovanje aplikacijske logike i ograničenja sustava. To je način pristupanja mrežnim resursima korištenjem HTTP protokola i odgovarajućih ruta koje označavaju željeni resurs.

Odgovor najčešće dolazi u obliku JSON ili XML formata. Vanjske aplikacije mogu koristiti REST API za upite i ažuriranja aplikacijskih podataka. Njegovi resursi se mogu koristiti bez ikakve konfiguracije, a podrška kreira, radi upit, ažurira i briše operacije na resursima koristeći standardne metode HTTP GET, POST, PUT i DELETE. [5]



Slika 4.3: Testiranje REST sučelja pomoću programa Postman.

4.1.6. CSV

Zarezom odvojene vrijednosti (CSV) je tekstualna datoteka koja, kao što joj i ime govori, koristi zarez za odvajanje vrijednosti. Svaki redak datoteke predstavlja zapis podataka. Ti zapisi se sastoje od jednog ili više polja koji su odvojeni separatorom, zarezom. CSV datoteke su dizajnirane kako bi bile jednostavne za izvoz i uvoz u programe. Njihov rezultat je jednostavno čitljiv za ljude te ga možemo vidjeti unutar tekstualnih uređivača poput Notepada ili Microsoft Excela. [4]

4.2. Ispitivanje programskog rješenja

U ovom potpoglavlju pokazat će se najbitniji dijelovi koda koji su zaslužni za povezivanje s vanjskim sustavima, poput baze podataka, API sučelja..

Ispitni slučaj 1: Registracija korisnika

Izvorni kod:

```
fAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
            Toast.makeText(Register.this, "User created.", Toast.LENGTH_SHORT).show();

            userID = fAuth.getCurrentUser().getUid();
            DocumentReference documentReference = fStore.collection("users").document(userID);

            Map<String, Object> user = new HashMap<>();
            user.put("email", email);
            user.put("fullName", fullName);
            user.put("phoneNumber", phoneNumber);
            documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    Log.d("message", "User profile is created for" + userID);
                }
            });

            startActivity(new Intent(getApplicationContext(), Sensors.class));
        } else {
            Toast.makeText(Register.this, "Error !" + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
            progressBar.setVisibility(View.INVISIBLE);
        }
    }
});
```

Ispitni slučaj 2: Prijava korisnika

Izvorni kod:

```
fAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
            Toast.makeText(Login.this, "Logged in successfully", Toast.LENGTH_SHORT).show();
            startActivity(new Intent(getApplicationContext(), Sensors.class));
            progressBar.setVisibility(View.INVISIBLE);
        } else {
            Toast.makeText(Login.this, "Error !" + task.getException().getMessage(), Toast.LENGTH_SHORT).show();
            progressBar.setVisibility(View.INVISIBLE);
        }
    }
});
```

Ispitni slučaj 3: Pristup DHMZ podacima

Izvorni kod:

```
OkHttpClient client = new OkHttpClient();
String url = "https://prognosa.hr/sedam/hrvatska/7d_meteogrami.xml";

okhttp3.Request request = new okhttp3.Request.Builder()
    .url(url)
    .build();

client.newCall(request).enqueue(new Callback() {
    @Override
    public void onFailure(Call call, IOException e) {
        e.printStackTrace();
    }

    @Override
    public void onResponse(Call call, okhttp3.Response response) throws IOException {
        if (response.isSuccessful()) {
            final String myResponse = response.body().string();
            pom = myResponse;
            parseXML(myResponse);
        }
    }
});
```

Ispitni slučaj 4: Pristup podacima sa senzora

Izvorni kod:

```
String data = "from(bucket: \"telegraf\")\r\n"
+ "    |> range( start : -24h)\r\n"
+ "    |> filter (fn: (r) => r._measurement == \"HUM\" and r.id_wasp == \"SAP02\")";
try {
    byte[] out = data.getBytes();
    url = new URL("http://iotat.tel.fer.hr:57786/api/v2/query?org=fer");
    HttpURLConnection http = (HttpURLConnection) url.openConnection();
    http.setRequestMethod("POST");
    http.setDoOutput(true);
    http.setInstanceFollowRedirects(true);
    http.setRequestProperty("Authorization", "Token MiHVUqz8mSlPn0yjCZ4eHDHM6lfApnojDYBelOAhxJgBkZPbFC_6V0r09ljoqr3gz8MaG7d-PQDNW4e0yA6pgw==");
    http.setRequestProperty("Accept", "application/csv");
    http.setRequestProperty("Content-type", "application/vnd.flux");
    http.connect();

    try {
        OutputStream stream = http.getOutputStream();
        stream.write(out);
    }
    catch (Exception e) {
        e.printStackTrace();
    }

    BufferedReader in = new BufferedReader(new InputStreamReader(http.getInputStream()));
    String inputLine;
    StringBuffer response = new StringBuffer();
    while((inputLine = in.readLine()) != null){
        response.append(inputLine);
    }
}
```

```
}  
}
```

Ispitni slučaj 5: Pristup korisničkim podacima iz baze podataka

Izvorni kod:

```
fAuth = FirebaseAuth . getInstance () ;  
fStore = FirebaseFirestore . getInstance () ;  
user = fAuth . getCurrentUser () ;  
userId = user . getUid () ;  
noteReference = fStore . collection ( " users " ) . document ( userId ) ;  
  
noteReference . get ()  
    . addOnSuccessListener ( new OnSuccessListener < DocumentSnapshot > () {  
        @Override  
        public void onSuccess ( DocumentSnapshot documentSnapshot ) {  
            if ( documentSnapshot . exists () ) {  
                email . setText ( documentSnapshot . getString ( " email " ) ) ;  
                fullName . setText ( documentSnapshot . getString ( " fullName " ) ) ;  
                phoneNumber . setText ( documentSnapshot . getString ( " phoneNumber " ) ) ;  
                userName . setText ( documentSnapshot . getString ( " fullName " ) ) ;  
            }  
            else {  
                Toast . makeText ( MyProfile . this , " Document does not exist " , Toast . LENGTH_SHORT ) . show () ;  
            }  
        }  
    } )  
    . addOnFailureListener ( new OnFailureListener () {  
        @Override  
        public void onFailure ( @NonNull Exception e ) {  
        }  
    } ) ;
```

Ispitni slučaj 7: Promjena korisničkih podataka

Izvorni kod:

```
saveBtn . setOnClickListener ( new View . OnClickListener () {  
    @Override  
    public void onClick ( View v ) {  
        if ( mFullName . getText () . toString () . isEmpty () || mEmail . getText () . toString () . isEmpty () || mPhoneNumber . getText () . toString () . isEmpty () ) {  
            Toast . makeText ( EditProfile . this , " One or many fields are empty . " , Toast . LENGTH_SHORT ) . show () ;  
            return ;  
        }  
  
        final String emailF = mEmail . getText () . toString () ;  
        user . updateEmail ( emailF ) . addOnSuccessListener ( new OnSuccessListener < Void > () {  
            @Override  
            public void onSuccess ( Void aVoid ) {  
                DocumentReference docRef = fStore . collection ( " users " ) . document ( user . getUid () ) ;  
                Map < String , Object > edited = new HashMap < > () ;  
                edited . put ( " email " , emailF ) ;  
                edited . put ( " fullName " , mFullName . getText () . toString () ) ;  
                edited . put ( " phoneNumber " , mPhoneNumber . getText () . toString () ) ;  
                docRef . update ( edited ) ;  
                Toast . makeText ( EditProfile . this , " Email is changed . " , Toast . LENGTH_SHORT ) . show () ;  
                Intent intent = new Intent ( EditProfile . this , Settings . class ) ;  
            }  
        } ) ;  
    }  
} ) ;
```

```

        startActivity ( intent );
    }
}). addOnFailureListener (new OnFailureListener () {
    @Override
    public void onFailure ( @NonNull Exception e ) {
        Toast.makeText( EditProfile . this ,e.getMessage(), Toast.LENGTH_SHORT).show();
    }
});
}
});

```

4.3. Upute za puštanje u pogon

Kako bi korisnik mogao dobiti uvid u aplikaciju, omogućena su mu dva pristupa opisana u sljedećim poglavljima.

4.3.1. Produkcijsko okruženje

Najlakši način za pokretanje aplikacije je preko APK formata. APK, odnosno **Android application package** je format datoteke koji koristi operativni sustav Android i niz drugih operativnih sustava temeljenih na Androidu za distribuciju te instalaciju mobilnih aplikacija, mobilnih igara i međuopreme. APK će biti dostupan za korisnika preko GitLab-a (nastavak .api). Nakon što je korisnik kliknuo na zadani APK, aplikacija će se instalirati na njegov uređaj te će korisnik morati potvrditi sve zahtjeve kako bi se na kraju ona mogla otvoriti. Aplikacija je sigurna te je korisnik zaštićen od bilo kakvog virusa.

4.3.2. Lokalno pokretanje za razvoj

Za lokalno pokretanje aplikacije potrebno je preuzeti projekt s GitLab-a te importati ga u Android Studiju. Nakon toga će se instalirati sve komponente koje su se koristile te će korisnik moći pokrenuti aplikaciju. Korisnik ima mogućnost pokretanja aplikacije na različitim uređajima, poput tableta, mobitela, pametnih satova... U gornjoj alatnoj traci odabire gumb **Run app**. Aplikacija će se zatim otvoriti na emulatoru koji je besplatan i jednostavan za testiranje Android aplikacije. Ako korisnik želi zaustaviti testiranje aplikacije u gornjoj alatnoj traci treba odabrati gumb **Stop**.

4.4. Mogući problemi

Naravno, postoje određeni problemi koji se mogu dogoditi prilikom izvođenja programa. Najčešći problem je status senzora koji u bilo kojem trenutku može biti nedostupan te tako neće moći biti u mogućnosti za detaljan pregled korisniku. Taj problem je riješen uz pomoć jednog senzora koji će u svakom trenutku biti dostupan za pregled.

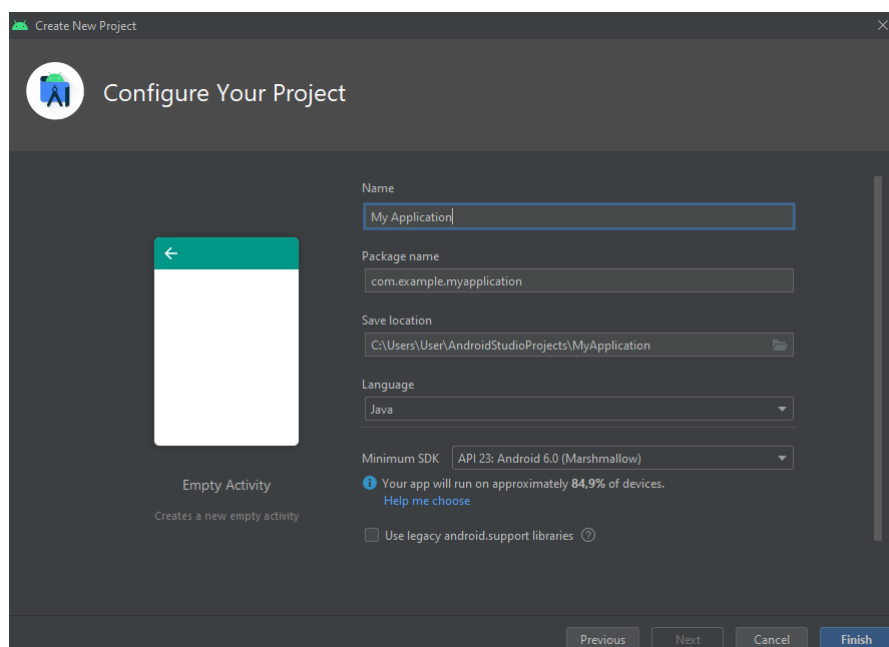
Još jedna mogućnost je da jedna od baze podataka jednostavno ne bude funkcionalna te tako korisnik neće moći uopće pristupiti senzorima. S obzirom na to da se kao baza podataka koristila Googleova platforma Firebase, šanse za tako nešto su ipak minimalne.

5. Android aplikacija

Kako bi korisnici mogli koristiti sustav za prikaz senzorskih očitavanja, izrađena je mobilna Android aplikacija kao glavni način interakcije sa sustavom. Koristeći službenu dokumentaciju te raznu literaturu aplikacije se izrađivala kontinuirano, kroz više iteracija, nadograđivala novijim tehnologijama, modernijim bibliotekama kako bi rezultat na kraju bio što bolji.

5.1. Stvaranje projekta

Ulaskom u Android Studio nude se mogućnosti stvaranja novog projekta, učitavanje projekta iz drugih izvora i druge. Kako bi se stvorila aplikacija odabiremo **Start a new Android Studio project**.

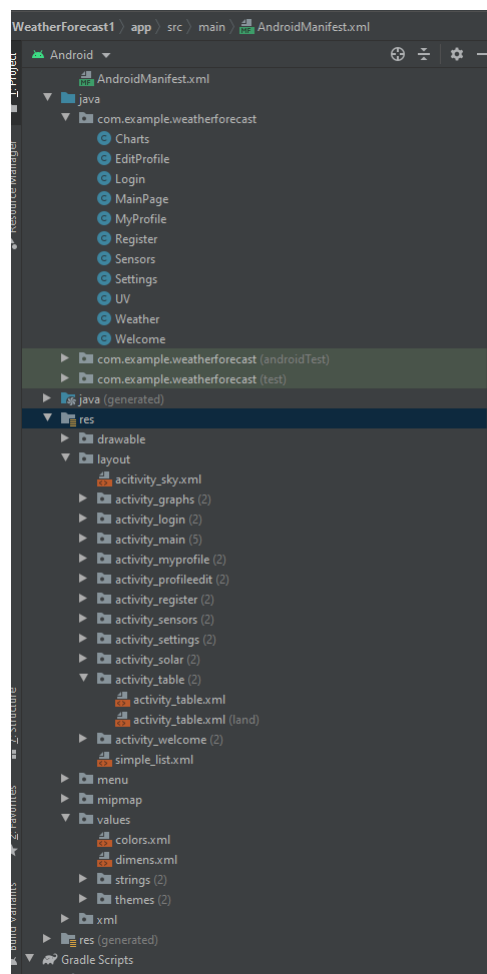


Slika 5.1: Stvaranje novog projekta

Projekt se sastoji od tri glavna dijela: manifests, java i res. Struktura je zamišljena tako da se u manifest dijelu nalaze važne informacije ili određene potvrde koje su potrebne za izradu

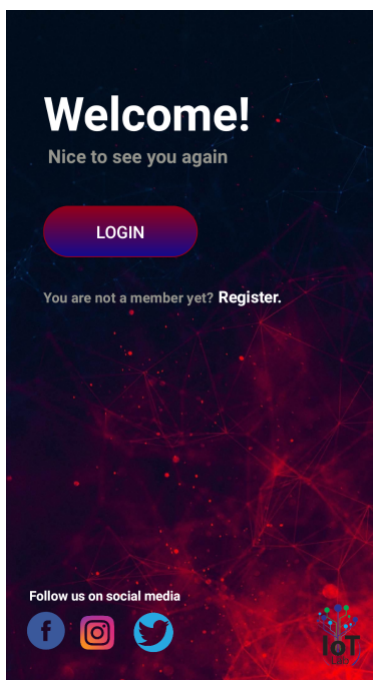
aplikacije. Ako aplikacija zahtijeva korištenje interneta mora se upisati nova linija koda koja glasi **uses-permission android:name="android.permission.internet"**. Bez toga se ne može funkcionirati s vanjskim sustavima koji su povezani preko Interneta.

Ako pratimo strukturu projekta, vidjet ćemo kako prvo na red dolazi java dio. Ovdje se nalaze klase, pisane u programskom jeziku Java, od kojih je svaka povezana sa svojim page layoutom. Također, u ovom dijelu se povezuje aplikacija s vanjskim sustavima. Zatim slijedi res dio(resursi), koji je zapravo odgovoran za spremanje svih slika, fontova, stilova. Ovdje se spremaju podaci koji će zatim biti prikazani u aplikaciji. Većinu resursa čine xml datoteke u kojima su zapisani svi ti podaci. Na kraju dolazimo do foldera gradle. Ovdje su sadržane biblioteke koje aplikacija koristi. Ako se želi dodati određena biblioteka u aplikaciju to se može napraviti u datoteci **build.gradle**. Za rad ove aplikacije trebat će implementirati dodatne biblioteke uz standardno dobivene od Android Studija. Biblioteke koje su se koristile u ovoj aplikaciji uglavnom su bile povezane s različitim stilovima i fontovima.



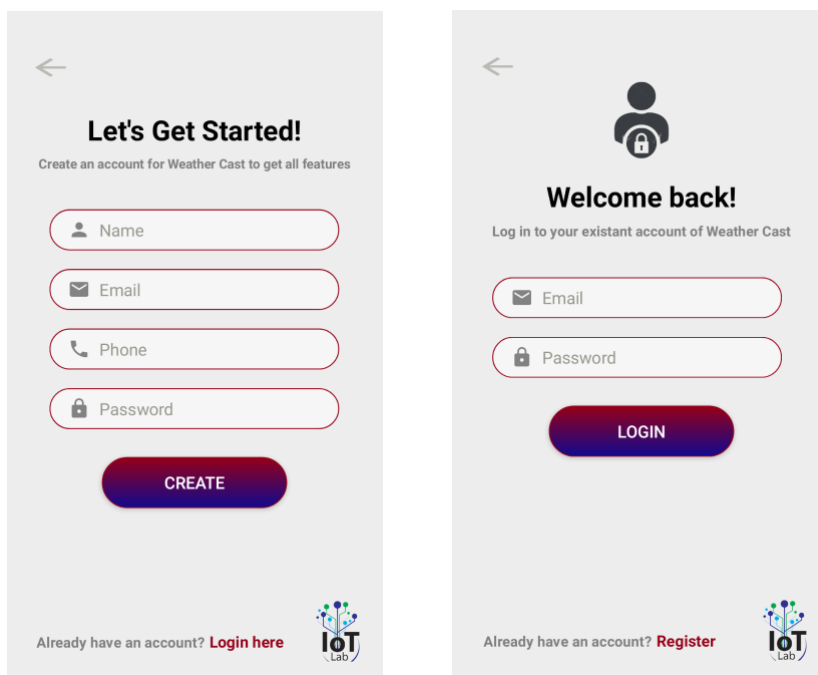
Slika 5.2: Struktura projekta aplikacije Weather Forecast

5.2. Funkcionalnosti aplikacije



Slika 5.3: Welcome page

Nakon pokretanja sustava, korisnik je preusmjeren na početnu stranicu koja nudi mogućnosti prijavljivanja ili registracije korisnika u sustav te pristup socijalnim mrežama s jednostavnim klikom na njihov gadget.

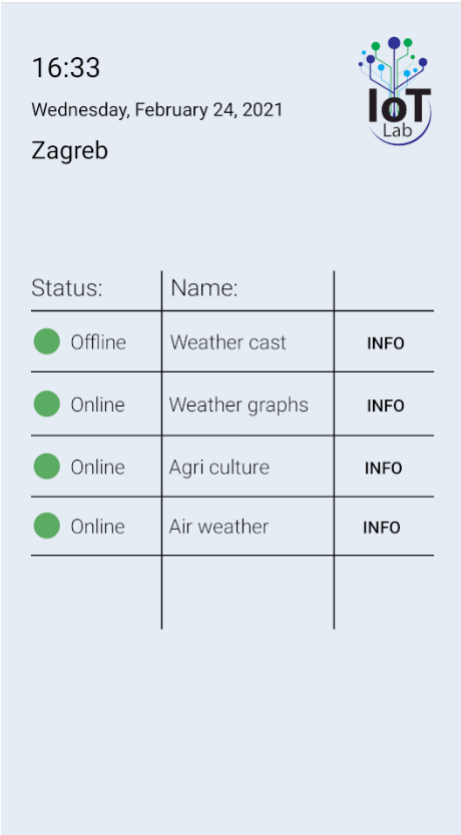



Slika 5.4: Register and Login

Na stranici Registriraj se korisnik može stvoriti svoj vlastiti račun. Za kreiranje novog računa potrebni su sljedeći podaci:

- korisničko ime
- email adresa
- broj telefona
- lozinka

Na stranici Prijavi se korisnik se može prijaviti u svoj račun ako ga je prethodno već napravio. Za prijavu u sustav potrebni su **korisničko ime** te **lozinka**.

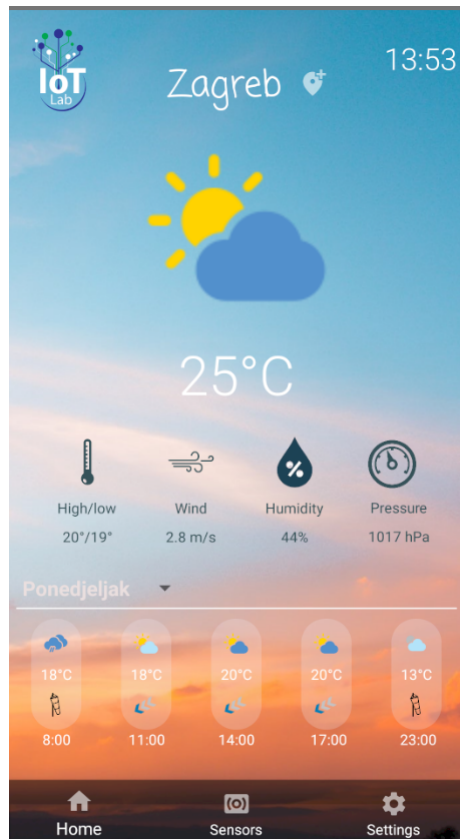


| 16:33 | |  |
|------------------------------|----------------|--|
| Wednesday, February 24, 2021 | | |
| Zagreb | | |
| Status: | Name: | |
| Offline | Weather cast | INFO |
| Online | Weather graphs | INFO |
| Online | Agri culture | INFO |
| Online | Air weather | INFO |
| | | |

Slika 5.5: Sensors page

Ako je korisnik uspješno odradio registraciju ili prijavu na svoj račun, preusmjeren je na sljedeću stranicu koja je zapravo i ključ cijele aplikacije. Riječ je o stranici Senzori na kojoj korisnik može uvidjeti sve raspoložive senzore zajedno s njihovim statusom. Senzor može biti dostupan ili nedostupan. U slučaju da je senzor dostupan korisnik će moći odabrati opciju info u kojoj će imati uvid u sve podatke koje senzor nudi zajedno s grafičkim prikazom. Ako je pak senzor nedostupan ta opcija će biti onemogućena te će korisnik morati odabrati neki drugi dostupni senzor. Ako se slučajno dogodi da su svi senzori nedostupni, priredio sam

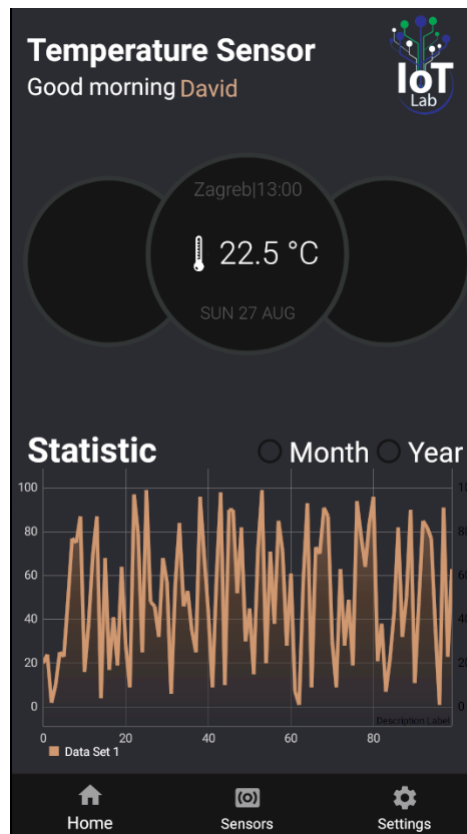
čvor **Vremenska prognoza** koja će uvijek biti raspoloživa za korisnika jer svoje podatke uzima s **Accuweather** te **DHMZ** izvora(više o tome u nastavku).



Slika 5.6: Main page

Prva stranica, ujedno i glavna Vremenska prognoza, prikazuje najnovije vremenske podatke za područje grada Zagreba. Korisnik ima uvid u različite podatke poput trenutne temperature, najviše/najniže temperature u danu, brzine vjetra, vlažnosti i tlaka. Također ima uvid u sedmodnevnu prognozu. Podaci se uzimaju od stranice DHMZ, a povezani su preko XML-a koji je otvoren i može se naći na njihovoj službenoj stranici.

Na dnu stranice nalazi se i prozor s "prečacima" do određenih aktivnosti. Ovdje možemo pronaći Home, Sensors te Settings koji su korisniku onda dostupni s jednim klikom. Ovu funkcionalnost se može pronaći i na sljedećim aktivnostima opisanih dalje.

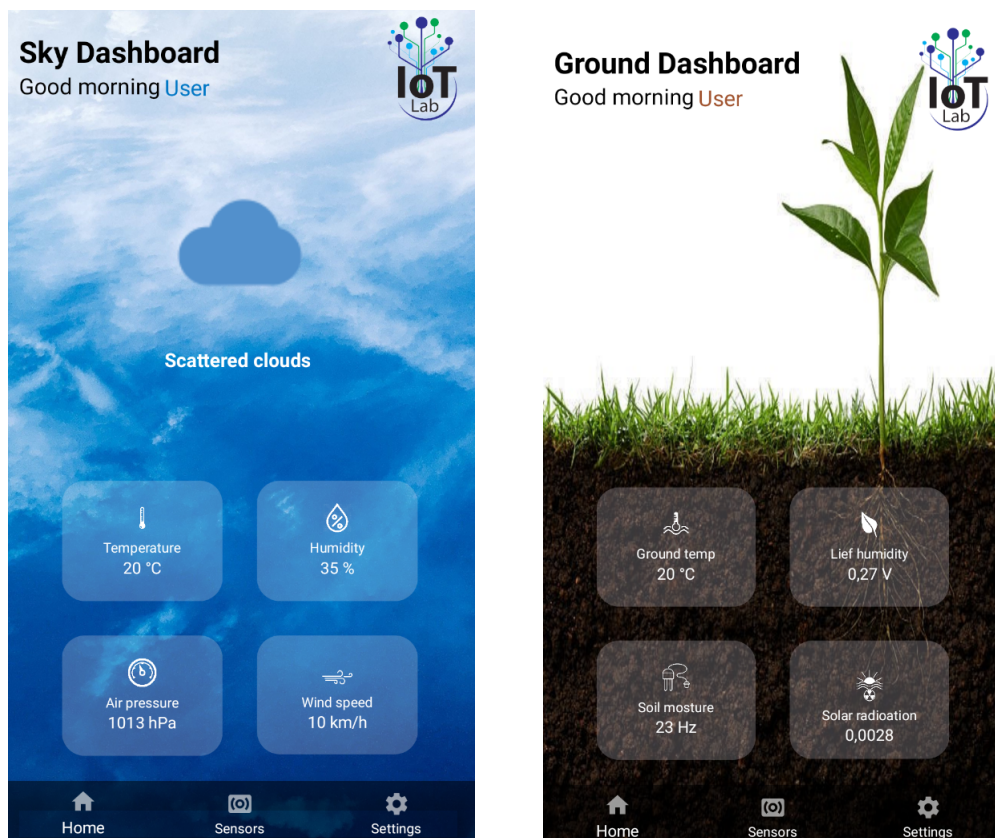


Slika 5.7: Weather graphs

Stranica Grafički prikaz vremenskih podataka, kao što joj i ime govori, pruža grafički uvid u više vrsta vremenskih podataka među kojima su temperatura zraka, vlažnost, tlak, solarna radijacija te količina kiše. Korisnik može odabrati željeni grafički prikaz u odnosu na raspon vremena od tjedan, mjesec ili godinu dana s ciljem da vidi kako se koji od odabranog parametra mijenjao u zadanom rasponu. Osim tih podataka, korisnik ima uvid u trenutnu bateriju senzora zajedno s vremenom u kojem su podaci bili očitani.

Podaci su parsirani iz dobivenog CSV filea, koji je povezan preko REST API sučelja, te se ažuriraju svaki puta kada korisnik odabere nešto drugo na stranici.

Grafovi su složeni tako da korisnik najprije odabere vremenski period te se na temelju toga pošalje POST zahtjev prema bazi podataka, zatim se ti podaci parsiraju spremanjem u mapu u kojoj se nalazi podatak i vrijeme te na kraju se ta mapa sprema u graf. Također se može primijetiti kako i ovdje postoji prozor s prečacima kao i na prethodnoj aktivnosti.

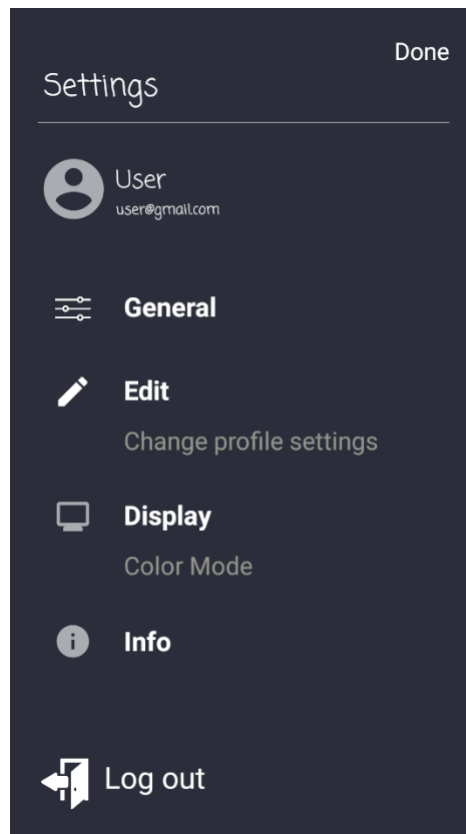


Slika 5.8: Sensor dashboards

Senzori su podijeljeni u dvije osnovne sekcije. Prva sekcija pruža korisniku uvid u trenutne podatke vremena zraka. Dakle to uključuje brzinu vjetra, vlažnost zraka, temperaturu zraka te tlak zraka. Također korisnik ima vizualan logo trenutnog vremena. Drugu sekciju obuhvaćaju podaci vremena na tlu. Dakle to su vlaga lista, temperatura tla, vlaga tla te solarna radijacija.

Podaci su i ovdje parsirani iz dobivene CSV datoteke samo što su sada uzeti samo zadnji podaci. Treba imati u obzir da su podaci vremenski ograničeni senzorom, dakle ne smije se očekivati da će senzor svake sekunde ili minute davati svježije podatke.

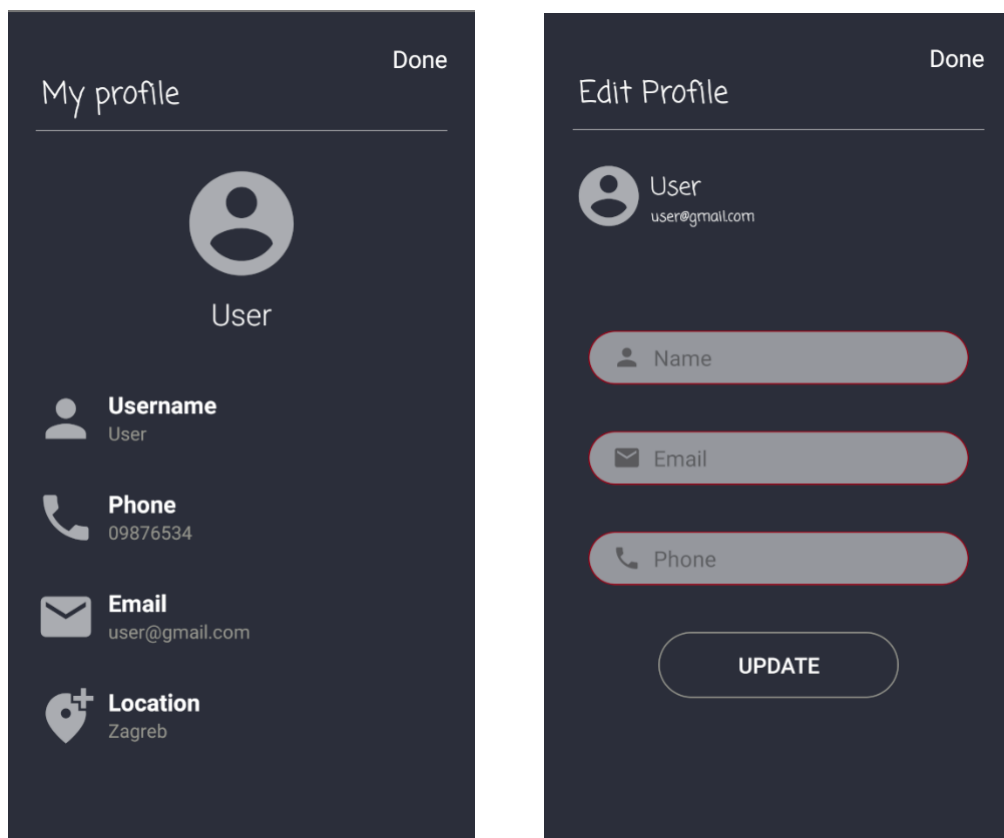
Ovo su također stranice s prozorom s prečacima prema postavkama te senzorima.



Slika 5.9: Settings page

Stranica Settings prikazuje osnovne podatke o korisniku, (link) prema kompletnim podacima o korisniku, (link) prema informacijama o trenutnoj verziji aplikacije, (link) za zamjenu korisničkih podataka te nudi opciju 'log out' iz aplikacije.

Aktivnost je direktno povezana s bazom podataka Firebase jer se na početku stranice moraju prikazati korisničko ime te e-mail adresa.



Slika 5.10: My profile and edit

Stranica *Change profile* nudi korisniku mogućnost promjene svojih podataka poput e-mail adrese, telefona te imena. Stranica *My profile* daje korisniku uvid u njegove podatke uključujući Full name, e-mail adresu, telefona te adrese. Stranica *Info* pruža najnovije vijesti o aplikaciji zajedno s posljednjom verzijom aplikacije i izmjenama koje su se dogodile u njoj.

5.3. Zahtjevi i karakteristike sustava

Android aplikacija dostupna je za sustav Android 6.0(Marshmallow) pa na dalje. Aplikacija je testirana na različitim uređajima te se pokazala dostojna za rad na različitim platformama. Također aplikaciji se može pristupiti i u vodoravnom načinu rada, a u slučaju da je zaslon malen, aktivira se ScrollView koji omogućuje da korisnik "scrolla" po aplikaciji kako bi došao do željenog podatka.

5.4. Moguća poboljšanja

Stvaranje aplikacije je proces koji nikad ne prestaje, ali u jednom trenutku trebamo odrediti minimum koji bi bio idealan za početak korištenja proizvoda. U ovom poglavlju navest će se neke ideje koje bi se mogle nadodati unutar aplikacije kako bi korisnik imao što bolji doživljaj aplikacije te kako bi mu aplikacija bila što korisnija u svakodnevnome životu.

5.4.1. Nadogradnja baze podataka

Trenutno se u bazi podataka nalaze samo osnovni korisnički podaci zajedno s par njegovih preferencija. Cilj je proširiti te preferencije kako bi aplikacija, naprimjer, sama znala koje grafičke podatke korisnik najviše pregledava kako bi dohvat do njih bio što jednostavniji. Također, bilo bi korisno kada bi korisnik mogao ocijeniti senzorski podatak ili dati neku povratnu obavijest koja bi se spremala u bazu kako bi se sustav što više razvio. Baza podataka bi se mogla proširiti tako da uzima podatke koje vraća Rest API sučelje te ih stavlja u svoju tablicu. Tako bi uzimanje podataka s aplikacije prema bazi bilo vrlo preglednije i jednostavnije za korištenje.

5.4.2. Korisničko iskustvo

Ako je mobilna aplikacija jedini način korisničke interakcije sa sustavom, ona mora biti vizualno atraktivna i pristupačna, što je najbolje prepustiti profesionalnom dizajneru prije puštanja aplikacije u uporabu. Kako bi se povećao potencijalni broj korisnika, osim mobilne aplikacije moglo bi se stvoriti i mobilnu Web aplikaciju koja će na sličan način, pa možda i bolji, prikazati sve što i mobilna aplikacija. S Web aplikacijom, grafički prikazati bi mogao biti vrlo detaljniji i ljepši za oko korisnika.

Također, mogao bi se stvoriti i gadget koji će biti prikazan korisniku na početnoj stranici mobitela, a koji bi mogao sadržavati i neke podatke od vremenske prognoze s aplikacije.

5.4.3. Proširenje sustava

Trenutno su senzori postavljeni na lokaciji grada Zagreba, ali definitivno bi se korisniku privukla dodatna pažnja kada bi taj raspon gradova bio veći. Aplikacija bi se u tom slučaju mogla vrlo lako prilagoditi promjenama.

6. Zaključak i budući rad

Zadatak za završni rad bio je razvoj programske podrške na razini klijenta za pokretnog korisnika s operacijskim sustavom Android u kojem će biti prikazani različiti podaci s različitih čvorova. Korisniku također treba biti omogućena mogućnost odabira prikaza željenih senzorskih podataka. Kako bi se najbolje dočarao željeni prikaz, trebalo je proučiti različite modele stvaranja i primjene dostupnih senzorskih podataka.

Zadatak je podijeljen u dva ključna dijela. Prvi dio bio je sami izgled aplikacije, a drugi dio bio je povezivanje servera sa svim grafičkim komponentama.

Najveći tehnički izazov bio je implementacija podataka iz baze podataka prema mojoj aplikaciji te prikaz u različitim grafovima i tablicama, no nakon određenog vremena i taj problem je bio svladan.

Ovaj završni rad zahtijevao je usvajanje tehničkog znanje pisanja u Android Studiju, korištenje baze podataka Firebase, parsiranje kroz CSV i JSON i XML, služenje s različitim Aplikacijsko programskim sučeljima(API) te pisanje stručnoga rada. Najveća prepreka bilo je samo učenje tehnologija potrebnih za ostvarivanje projekta.

LITERATURA

- [1] Iot: top 5 pametnih rješenja u primjeni. 2018. URL <https://www.inspireme.hr/inspire/vijesti/iot-top-5-pametnih-rjesenja-u-primjeni/>.
- [2] *Kako instalirati Telegraf, InfluxDB i Grafana (Tig Stack) na Ubuntu Linux*, 2021. URL <https://hr.admininfo.info/c-mo-instalar-telegraf>.
- [3] Antonio Dujmović. Upravljanje videonadzorom cestovnog prometa mobilnim uređajem na android platformi. 2010.
- [4] Chris Hoffman. What is a csv file, and how do i open it? 2018.
- [5] IBM. Rest api. 2018. URL <https://www.ibm.com/docs/hr/mam/7.6.1?topic=apis-rest-api>.
- [6] Damir Kirasić. Xml tehnologija i primjena u sustavima procesne informatike. 2019.
- [7] Drago Radić. Android operativni sustav. 2019.
- [8] Darko Pavlović Sanja Šijanović Pavlović, Antonijo Bolanča. „internet of things“ i „blockchain“ kao alati razvoja fleksigurnog energetskog sektora. 2018.
- [9] Ajay Sarangam. History of java: An important guide in just 4 points. 2021. URL <https://www.jigsawacademy.com/blogs/java/history-of-java/>.
- [10] Android Studio. The official ide for android, 2017. URL <https://developer.android.com/studio/index.html>.

POKRETNI KLIJENT ZA PRAĆENJE SENZORSKIH OČITANJA

Sažetak

U završnom radu razvijena je Android mobilna aplikacija za pokretnog korisnika s operacijskim sustavom Android u kojem će biti prikazani različiti podaci s različitih čvorova. Korisnik se prijavljuje sa svojim podacima u sustav koji komunicira s bazom podataka, te zatim ima mogućnost odabira različitih senzorskih očitavanja na pregled. Korisniku se također nude određene funkcionalnosti, odnosno njegove preferencije, koje će se zatim realizirati u aplikaciji. Cilj aplikacije je omogućiti svakom korisniku što lakši i brži pronalazak raznih vremenskih podataka u aktualnom razdoblju. Aplikacija se lako instalira na mobitelima u par koraka te je sigurna i dostupna za sve Android uređaje.

Ključne riječi: Android, senzorski podaci, baza podataka, preferencije korisnika..

MOBILE CLIENT FOR MONITORING SENSORY READINGS

Abstract

In the final work, an Android mobile application was developed for a mobile user with the Android operating system in which different data from different nodes will be displayed. The user logs in with his data to the system that communicates with the database, and then has the option of selecting different sensor readings for review. The user is also offered certain functionalities, ie his preferences, which will then be realized in the application. The goal of the application is to enable each user to find various weather data in the current period as easily and quickly as possible. The application is easily installed on mobile phones in a few steps and is safe and available for all Android devices.

Keywords: Android, sensor readings, database, user preferences

POPIS SLIKA

| | |
|---|----|
| 2.1. Sensor dashboard | 3 |
| 2.2. Primjer Iot tehnologije | 4 |
| 3.1. Firebase korisnici prijavljeni u sustav | 7 |
| 4.1. Android studio | 9 |
| 4.2. XML file sa stranice DHMZ | 11 |
| 4.3. Testiranje REST sučelja pomoću programa Postman. | 12 |
| 5.1. Stvaranje novog projekta | 18 |
| 5.2. Struktura projekta aplikacije Weather Forecast | 19 |
| 5.3. Welcome page | 20 |
| 5.4. Register and Login | 20 |
| 5.5. Sensors page | 21 |
| 5.6. Main page | 22 |
| 5.7. Weather graphs | 23 |
| 5.8. Sensor dashboards | 24 |
| 5.9. Settings page | 25 |
| 5.10. My profile and edit | 26 |