Internet aplikacija za udaljeno upravljanje i nadziranje pametnih uređaja

Omazić, Antonio

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

Internet aplikacija za udaljeno upravljanje i nadziranje pametnih uređaja

Završni rad

Antonio Omazić

Osijek 2022

SADRŽAJ

1. UVOD

Cilj ovog završnog rada je napraviti web aplikaciju koja omogućuje korisniku nadzor i upravljanje Arduino mikrokontrolera preko interneta. Arduino IDE nudi mogućnosti spajanja i slanja podataka na internet ali korisnik može pristupiti tim podacima samo ako je spojen na isti internet kao i mikrokontroler i zahtjeva pisanje HTML kôda u samom Arduino IDE programu za što taj program nije namijenjen i dosta je ograničen u tom pogledu. Web aplikacija opisana u ovom završnom radu omogućuje korisniku znatno olakšano postavljanje podataka mikrokontrolera na internet i slanje podataka na mikrokontroler s interneta korištenjem raznih alata na web aplikaciji bez obzira na koji internet su spojeni mikrokontroleri i korisnik.

Web aplikacija radi na način da kada se korisnik registrira na aplikaciji i potvrdi e-poštu, dobije poseban kôd koji upisuje zajedno sa svojim korisničkim imenom, lozinkom, e-poštom, i podatcima za spajanje na internet na predviđeno mjesto u Arduino IDE predlošku koji korisnik može preuzeti s web aplikacije nakon što se prijavi. Predložak još sadrži i ostale upute za korištenje.

Poglavlje opis primijenjene tehnologije opisuje tehnologije korištene u izradi rada. Poglavlje je podijeljeno je u više potpoglavlja koja opisuju svaku tehnologiju detaljnije.

Poglavlje izrada web aplikacije opisuje proces izrade i funkcionalnost web aplikacije izrađene u ovom radu. Poglavlje je podijeljeno u tri potpoglavlja koja opisuju izradu svakog zasebnog dijela web aplikacije detaljno i potpoglavlje koji prikazuje izgled korisničkog sučelja web aplikacije opisane u ovom završnom radu.

Potpoglavlje Firestore opisuje proces izrade Cloude Firestore baze podataka korištene u ovom završnom radu i postavljanje pravila autentikacije za web aplikaciju.

Potpoglavlje kreiranje Spring Boot pozadinske aplikacije detaljno opisuje proces izrade pozadinske aplikacije korištenjem Spring Boot okvira. Kreiranje Angular sučelja potpoglavlje opisuje proces kreiranja sučelja u Angular okviru.

Kreiranje Arduino IDE predloška opisuje Arduino IDE predložak koji sadrži kôd za spajanje Arduino uređaja na web aplikaciju.

Zadnje potpoglavlje prikaz funkcionalnosti i izgleda web aplikacije prikazuje izgled korisničkog sučelja web aplikacije te proces korištenja web aplikacije opisane u ovom radu.

1.1 Zadatak završnog rada

Cilj ovog završnog rada je izrada web aplikacije koja ima mogućnosti registracije, prijave te nadzora i upravljanja mikrokontrolera. Potrebno je izraditi sučelje (engl. *frontend*) što je dio aplikacije koji korisnik vidi i s kojim se korisnik koristi te pozadinski dio (engl. *backend*) koji rukuje s podatcima iz baze podataka i aplikacijsko programsko sučelje (engl. application programming interface, API) s kojim sučelje i pozadinski dio komuniciraju. Potrebno je izraditi bazu podataka te napraviti Arduino IDE predložak koji sadrži kôd za spajanje na bazu podataka i upute za korištenje aplikacije

2. SLIČNA RJEŠENJA

Na tržištu postoje rješenja ,ali većina postojećih aplikacija su mobilne aplikacije koje komuniciraju s mikrokontrolerom preko bluetootha što drastično ograničava udaljenost na kojoj se mogu nadzirati i upravljati.

2.1 Blynk

Primjer aplikacije koja može komunicirati s mikrokontrolerima preko interneta je Blynk. Blynk je platforma za Android i iOS koja omogućava rad s Arduino čipovima i ostalim mikrokontrolerima preko interneta, univerzalne serijske sabirnice (engl. Universal Serial Bus, USB), Bluetootha i Etherneta. Blynk nudi mnoge alate poput gumbova, grafova, i slično. koji omogućavaju korisniku kreiranje zasebnog sučelja za upravljanje ili nadziranje mikrokontrolera [4]. Primjer jedno takvog sučelja je prikazan na slici 2.1. Glavna razlika između Blynk i web aplikacije opisane u ovom radu je to što je Blynk dostupan samo za mobilne uređaje i web aplikacija se isključivo fokusira na Arduino čipove dok Blynk radi s više različitih vrsta.

Slika 2.1 primjer Blynk korisničkog sučelja [1]

2.2 MQTT protokol aplikacije

Drugi primjer sličnog rješenja je mobilna aplikacija MQTT Dash. MQTT Dash omogućuje korisnicima izradu nadzornih ploča za uređaje koje podržavaju MQTT lagani mrežni protokol za komunikaciju između uređaja i interneta. Slično kao i Blynk korisnici mogu na intuitivan način izraditi korisničko sučelje za svoje potrebe ali za razliku MQTT Dash sadrži podršku za skriptiranje korištenjem JavaScript jezika što omogućava izradu kompleksnijih sučelja.

2.3 Mqtt Dashboard - IoT and Node-RED controller

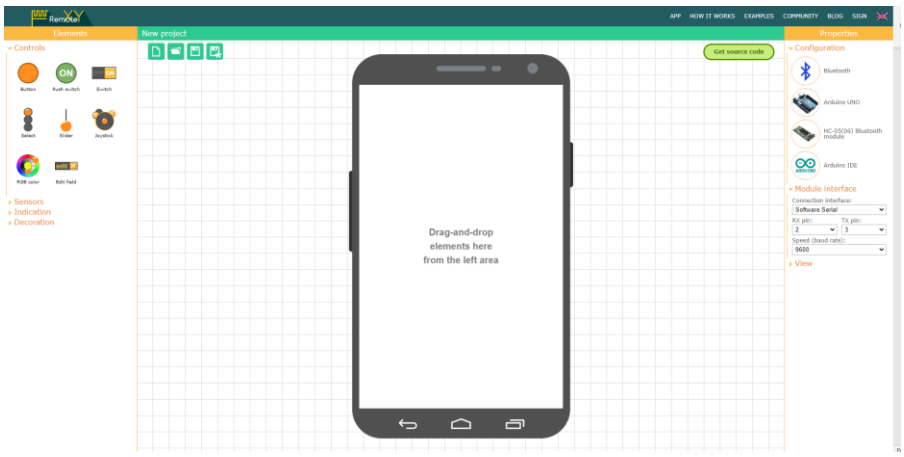
Mqtt Dashboard je jednostavna mobilna aplikacija za kontrolu uređaja s omogućenim MQTT-om i upravljanje sustavom kućne automatizacije. Kompatibilan je s Node-RED, Tasmota Sonoff, i svim internetskim Arduino pločama.

2.4 RaspController

Aplikacija RaspController omogućuje jednostavno daljinsko upravljanje Raspberry Pi uređajima. Omogućuje upravljanje datotekama, kontrolu GPIO priključaka, slanje naredbi izravno putem terminala, pregledavanje slika s povezane kamere i prikupljanje podataka s različitih senzora.

2.5 RemoteXY

Popularna alternativa Blynk aplikaciji je mobilna aplikacija RemoteXY: Arduino control koja poput ostalih navedenih primjera komunicira s uređajem preko interneta i omogućava korisniku kreiranje zasebnog korisničkog sučelja. Ali ono što razlikuje RemoteXY: Arduino control od ostalih rješenja je način kreiranja sučelja. RemoteXY: Arduino control koristi posebnu web aplikaciju prikazana na slici 2.2 koja na intuitivan način omogućuje korisniku izradu kompleksnih sučelja i povezivanje na sam Arduino uređaj.



Slika 2.2 Web aplikacija za izradu sučelja i povezivanje na Arduino uređaj RemoteXY: Arduino control aplikacije [2]

3. OPIS PRIMJENJENE TEHNOLOGIJE

U ovom poglavlju su opisane tehnologije korištene u izradi ovoga završnog rada. Izrada web aplikacije podijeljena je u tri dijela: korisnički dio, pozadinski dio te Arduino IDE dio. Dijelovi su međusobno povezani ali tehnologije korištene u izradi svakog dijela znatno se razlikuju.

3.1 Angular

Angular je platforma i okvir (engl. *framework*[1]) otvorenog kôda (engl. *open source*[2]) koju je razvio Google, izgrađena na temelju TypeScript skriptnog programskog jezika za razvoj klijentskih web aplikacija. Moduli, komponente i servisi čine temelje koncepte Angulara. Temeljni građevni blokovi Angular aplikacija su Angular komponente organizirane u NgModul module. NgModul moduli prikupljaju međusobno povezani kôd u funkcionalne skupove kojim se definira Angular aplikacija. Aplikacija mora imati korijenski modul koji podiže sustav i obično ima mnogo više komponenti značajki.

Angular kreira jednostranične web aplikacije. Takav pristup smanjuje vrijeme potrebno za zamjenu sadržaja u prikazu na stranici zato što preglednik ne mora učitati cijelu web stranicu. Angular jednostranični pristup ostvaruje tako da su u komponentama definirani pogledi (engl. *views*), što su skupovi elemenata zaslona koje Angular bira i mijenja u skladu s logikom programa i podataka. Komponente koriste usluge koje pružaju specifične funkcije koje nisu izravno povezane s pogledima. Pružatelji usluga se uključuju u komponente kao ovisnost (engl. *dependencies*), što čini kôd modularnim, ponovno upotrebljivim i učinkovitim. Obično komponente sadrže više pogleda koji su raspoređeni hijerarhijski. Za definiranje navigacijskih puteva među pogledima koristi se usluga usmjerivača (engl. *router*) [3].

3.2 HTML

U Angular-u za kreiranje HTML (engl. *Hyper Text Markup Language*). dokumenata koristi se standardni opisni jezik HTML. HTML-om je definirana struktura web stranice

[1] *framework* - osnovna konceptualna struktura
[2] *open source* – softwere s javno dostupnim izvornim kôdom

korištenjem predefiniranih oznaka (engl. *tags*) koje preglednik koristi kao smjernice kako bi prikazao sadržaj korisniku. Oznake tvore elemente koji su gradivni blokovi HTML dokumenta [4].

3.3 CSS

Za definiranje izgleda u Angular-u koriste se kaskadni stilovi odnosno CSS (engl. *Cascading Style Sheets*). CSS je opisni jezik za dodavanje i mijenjanje stilova što uključuje mijenjanje: boja, fontova, uređivanje tablica, i slično. Korištenjem CSS-a odvaja se sadržaj od prezentacijskih pravila, čime sav kôd postaje pregledniji i proces mijenjanja izgleda stranice postaje brži. Opći oblik deklaracije CSS stila sastoji se od selektora kojim se određuje na koje sve elemente će se primjenjivati stilsko pravilo i deklaracije kojom se definira stil elementa [5].

3.4 BOOTSRAP

Kako bi se dodatno poboljšao dizajn i olakšao proces dizajniranja web aplikacije postoji Bootstrap. Bootstrap je besplatna biblioteka usmjerena na razvoj responzivnih web sučelja, koja se sastoji od CSS i JavaScript datoteke. JavaScript datoteka je opcionalna i uključuje se za dodavanje funkcionalnosti. Za dodavanje dodatnih funkcionalnosti moguće je uključiti i jQuery biblioteku. [6]

3.5 TypeScript

Angular je izgrađen korištenjem TypeScript-a i svaka Angular komponenta ima TypeScript datoteku, tako za dodavanje funkcionalnosti u Angular web aplikaciji potrebno je koristiti TypeScript programski jezik. TypeScript je JavaScript bazirani programski jezik koji je razvio i održava Microsoft. To je strogi sintaktički nadskup JavaScripta i jeziku dodaje tipove podataka (Number, String, Boolean, Array, Tuple, Enum, Unknown, Any, Void, Null, Undefined, Never, Object), i dodatne funkcionalnosti. TypeScript je objektno orijentirani jezik dok JavaScript nije što čini TypeScript pogodnijim za razvoj za razvoj velikih aplikacija od JavaScripta koji u velikim projektima postaje vrlo kompleksan i nečitljiv. Postojeći JavaScript

programi su valjani TypeScript programi budući da je nadskup JavaScript-a. TypeScript se koristi za razvoj aplikacija koje se izvršavaju na strani poslužitelja i na strani klijenta [7].

3.6 Spring Boot

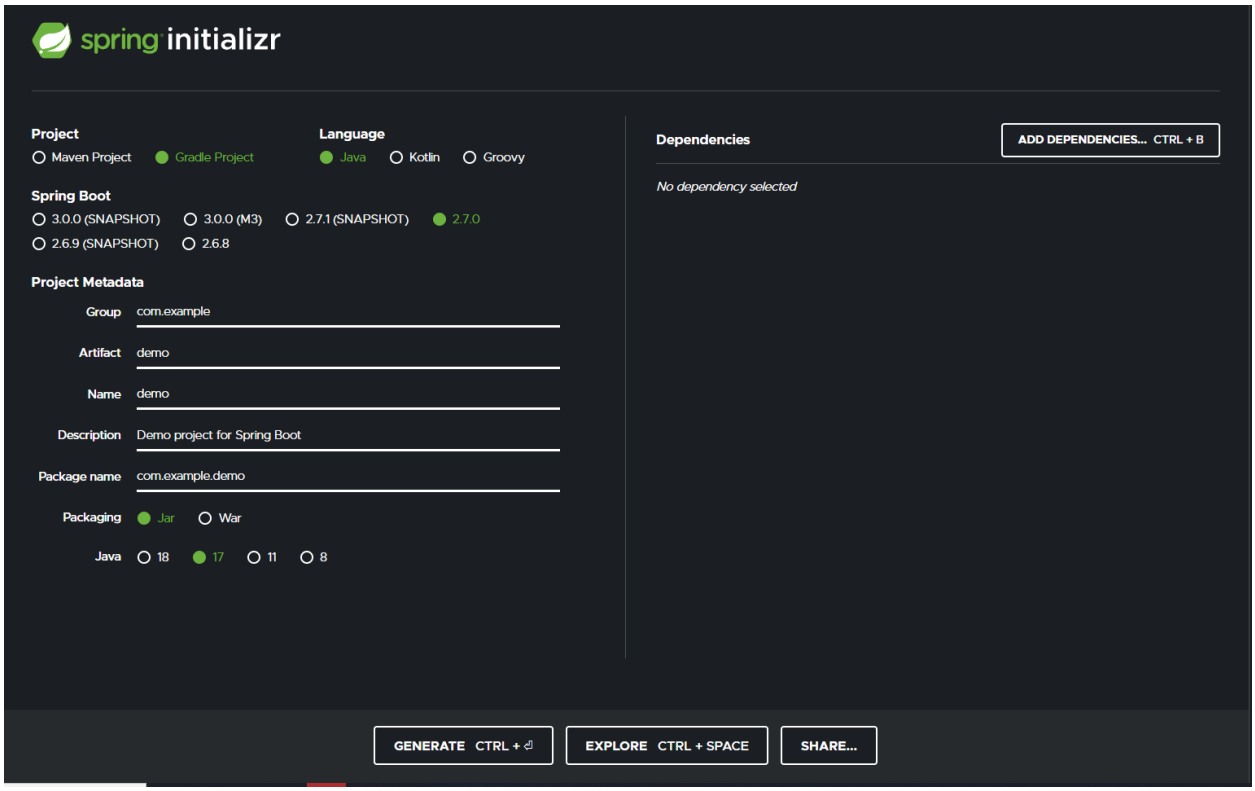
Spring Boot je okvir otvorenog kôda temeljen na Javi koji se koristi za stvaranje mikro usluga[3] (engl. *Micro Service*) [8]. Spring Boot je razvio Pivotal Team.

Značajke Spring boota:

- Kreiranje samostalnih Spring aplikacija
- Ugrađeni Tomcat, Jetty ili Undertow (nema potrebe za postavljanjem WAR datoteka)
- Mogućnost uvođenja „početne“ ovisnosti za pojednostavljenje konfiguracije izrade
- Mogućnost automatskog konfiguriranja Spring i biblioteka neovisnih programera kad god je to moguće
- Značajke spremne za proizvodnju kao što su metrika, provjere zdravlja i eksterna konfiguracija
- Nema generiranja koda i nema zahtjeva za XML konfiguracijom [9]

Za olakšano kreiranje Spring Boot projekta koristi se web stranica Spring Initilizr (Slika 3.1) koja generira sve odabrane ovisnosti i sve potrebne datoteke.

[3] *Mikro usluge (engl. Micro service)* je arhitektura koja omogućuje programerima samostalno razvijanje i implementiranje usluga.



Slika 3.1 Spring Initilizr [10]

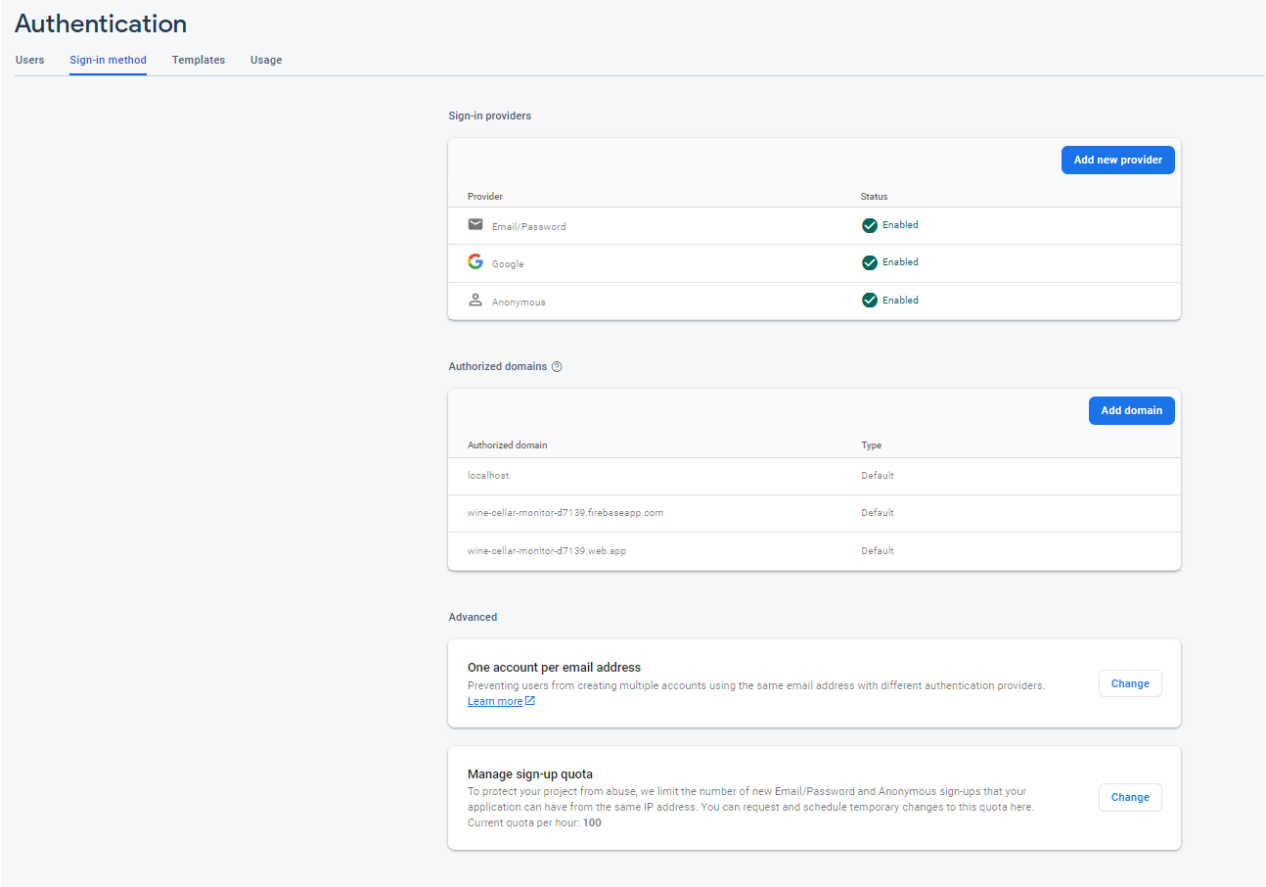
Spring boot aplikacija čini pozadinski dio web aplikacije opisane u završnom radu. Zadaća Spring boot dijela web aplikacije je rukovanje podacima iz baze podataka tako da ovisno o zahtjevu koji dolazi s korisničkog sučelja šalje podataka na sučelje ili šalje primljene podatke s sučelja na bazu podataka.

3.7 Java

Kao što je navedeno za kreiranje Spring boot aplikacije koristi se Java programski jezik, što je objektno orijentirani jezik visoke razine, temeljen na klasama, napravljen na način da sadrži što manje ovisnosti o implementaciji. Java je programski jezik opće namjene sa svojstvom da nema potrebe za ponovnim prevođenjem kako bi se Java aplikacija mogla izvoditi na drugim platformama koje podržavaju Javu. Java aplikacije su obično prevedene u bajt kôd koji ima mogućnost izvođenja na bilo kojem Java virtualnom stroju (JVM) neovisno o arhitekturi računala. Java sintaksa sliči programskim jezicima obitelji C s tim da sadrži manje niskorazinskih objekata od njih. Vrijeme izvršavanja Jave (engl. *runtime*) pruža dinamičke mogućnosti (kao što je refleksija i modifikacija koda izvođenja) koje obično nisu dostupne u tradicionalnim prevedenim jezicima [11].

3.8 Firebase

Firebase je platforma za izradu web i mobilnih aplikacija pod vlasništvom Googlea. Firebase nudi mnoge alate i usluge koje pomažu pri razvoju i održavanju aplikacija. Dvije glavne usluge koje su korištene u izradi ovog rada su Authentication i Cloud Firestore. Authentication se koristi za provjeru prava pristupa korisnicima koji su se registrirali na web aplikaciju i nudi alate za upravljanje registriranim korisnicima (brisanje, resetiranje lozinke i onemogućavanje računa), mijenjanje načina prijavljivanja, mijenjane dopuštenih domena i još razne druge (slika 3.2). Authentication sadrži kôd potreban za jednostavnu implementaciju sustava registriranja i prijavljivanja za Angular. Cloud Firestore je baza podataka dokumenata NoSQL (engl. *No Structured Query Language*) tipa koja daje mogućnosti jednostavnog pohranjivanja, sinkronizaciju i upita podataka za web i mobilne aplikacije, te služi kao glavna baza podataka za ovaj projekt [12].



Slika 3.2 Firebase Authentication

3.9 Arduino IDE

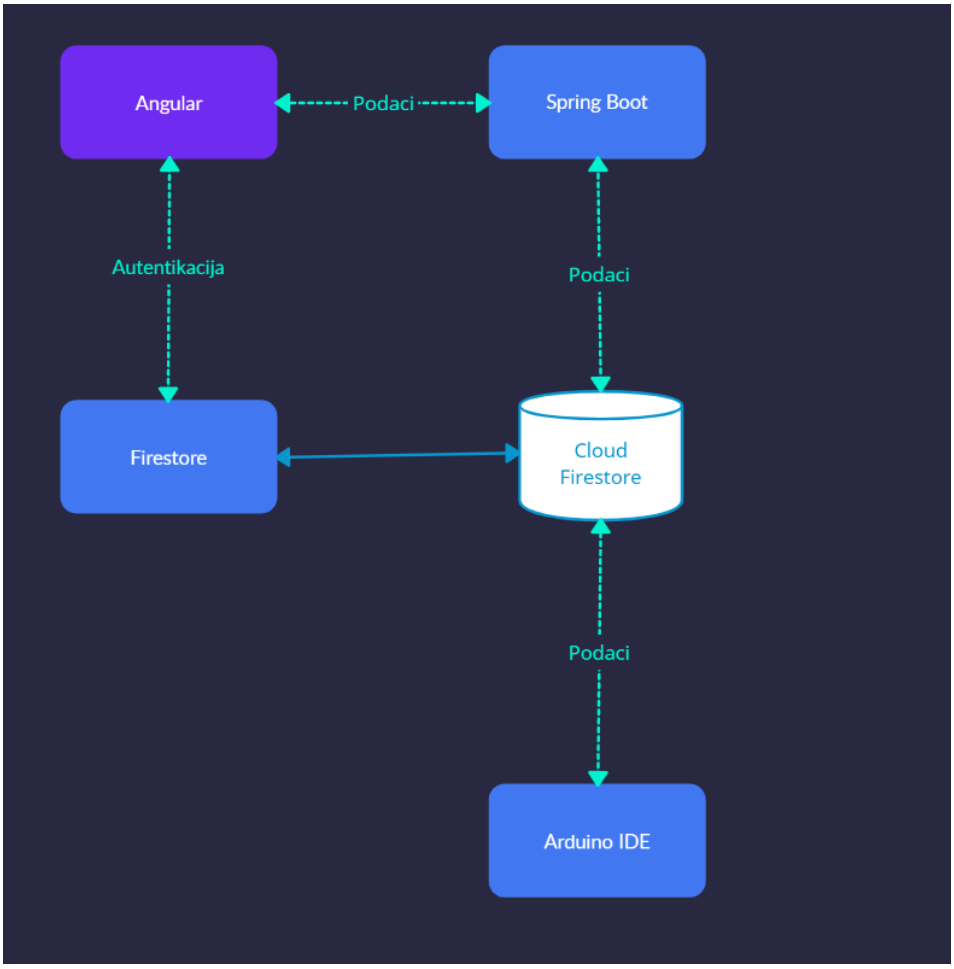
Integrirano razvojno okruženje Arduino (engl. *Arduino Integrated Development Environment*) je okruženje otvorenog kôda za razvoj Arduino aplikacija. Arduino IDE sadrži uređivač teksta za pisanje kôda, konzolu, područje za poruke, alatnu traku s gumbovima za uobičajene funkcije i niz izbornika. Koristi se za povezivanje s hardverom preko univerzalne serijske sabirnice kako bi učitao programe na Arduino čip i komunicirao s njima. Arduino IDE koristi C kao programski jezik [13].

3.10 REST

Prijenos reprezentativnog stanja (engl. *Representational state transfer* REST) je stil softverske arhitekture koji opisuje jedinstveno sučelje između fizički odvojenih komponenti. REST definira četiri ograničenja sučelja: identifikacija resursa, manipulacija resursima, samoopisne poruke i hipermedija kao pokretač stanja primjene. REST se koristi za stvaranje pouzdanih web API-ja. Web API koji poštuje REST ograničenja neformalno je opisan kao RESTful. RESTful web API djelomično se temelji na HTTP (engl. *Hypertext Transfer Protocol*) zahtjevima za pristup resursima putem URL (*engl. Uniform Resource Locator*) kodiranih parametara. Odgovori na HTTP zahtjeve su obično formatirani u JSON (engl. *JavaScript Object Notation*) ili XML (engl. *EXtensible Markup Language*) format za prijenos podataka [14].

4. IZRADA WEB APLIKACIJE

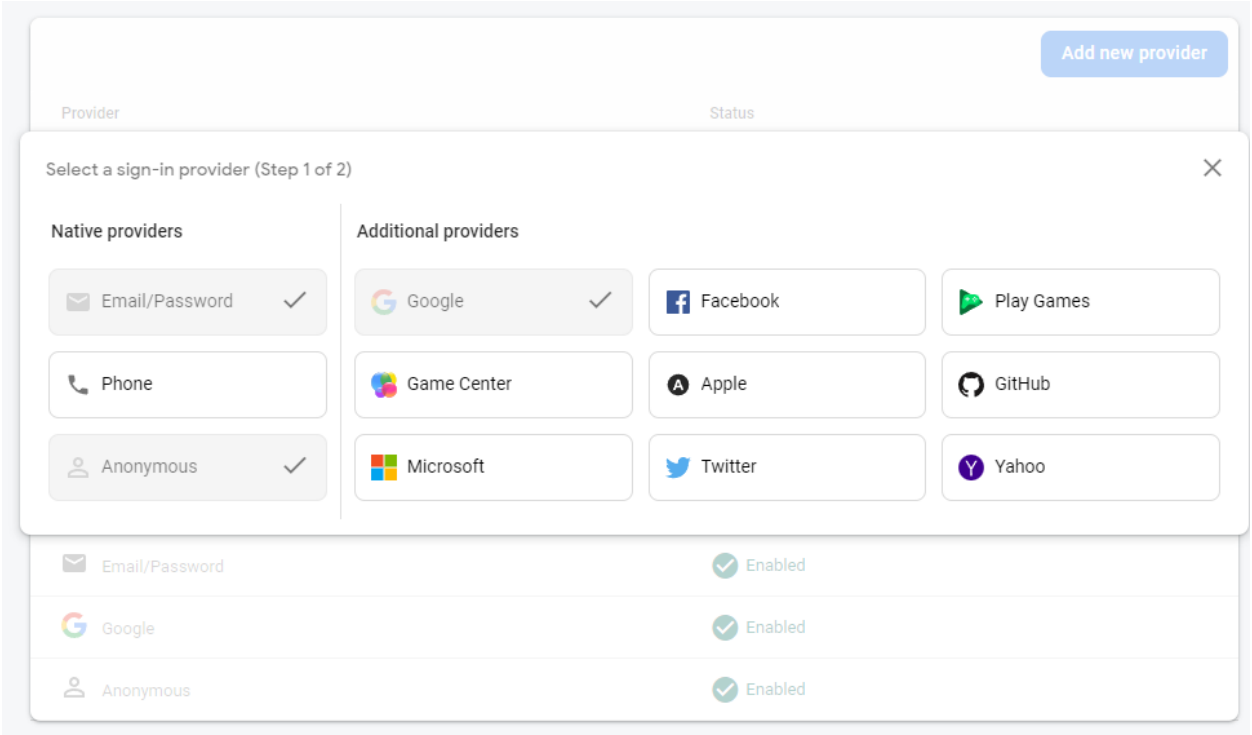
U ovom poglavlju je opisan proces izrade web aplikacije korištenjem navedenih tehnologija. Sam proces izrade je podijeljen u četiri dijela gdje svaki dio koristi različite tehnologije i ima različitu ulogu. Osnovna struktura aplikacije prikazana na slici 4.1 prikazuje svaki pojedini dio aplikacije i njegove veze s ostalim dijelovima. Prikazuje kako Firestore povezuje sve dijelove u jednu cjelinu. Pozadinsku aplikaciju čini Spring Boot aplikacija koji prima podatke s Cloud Firestora i dalje ih po potrebi korisnika šalje na Angular koji čini sučelje, ali isto tako može i primati nove vrijednosti podataka s sučelja te im ažurirati vrijednost na Cloud Firestore bazi podataka. Arduino IDE je isto izravno povezan s Cloud Firestore bazom podataka tako da ima mogućnosti slanja novih podatka i primanja podataka s baze podataka. Angular je izravno povezan s Firestore autentikacijom kako bi Firestore mogao upravljati registracijom i prijavom korisnika na aplikaciju.



Slika 4.1 Osnovna struktura web aplikacije za upravljanje i nadzor mikrokontrolera

4.1 Firestore

Proces izrade Firestore projekta i Cloud Firebase baze podataka vrlo je jednostavan. Potrebno se prijaviti ili registrirati na stranicu te nakon uspješne prijave kliknuti na Go to console. U konzoli potrebno je kliknuti na add project te ispuniti podatke za projekt. Autentikacija je već automatski napravljena i kako bi se omogućila prijava na web aplikaciju potrebno je otići na Sign-in method i postaviti postavke kao na slici 4.2.



Slika 4.2 Postavke prijave web aplikacije

Za kreiranje Cloud Firestore baze podataka potrebno je samo kliknuti na create u Firestore Database-u i postaviti lokaciju servera koja je u ovoj aplikaciji eur3 (europe-west).

4.2 Kreiranje Spring Boot pozadinske aplikacije

Idući korak u izradi web aplikacije je napraviti pozadinsku aplikaciju koja razmjenjuje podatke s bazom podataka i sučeljem. Tehnologija koja se koristiti za izradu pozadinske aplikacije je Spring Boot napisan u Java programskom jeziku. Prvi korak u izradi Spring Boot

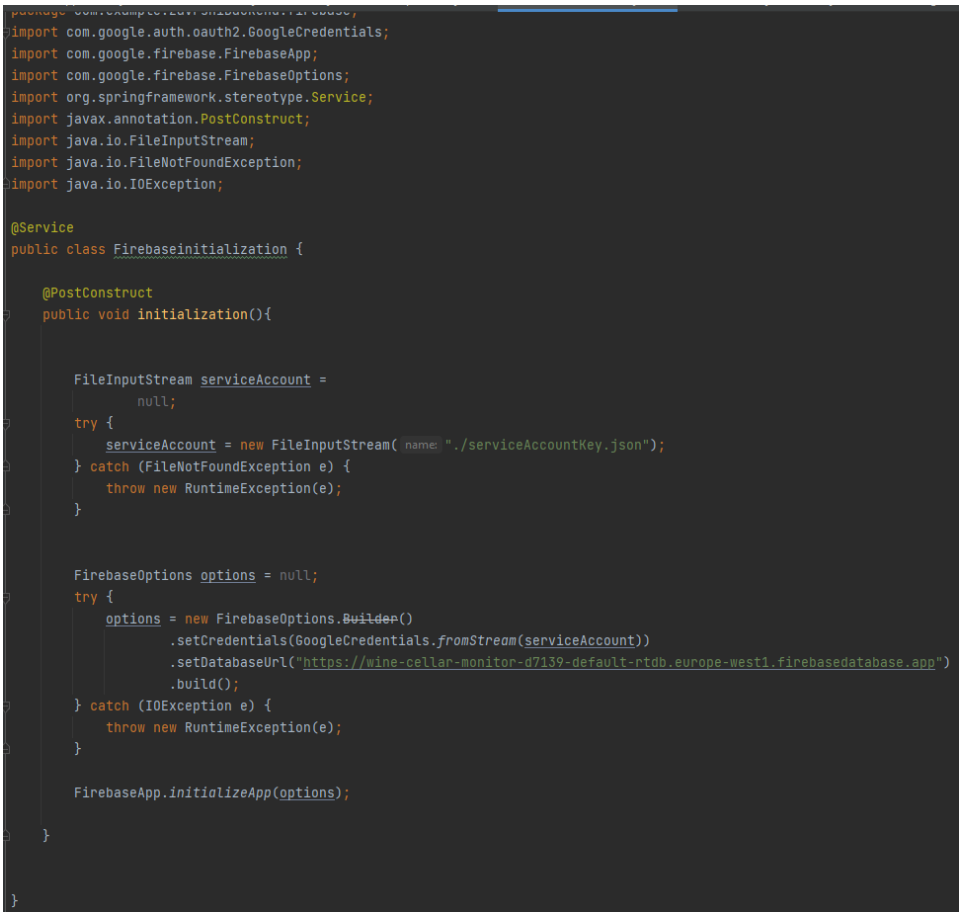
aplikacije je generiranje datoteka projekta koristeći uslugu Spring Initializr. Opcije projekta postavljenje na Gradle Project, Java Language i Spring Web kao dodana ovisnost.

Nakon generiranja datoteka projekt se otvara koristeći InteliJ integrirano razvojno okruženje. Kako bi se povezala Spring Boot aplikacija i Cloud Firestore baza podataka potrebno je dodati novu ovisnost u bild.gradle datoteku (Odsječak programskog kôda 4.3)

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    implementation 'com.google.firebase:firebase-admin:8.1.0'
}
```

Odsječak programskog kôda 4.3 Dodavanje nove ovisnosti potrebne za Firestore

Dodavanje te ovisnosti dodaje sve funkcije i klase potrebne za rad s Cloud Firestore bazom podataka u projekt. Kako bi se Spring Boot spojio s bazom podataka potrebno je napraviti novu „@Service“ anotiranu Java klasu Firebaseinitialization u novonastalom paketu Firestore (Odsječak programskog kôda 4.4). Za uspješno inicijaliziranje baze podataka Spring Boot-u su potrebni Url baze i datoteka konfiguracije baze podataka koju je potrebno preuzeti s Firestore stranice projekta. Klasa Firebaseitialization sadrži jednu metodu initialization koja je anotirana s „@PostConstruct“ kako bi se izvršila nakon injekcije ovisnosti. Uloga metode je spajanje baze podataka i Spring boot aplikacije.

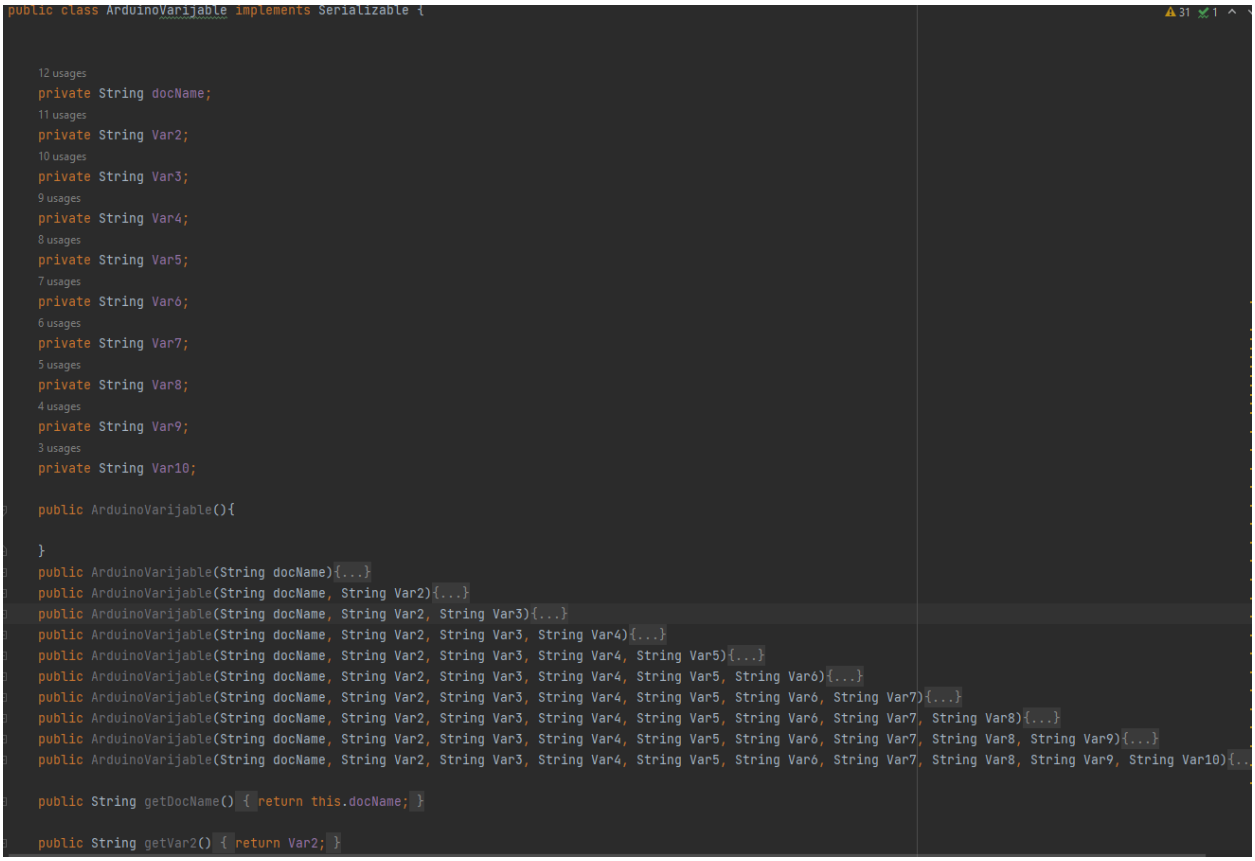


```
import com.google.auth.oauth2.GoogleCredentials;
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseOptions;
import org.springframework.stereotype.Service;
import javax.annotation.PostConstruct;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

@Service
public class Firebaseinitialization {

    @PostConstruct
    public void initialization(){


        FileInputStream serviceAccount =
                null;
        try {
            serviceAccount = new FileInputStream( name: "./serviceAccountKey.json");
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }


        FirebaseOptions options = null;
        try {
            options = new FirebaseOptions.Builder()
                    .setCredentials(GoogleCredentials.fromStream(serviceAccount))
                    .setDatabaseUrl("https://wine-cellar-monitor-d7139-default-rtdb.europe-west1.firebasedatabase.app")
                    .build();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        FirebaseApp.initializeApp(options);

    }


}
```

Odsječak programskog kôda 4.4 Kôd klase FirebaseInitalization

Idući korak je napraviti model paket koji sadržava sve potrebne model klase. Model klase Spring Boot koristi za pretvaranje dolazećih JSON[4] ili ostalih vrijednosti u objekt model klase s kojima se dalje može lagano rukovati. Prva potrebna model klasa je ArduinoVarijable (Odsječak programskog kôda 4.5) koja služi kao model klasa za podatke koje pristižu s baze podataka.

[4] „JSON (engl. JavaScript Object Notation) je otvoreni standardni format datoteke i format za razmjenu podataka koji koristi ljudski čitljiv tekst za pohranu i prijenos podataka objekata koji se sastoje od parova atribut-vrijednost“ [14].



Odsječak programskog kôda 4.5 Model klasa ArduinoVarijable

Klasa sadrži svojstva koja predstavljaju sva polja dokumenata baze podataka i pošto svaki korisnik neće koristiti isti broj varijabla u klasi je potrebno napraviti konstruktor za sve slučajeve. Klasa sadrži još get i set metode svih svojstava kako bi mogli dohvatiti i mijenjati vrijednost varijabla izvan klase. Druga potrebna model klasa je UpdateVar (Odsječak programskog kôda 4.6) koja služi kao model klasa za JSON koji pristiže s sučelja za promjenu vrijednosti polja određenog dokumenta.

```java
public class UpdateVar {
    3 usages
    private  String docName;
    3 usages
    private  String updatetarget;
    3 usages
    private  Integer updateValue;

    public UpdateVar(){}
    public UpdateVar(String docName, String updatetarget, Integer updateValue){
        this.docName = docName;
        this.updatetarget = updatetarget;
        this.updateValue = updateValue;
    }

    1 usage
    public String getDocName() { return docName; }

    1 usage
    public String getUpdatetarget() { return updatetarget; }

    1 usage
    public Integer getUpdateValue() { return updateValue; }

    public void setDocName(String docName) { this.docName = docName; }

    public void setUpdatetarget(String updatetarget) { this.updatetarget = updatetarget; }

    public void setUpdateValue(Integer updateValue) { this.updateValue = updateValue; }
}
```

Odsječak programskog kôda 4.6 Model klasa UpdateVar

Sljedeći korak je napraviti klasu servisa ArduinoVarijablaService u novo napravljenom paketu service (Odsječak programskog kôda 4.7) koja sadrži sve metode koje REST kontroleri pozivaju. Klasa je anotirana s „@Service“ i sadrži dvije metode: getArduinoVarijable i updateArduinoVarijablu. Uloga getArduinoVarijable metode je dohvaćanje svih polja dokumenta s proslijeđenim nazivom te pretvaranje primljenih podataka iz baze podataka u objekt model klase ArduinoVarijable. Metoda vraća instancu klase ArduinoVarijable. Metoda updateArduinoVarijablu prima ime dokumenta, naziv polja dokumenta i novu vrijednost polja. Koristeći te podatke metoda ažurira vrijednost danog polja na Cloud Firestore bazi podataka.

```java
@Service
public class ArduinoVarijableService {


    1 usage
    public ArduinoVarijable getArduinoVarijable(String name) throws ExecutionException, InterruptedException {
    Firestore dbFirestore = FirestoreClient.getFirestore();
    DocumentReference documentReference = dbFirestore.collection( path: "ArduinoVarijable").document(name);
    ApiFuture<DocumentSnapshot> future=documentReference.get();
    DocumentSnapshot document=future.get();
    ArduinoVarijable arduinoVarijable = null;
    if(document.exists()){

        arduinoVarijable = document.toObject(ArduinoVarijable.class);
        return arduinoVarijable;
    }
    else {
        return null;
    }
}


        1 usage
        public String updateArduinoVarijablu(String varname, String varvalue, String docname) throws ExecutionException, InterruptedException {
Firestore dbFirestore = FirestoreClient.getFirestore();
ApiFuture<WriteResult> collectionApiFuture = dbFirestore.collection( path: "ArduinoVarijable").document(docname).update(varname, varvalue);
return collectionApiFuture.get().getUpdateTime().toString();

        }

}
```

Odsječak programskog kôda 4.7 Servisna klasa ArduinoVarijablaService

Zadnji korak izrade Spring Boot pozadniske aplikacije je definiranje krajnje točke URL-a za svaku metodu iz klase servisa kako bi ih sučelje moglo pozivati. Taj proces se odrađuje putem REST kontrolera. Potrebno kreirati klasu ArduinoVarijableResource (Odsječak programskog kôda 4.8) koja je anotirana s „@RestController“ i „@RequestMapping“ što je anotacija koja predstavlja prvi dio krajnje točke URL-a. Kako bi se mogle koristiti metode klase servisa potrebno je implementirati klasu ArduinoVarijablaService. Anotacija „@GetMapping“ mapira HTTP GET zahtjev na getArduinoVarijable metodu s time da anotacija „@PathVariable“ označava da je potreban dodatni parametar u krajnjoj točki koji se navodi poslije „get/“ u URL-u. Anotacija „@PutMapping“ mapira PUT HTTP zahtjev na metodu updateArduinoVarijablu s time da anotacija „@RequestBody“ zahtjeva da parametar metode treba biti vezan za tijelo web zahtjeva koje se prosljeđuje kroz HttpMessageConverter.

```java
@RestController
@RequestMapping("/ArduinoVarijable")
public class ArduinoVarijableResource {
3 usages
@Autowired
    private final ArduinoVarijableService arduinoVarijableService;
    public ArduinoVarijableResource(ArduinoVarijableService arduinoVarijableService) {
        this.arduinoVarijableService = arduinoVarijableService;
    }

@GetMapping("/get/{name}")
    public ArduinoVarijable getArduinoVarijable(@PathVariable String name) throws ExecutionException, InterruptedException {
return arduinoVarijableService.getArduinoVarijable(name);

}

@PutMapping("/update")
    public String updateArduinoVarijable(@RequestBody UpdateVar updatevar) throws ExecutionException, InterruptedException {
        return arduinoVarijableService.updateArduinoVarijablu(updatevar.getUpdatetarget(), updatevar.getUpdateValue().toString(), updatevar.getDocName());

}

}
```

Odsječak programskog kôda 4.8 Klasa ArduinoVarijableResource

4.3 Kreiranje Angular sučelja

Angular aplikacija predstavlja korisničko sučelje. Za kreiranje Angular aplikacije i svih potrebnih datoteka potrebno je napisati komandu ng new u Windows Command Prompt-u. Nakon uspješnog generiranja datoteka, projekt se može otvoriti u bilo kojem uređivaču kôda koji će za ovaj završni rad biti Visual Studio Code.

Za sustav autentikacije aplikacija koristi Firestore što znači da je potrebno dodati Firestore pakete u projekt kako bi se omogućio rad s Firestore alatima. Paketi se dodaju u Angular upisivanjem komande npm install firebase @angular/fire u Windows Command Prompt. Kako bi se Angular spojio na Firebase projekt, potrebno je kao i u Spring Boot-u dodati konfiguracijske podatke Firestore projekta u environment.ts datoteku (Odsječak programskog kôda 4.9)

```
export const environment = {
  production: false,
  firebase: {
    apiKey: "AIzaSyDsSCZYlKAseOragmEPxZb0SJ_0vsCuYOE",
    authDomain: "wine-cellar-monitor-d7139.firebaseapp.com",
    databaseURL: "https://wine-cellar-monitor-d7139-default-rtdb.europe-west1.firebasedatabase.app",
    projectId: "wine-cellar-monitor-d7139",
    storageBucket: "wine-cellar-monitor-d7139.appspot.com",
    messagingSenderId: "700911110993",
    appId: "1:700911110993:web:0cb62c05afa98268f0a170",
    measurementId: "G-D0EQ263K70"
  }
};
```

Odsječak programskog kôda 4.9 Konfiguracijski podatci Firestore projekta u enviroment.ts datoteci

Kako bi se mogao kreirati potpuni sustav autentikacije potrebno je generirati Angular komponente za : prijavu (engl. *sign-in*), registraciju (engl. *sign-up*), zaboravljenu lozinku (engl. *forgot-password*) i potvrdu e-pošte (engl. *verify-email*). Sve komponente se generiraju komandom ng g ime komponente. Za definiranje navigacijskih putova između novonastalih komponenti napravljena je datoteka app-routing.module.ts (Odsječak programskog kôda 4.10)

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { SignInComponent } from './components/sign-in/sign-in.component';
import { SignUpComponent } from './components/sign-up/sign-up.component';
import { DashboardComponent } from './components/dashboard/dashboard.component';
import { ForgotPasswordComponent } from './components/forgot-password/forgot-password.component';
import { VerifyEmailComponent } from './components/verify-email/verify-email.component';
import { AuthGuard } from './shared/guard/auth.guard';
const routes: Routes = [
  { path: '', redirectTo: '/sign-in', pathMatch: 'full' },
  { path: 'sign-in', component: SignInComponent },
  { path: 'register-user', component: SignUpComponent },
  { path: 'dashboard', component: DashboardComponent, canActivate: [AuthGuard] },
  { path: 'forgot-password', component: ForgotPasswordComponent },
  { path: 'verify-email-address', component: VerifyEmailComponent },
];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
export class AppRoutingModule {}
```

Odsječak programskog kôda 4.10 Kôd za usmjerivanje u datoteci app-routing.module.ts

Klasa sučelja user služi kao shema za objekt User (Odsječak programskog v 4.11). Klasa sučelja omogućuje pristup podacima korisnika što omogućuje njihovo korištenje u HTML komponenti.

```
1  export interface User {
2      uid: string;
3      email: string;
4      displayName: string;
5      photoURL: string;
6      emailVerified: boolean;
7  }
8
```

Odsječak programskog kôda 4.11 Klasa sučelja user

U Angular aplikaciji svaka komponenta se sastoji od HTML i TypeScript kôda tako za kreiranje uspješnog sustava autentikacije potrebno je kreirati TypeScript datoteku koja sadrži svu logiku za takav sustav i čije funkcije pozivaju HTML datoteke time i korisnik. U tu svrhu kreirana je datoteka auth.service.ts komandom ng g s shared/services/auth. Funkcija koja počinje na liniji 21 i završava na liniji 31 sprema korisnika lokalno ili ga postavlja na null ako korisnik ne postoji. Funkcija SignIn (Slika 4.12 linija 32) poziva Firebase funkciju za prijavu (Slika 4.12 linija 34) i u slučaju uspješne prijave Angular usmjerivač preusmjerava korisnika na dashboard komponentu. Funkcija Sign-up (Slika 4.12 linija 45) poziva Firebase funkciju za registraciju (Slika 4.12 linija 47) i u slučaju uspješne registracije poziva funkciju SendVerificationMail (Slika 4.13 Linija 56) koja šalje e-poštu potvrde. Funkcija ForgotPassword (Slika 4.13 linija 63) poziva Firebase funkciju (Slika 4.13 linija 65) za slanje e-pošte za resetiranje lozinke isLoggedIn (Slika 4.13 linija 74). Funkcuha postavlja vrijednost emailVerified na true u slučaju da je korisnik potvrdio e-mail. GoogleAuth (Slika 4.13 linija 78) funkcija daje mogućnost prijave Google računom. SetUserData (Slika 4.14 linija 99) funkcija postavlja vrijednosti objekta sučelja user što omogućuje korištenje tih vrijednosti u HTML komponenti. SignOut funkcija (Slika 4.14 linija 114) briše lokalno spremljenog korisnika i usmjeruje korisnika na komponentu prijave.

```
export class AuthService {
  userData: any;
  constructor(
    public afs: AngularFirestore,
    public afAuth: AngularFireAuth,
    public router: Router,
    public ngZone: NgZone
  ) {
    this.afAuth.authState.subscribe((user) => {
      if (user) {
        this.userData = user;
        localStorage.setItem('user', JSON.stringify(this.userData));
        JSON.parse(localStorage.getItem('user')!);
      } else {
        localStorage.setItem('user', 'null');
        JSON.parse(localStorage.getItem('user')!);
      }
    });
  }
  SignIn(email: string, password: string) {
    return this.afAuth
      .signInWithEmailAndPassword(email, password)
      .then((result) => {
        this.ngZone.run(() => {
          this.router.navigate(['dashboard']);
        });
        this.SetUserData(result.user);
      })
      .catch((error) => {
        window.alert(error.message);
      });
  }
  SignUp(email: string, password: string) {
    return this.afAuth
      .createUserWithEmailAndPassword(email, password)
      .then((result) => {
        this.SendVerificationMail();
        this.SetUserData(result.user);
      })
      .catch((error) => {
        window.alert(error.message);
      });
```

Slika 4.12 Odsječak programskog kôda datoteke auth.servise.ts (prvi dio)

```
SendVerificationMail() {
  return this.afAuth.currentUser
    .then((u: any) => u.sendEmailVerification())
    .then(() => {
      this.router.navigate(['verify-email-address']);
    });
}
ForgotPassword(passwordResetEmail: string) {
  return this.afAuth
    .sendPasswordResetEmail(passwordResetEmail)
    .then(() => {
      window.alert('Password reset email sent, check your inbox.');
    })
    .catch((error) => {
      window.alert(error);
    });
}

get isLoggedIn(): boolean {
  const user = JSON.parse(localStorage.getItem('user')!);
  return user !== null && user.emailVerified !== false ? true : false;
}
GoogleAuth() {
  return this.AuthLogin(new auth.GoogleAuthProvider()).then((res: any) => {
    if (res) {
      this.router.navigate(['dashboard']);
    }
  });
}
AuthLogin(provider: any) {
  return this.afAuth
    .signInWithPopup(provider)
    .then((result) => {
      this.ngZone.run(() => {
        this.router.navigate(['dashboard']);
      });
      this.SetUserData(result.user);
    })
    .catch((error) => {
      window.alert(error);
    });
}
```

Slika 4.13 Odsječak programskog kôda datoteke auth.servise.ts (drugi dio)

```

  SetUserData(user: any) {
    const userRef: AngularFirestoreDocument<any> = this.afs.doc(
      `users/${user.uid}`
    );
    const userData: User = {
      uid: user.uid,
      email: user.email,
      displayName: user.displayName,
      photoURL: user.photoURL,
      emailVerified: user.emailVerified,
    };
    return userRef.set(userData, {
      merge: true,
    });
  }
  SignOut() {
    return this.afAuth.signOut().then(() => {
      localStorage.removeItem('user');
      this.router.navigate(['sign-in']);
    });
  }
}
```

Slika 4.14 Odsječak programskog kôda datoteke auth.servise.ts (treći dio)

Sljedeći korak kreiranja sustava autentikacije je napraviti objekt tipa auth.service (Odsječak programskog kôda 4.15) u TypeScript datotekama svih komponenti, što omogućuje pozivanje tih funkcija u HTML komponentama.

```
constructor(
  public authService: AuthService
) { }
ngOnInit() { }
```

Odsječak programskog kôda 4.15 kreiranje objekta tipa auth.service

Zadnji korak je pisanje HTML i CSS kôda (Odsječak programskog kôda 4.16, 4.17, 4.18, 4.19) za uređivanje izgleda svake komponente. Dodavanje „(click)“ atributa HTML oznakama Angular omogućuje pozivanje funkcije kada korisnik pritisne na HTML element s tim atributom. Za dodatno poboljšanje izgleda i za ostvarivanje responzivnosti svaka komponenta sadrži bootstrap klase.

```
<section class="vh-100 bg-image"
  style="background-image: url('https://images.pexels.com/photos/270360/pexels-photo-270360.jpeg');">
  <div class="mask d-flex align-items-center h-100 gradient-custom-3">
    <div class="container h-100">
      <div class="row d-flex justify-content-center align-items-center h-100">
        <div class="col-12 col-md-9 col-lg-7 col-xl-6">
          <div class="card" style="border-radius: 15px;">
            <div class="card-body p-5">
              <h2 class="text-uppercase text-center mb-5">Sign in</h2>
              <form>
                <div class="form-outline mb-4">
                  <input type="email" id="form3Example3cg" class="form-control form-control-lg" placeholder="Email" #userName />
                  <label class="form-label" for="form3Example3cg">Your Email</label>
                </div>
                <div class="form-outline mb-4">
                  <input type="password" id="form3Example4cg" class="form-control form-control-lg" placeholder="Password" #userPassword required />
                  <label class="form-label" for="form3Example4cg">Password</label>
                </div>
                <div class="d-flex justify-content-center">
                  <button type="button"
                    class="btn btn-block btn-lg gradient-custom-4 text-body" (click)="authService.SignIn(userName.value, userPassword.value)">Log in</button>
                  <button type="button" class="btn" (click)="authService.GoogleAuth()">
                    <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor" class="bi bi-google" viewBox="0 0 16 16">
                      <path d="M15.545 6.558a9.42 9.42 0 0 1 .139 1.626c0 2.434-.87 4.492-2.384 5.885h.002C11.978 15.292 10.158 16 8 16A8 8 0 1 1 8 0a7.689 7.689 0 0 1 5.352 2.082l-2
                    </svg>
                    Log in with Google
                  </button>
                </div>
                <a class="text-center text-muted mt-5 mb-0" routerLink="/forgot-password">Forgot Password?</a>
                  <p class="text-center text-muted mt-5 mb-0">Don't have an account?<a routerLink="/register-user"
                      class="fw-bold text-body"><u > Sign Up</u></a></p>
                </form>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section>
```

Odsječak programskog kôda 4.16 HTML datoteka Sign-in komponente

```
<section class="vh-100 bg-image"
  style="background-image: url('https://images.pexels.com/photos/270360/pexels-photo-270360.jpeg');">
  <div class="mask d-flex align-items-center h-100 gradient-custom-3">
    <div class="container h-100">
      <div class="row d-flex justify-content-center align-items-center h-100">
        <div class="col-12 col-md-9 col-lg-7 col-xl-6">
          <div class="card" style="border-radius: 15px;">
            <div class="card-body p-5">
              <h2 class="text-uppercase text-center mb-5">Sign up</h2>
              <form>
                <div class="form-outline mb-4">
                  <input type="email" id="form3Example3cg" class="form-control form-control-lg" placeholder="Email" #userName />
                  <label class="form-label" for="form3Example3cg">Your Email</label>
                </div>
                <div class="form-outline mb-4">
                  <input type="password" id="form3Example4cg" class="form-control form-control-lg" placeholder="Password" #userPassword required />
                  <label class="form-label" for="form3Example4cg">Password</label>
                </div>
                <div class="d-flex justify-content-center">
                  <button type="button"
                    class="btn btn-block btn-lg gradient-custom-4 text-body" (click)="authService.SignUp(userName.value, userPassword.value)">Sign up</button>
                <button type="button" class="btn" (click)="authService.GoogleAuth()">
                  <svg xmlns="http://www.w3.org/2000/svg" width="20" height="20" fill="currentColor" class="bi bi-google" viewBox="0 0 16 16">
                    <path d="M15.545 6.558a9.42 9.42 0 0 1 .139 1.626c0 2.434-.87 4.492-2.384 5.885h.002C11.978 15.292 10.158 16 8 16A8 8 0 1 1 8 0a7.689 7.689 0 0 1 5.352 2.082l-2
                  </svg>
                  Continue with Google
                </button>
              </div>
              <p class="text-center text-muted mt-5 mb-0">Already have an account?<a routerLink="/sign-in"
                  class="fw-bold text-body"><u > Log In</u></a></p>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
```

Odsječak programskog kôda 4.17 HTML datoteka Sign-up komponente

```
<section class="vh-100 bg-image"
  style="background-image: url('https://images.pexels.com/photos/270360/pexels-photo-270360.jpeg');">
  <div class="mask d-flex align-items-center h-100 gradient-custom-3">
    <div class="container h-100">
      <div class="row d-flex justify-content-center align-items-center h-100">
        <div class="col-12 col-md-9 col-lg-7 col-xl-6">
          <div class="card" style="border-radius: 15px;">
            <div class="card-body p-5">
              <h2 class="text-uppercase text-center mb-5">Thank You for Registering</h2>

              <form>


                <div class="form-outline mb-4" *ngIf="authService.userData as user">
                  <p class="text-center">We have sent a confirmation email to <strong>{{user.email}}</strong>.</p>
                <p class="text-center">Please check your email and click on the link to verfiy your email address.</p>
                </div>
                <div class="d-flex justify-content-center">
                  <button type="button"
                  value="Reset Password" class="btn btn-block btn-lg gradient-custom-4 text-body" (click)="authService.SendVerificationMail()">Resend Verification Email</button>
                </div>
                <p class="text-center text-muted mt-5 mb-0">Go back to?<a routerLink="/sign-in"
                    class="fw-bold text-body"><u >Log In</u></a></p>

              </form>

            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
```

Odsječak programskog kôda 4.18 HTML datoteka verify-email komponente

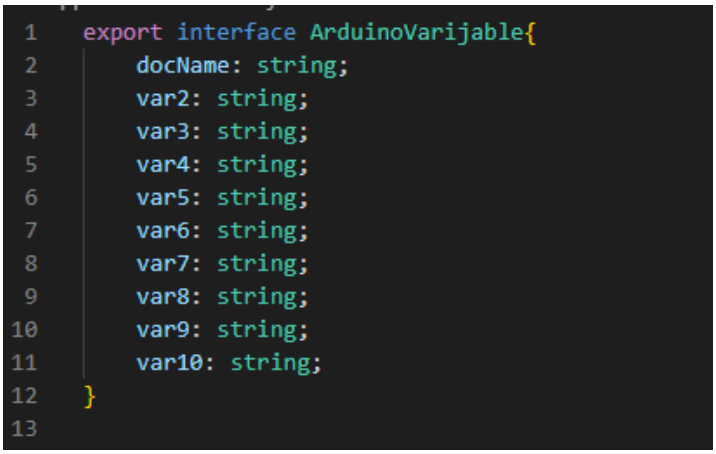
```
<section class="vh-100 bg-image"
  style="background-image: url('https://images.pexels.com/photos/270360/pexels-photo-270360.jpeg');">
  <div class="mask d-flex align-items-center h-100 gradient-custom-3">
    <div class="container h-100">
      <div class="row d-flex justify-content-center align-items-center h-100">
        <div class="col-12 col-md-9 col-lg-7 col-xl-6">
          <div class="card" style="border-radius: 15px;">
            <div class="card-body p-5">
              <h2 class="text-uppercase text-center mb-5">Reset password</h2>
              <form>
                <div class="form-outline mb-4">
                  <input type="email" id="form3Example3cg" class="form-control form-control-lg" placeholder="Email" #passwordResetEmail />
                  <label class="form-label" for="form3Example3cg">Please enter your email address to request a password reset</label>
                </div>
                <div class="d-flex justify-content-center">
                  <button type="button"
            value="Reset Password" class="btn btn-block btn-lg gradient-custom-4 text-body" (click)="authService.ForgotPassword(passwordResetEmail.value)">Send  Email</button>
                </div>
                <p class="text-center text-muted mt-5 mb-0">Go back to?<a  routerLink="/sign-in"
                    class="fw-bold text-body"><u >Log In</u></a></p>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
```

Odsječak programskog kôda 4.18 HTML datoteka forgot-password komponente

Nakon uspješne prijave Angular preusmjerava korisnika na dashboard komponentu koja direktno komunicira s pozadinskom aplikacijom. Kako bi se to ostvarilo potrebno je napraviti TypeScript datoteku arduino-varijable.service.ts (Odsječak programskog kôda 4.20) koja po strukturi zrcali Spring Boot klasi ArduinoVarijableResource i sučelja koja predstavljaju sheme objekta modela klase pozadinske aplikacije (Odsječak programskog kôda 4.21, 4.22). U datoteci su definirani URL pozadinske aplikacije i implementacije svih REST krajnjih točaka.

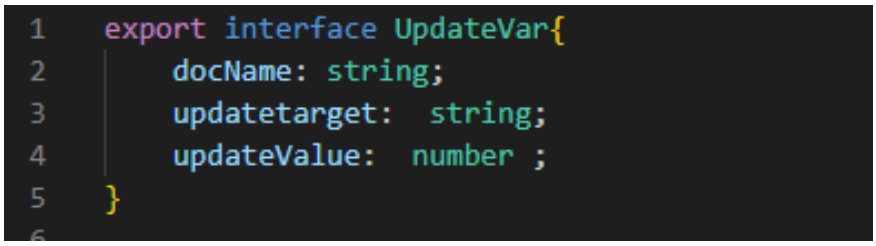
```
import { HttpClient, HttpParams } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { ArduinoVarijable } from './ArduinoVarijable';
import { UpdateVar } from './UpdateVar';

@Injectable({
  providedIn: 'root'
})
export class ArduinoVarijableService {

  constructor(private http: HttpClient) { }
  private apiServerUrl='http://localhost:8080/ArduinoVarijable';


  public getArduinoVarijable(docName: string) : Observable<ArduinoVarijable>{
    return this.http.get<ArduinoVarijable>(`${this.apiServerUrl}/get/${docName}`);
  }

  public updateArduinoVarijable(updatevar: UpdateVar) : Observable<any>{
    return this.http.put<any>(`${this.apiServerUrl}/update`,updatevar, { responseType: 'text' as 'json',});
  }


}

```

Odsječak programskog kôda 4.20 arduino-varijable.service.ts datoteka



```
export interface ArduinoVarijable{
    docName: string;
    var2: string;
    var3: string;
    var4: string;
    var5: string;
    var6: string;
    var7: string;
    var8: string;
    var9: string;
    var10: string;
}
```

Odsječak programskog kôda 4.21 Sučelje objekta ArduinoVarijable



```
export interface UpdateVar{
    docName: string;
    updatetarget:  string;
    updateValue:  number ;
}
```

Odsječak programskog kôda 4.22 Sučelje objekta UpdateVar

Zadnji korak izrade Angular aplikacije je dodavanje funkcionalnosti dashboard komponente pisanjem kôda u TypeScript datoteci te komponente i kreiranje korisničkog sučelja dashboard komponente pisanjem HTML kôda.

Div element main-container (Odsječak programskog kôda 4.23) sadrži upute za korištenje web aplikacije i gumb čiji pritisak poziva funkciju getArduinoVarijable (Odsječak programskog kôda 4.24). Funkcija svakih 1000ms poziva arduino-varijable.service funkciju getArduinoVarijable koja šalje HTTP GET zahtjev za vrijednost varijabla pozadinskoj aplikaciji.

Navigacijska traka (Odsječak programskog kôda 4.25) sadrži hiperlink (engl. *hyperlink*) element koji pritiskom poziva authService funkciju SignOut i gumb s bootstrap atributom data-toggle="collapse" čiji pritisak otvara navbarToggleExternalContent div element (Odsječak programskog kôda 4.26). Div element navbarToggleExtralContent sadrži gumbove koji predstavljaju svaku moguću varijablu, ali koristeći Angular *ngIf funkciju samo se ne null elementi prikazuju korisniku. Kao i gumb na navigacijskoj traci svaki gumb u navbarToggleExtralContent elementu sadrži atribut „data-toggle="collapse"“. Svaki gumb

predstavlja jednu varijablu i pritisak na gumb prikazuje novi div element ovisno o pritisnutom gumbu.

```
<div class="container-fluid" id="main-container">
  <div class="row">
    <main role="main" class="col-md-9 ml-sm-auto col-lg-10 px-4">
      <div class="inner-adjust">
        <div class="pt-3 pb-2 mb-3 border-bottom">
          <h1 class="h2">How to use</h1>
        </div>
        <div class="row" *ngIf="authService.userData as user">
          <div class="col-md-12">
            <div class="card card-body">
              <div class="tutorial">
                <p>User ID: <span id="uuid">{{user.uid}}</span></p>
                <p>Email Verified: <strong>{{user.emailVerified}}</strong></p>
                <h3>Step1</h3>
                <p>Download arduino IDE project template files -> <a href="./downloads/Firestore_Arduino/Firestore_getData_template/Firestore_getData_template.ino"
                  download="Firestore_getData_template.ino">
                  Firestore_getData_template.ino
                </a> -> <a href="./downloads/Firestore_Arduino/Firestore_sendData_template/Firestore_sendData_template.ino"
                download="Firestore_sendData_template.ino">
                Firestore_sendData_template.ino
              </a>
              </p>
                <h3>Step2</h3>
                <p>tools -> port -> your port -> tools -> Board -> your board</p>
                <h3>Step3</h3>
                <p>sketch -> Include Library -> Manage Libraries -> install Firebase Arduino Client Library for ESP8266 and ESP32 by Mobizt</p>
                <h3>Step4</h3>
                <p>Insert your credentials -> write your code in insert your code areas of the template </p>
                <p>For more info on how to use visit:  <a href="https://github.com/mobizt/Firebase-ESP-Client/tree/main/examples/Firestore" target="_blank">
                  https://github.com/mobizt/Firebase-ESP-Client/tree/main/examples/Firestore</a></p>
                <h3>Step5</h3>
                <p>Upload code to your ardunio device ->
                  click scan for ardunio variables on this website ->
                  if your ardunio variables were detected click plus sign on the top right corner -> c
                  lick detected variable and drag and drop varius elements to the drag and drop here field </p>
              </div>

            </div>
          </div>

<button class="btn scanbutt" (click)="getArduinoVarijable(user.uid)">Scan for arduino variables</button>

```

Odsječak programskog kôda 4.23 Div element main-container

```
public getArduinoVarijable(userId: string): void {
  if(this.state){
  this.state = false;
  window.location.reload();
}
this.state = true;
  const reloadInterval = 1000;
  timer(0, reloadInterval).pipe(
    mergeMap(_ =>  this.ArduinoService.getArduinoVarijable(userId!))
  ).subscribe(
    (response: ArduinoVarijable) => {
      this.varijable = response;
      console.log(this.varijable);

    },
    (error: HttpErrorResponse) => {
      alert(error.message);
    }
  );
}
```

Odsječak programskog kôda 4.24 Funkcija getArduinoVarijable

```
426
427   <nav class="navbar navbar-dark bg-dark navbutton">
428     <li class="nav-item active">
429
430       <a class="logout" (click)="authService.SignOut()">
431       Log out
432       </a>
433     </li>
434     <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggleExternalContent" aria-controls="navbarToggleExternalContent"
435     aria-expanded="false" aria-label="Toggle navigation">
436       <svg xmlns="http://www.w3.org/2000/svg" width="30" height="30" fill="currentColor" class="bi bi-plus-lg" viewBox="0 0 16 16">
437         <path fill-rule="evenodd" d="M8 2a.5.5 0 0 1 .5.5v5h5a.5.5 0 0 1 0 1h-5v5a.5.5 0 0 1-1 0v-5h-5a.5.5 0 0 1 0-1h5v-5A.5.5 0 0 1 8 2Z"/>
438       </svg>
439     </button>
440   </nav>
441 </div>
442
```

Odsječak programskog kôda 4.25 Navigacijska traka

```
2  <div class="pos-f-t">
3    <div class="collapse" id="navbarToggleExternalContent">
4      <div class="bg-dark p-4" *ngIf="varijable as varijabla">
5        <div  class="row">
6          <div *ngIf="varijabla.var2" class="col-sm">
7            <button class="btn btn-primary" id="var2Button"  type="button" data-toggle="collapse" data-target="#var2" aria-expanded="false" aria-controls="collapseExample" >
8              variable 2
9              </button>
10         </div>
11         <div *ngIf="varijabla.var3" class="col-sm">
12           <button class="btn btn-primary" id="var3Button" type="button" data-toggle="collapse" data-target="#var3" aria-expanded="false" aria-controls="collapseExample">
13             variable 3
14             </button>
15         </div>
16         <div *ngIf="varijabla.var4" class="col-sm">
17           <button class="btn btn-primary" type="button" data-toggle="collapse" data-target="#var4" aria-expanded="false" aria-controls="collapseExample">
18             variable 4
19             </button>
20         </div>
21         <div *ngIf="varijabla.var5" class="col-sm">
22           <button class="btn btn-primary" type="button" data-toggle="collapse" data-target="#var5" aria-expanded="false" aria-controls="collapseExample">
23             variable 5
24             </button>
25         </div>
26         <div *ngIf="varijabla.var6" class="col-sm">
27           <button class="btn btn-primary" type="button" data-toggle="collapse" data-target="#var6" aria-expanded="false" aria-controls="collapseExample">
28             variable 6
29             </button>
30         </div>
31         <div *ngIf="varijabla.var7" class="col-sm">
32           <button class="btn btn-primary" type="button" data-toggle="collapse" data-target="#var7" aria-expanded="false" aria-controls="collapseExample">···
34             </button>
35         </div>
36         <div *ngIf="varijabla.var8" class="col-sm">
37           <button class="btn btn-primary " type="button" data-toggle="collapse" data-target="#var8" aria-expanded="false" aria-controls="collapseExample">···
39             </button>
40         </div>
41         <div *ngIf="varijabla.var9" class="col-sm">
42           <button class="btn btn-primary " type="button" data-toggle="collapse" data-target="#var9" aria-expanded="false" aria-controls="collapseExample">···
44             </button>
45         </div>
46         <div *ngIf="varijabla.var10" class="col-sm"> ···
50         </div>
51         </div>
52
```

Odsječak programskog kôda 4.26 Div element navbarToggleExtralContent

Svaka ladica s elementima sadržava iste tipove elemenata s razlikom da se elementi odnose na različitu varijablu ovisno o tome koji gumb je pritisnut da se otvori ladica. Prvi od tih tipova elemenata je igx-radial-gauge što je „igniteui-angular-gauges“ uvezeni element (Odsječak programskog kôda 4.27). Drugi tip elementa prikazuje brojčanu vrijednost varijable zajedno s mjernom jedinicom (Odsječak programskog kôda 4.28). Treći tip elementa je hiperlink čiji pritisak poziva funkciju openModal (Odsječak programskog kôda 4.29). Funkcija openModal postavlja vrijednost globalne varijable na predanu vrijednost i otvara forma (Odsječak programskog kôda 4.30) element koji je inače nevidljiv, na način da kreira gumb element koji cilja na form element i pritisne ga. Pritiskom na submit gumb u form elementu poziva se ChangeVarName funkcija koja mijenja tekst trećeg tipa elementa. Ostali tipovi elemenata funkcioniraju na sličan način kao i treći tip elementa s tim da četvrti omogućuje mijenjanje vrijednosti varijable, peti mijenjanje naziva varijable, šesti mijenjanje mjerne jedinice varijable, sedmi mijenjanje vrijednosti varijable pomoću klizaća i osmi mijenjanje vrijednosti varijable s 0 na 1 i obrnuto preko prekidač gumba (switch). Svi tipovi elementa imaju Angular cdkDrag atribut što omogućuje povlačenje i ispuštanje elemenata.

```
<div class="collapse pickitemcard" id="var2">
  <div class="row col-sm">
    <div class="pickitem" cdkDrag >
      <igx-radial-gauge
      #radialGauge
      height="330px"
      width="100%"
      value="{{varijabla.var2}}"
      interval=5
      minimumValue=0
      maximumValue=20
      labelInterval=5
      labelExtent=0.71
      minorTickCount=4
      minorTickEndExtent=.625
      minorTickStartExtent=.6
      minorTickStrokeThickness=1
      minorTickBrush = "#79797a"
      tickStartExtent=.6
      tickEndExtent=.65
      tickStrokeThickness=2
      tickBrush = "#79797a"
      needleShape="Triangle"
      needleEndWidthRatio=0.03
      needleStartWidthRatio=0.05
      needlePivotShape="CircleOverlay"
      needlePivotWidthRatio=0.15
      needleBaseFeatureWidthRatio=0.15
      needleBrush="#79797a"
      needleOutline="#79797a"
      needlePivotBrush="#79797a"
      needlePivotOutline="#79797a"
      isNeedleDraggingEnabled=false
      backingBrush="#fcfcfc"
      backingOutline="#d6d6d6"
      backingStrokeThickness=5
      scaleStartAngle=120
      scaleEndAngle=60
      scaleBrush="#d6d6d6"
      rangeBrushes="#F86232, #DC3F76, #7446B9"
      rangeOutlines="#F86232, #DC3F76, #7446B9">
    </igx-radial-gauge>
  </div>
```

Odsječak programskog kôda 4.27 igx-radial-gauge element

Odsječak programskog kôda 4.28 Drugi i treći tip elementa

Odsječak programskog kôda 4.29 Funkcija openModal

Odsječak programskog kôda 4.29 insertTextModal form element

Odsječak programskog kôda 4.30 ChangeVarName funkcija

```typescript
import { HttpErrorResponse } from '@angular/common/http';
import { Component, Input, OnInit } from '@angular/core';
import { FormControl, NgForm } from '@angular/forms';
import { mergeMap, Observable, timer } from 'rxjs';
import { ArduinoVarijableService } from 'src/app/arduino-varijable.service';
import { ArduinoVarijable } from 'src/app/ArduinoVarijable';
import { UpdateVar } from 'src/app/UpdateVar';
import { AuthService } from '../../shared/services/auth.service';
import { Options } from '@angular-slider/ngx-slider';
@Component({
  selector: 'app-dashboard',
  templateUrl: './dashboard.component.html',
  styleUrls: ['./dashboard.component.scss'],
})
export class DashboardComponent implements OnInit {
  constructor(public authService: AuthService, private ArduinoService: ArduinoVarijableService) {}
  public varijable!: ArduinoVarijable;
  public updatevar!: UpdateVar;
  public modaltarget!: string;
  public updatetarget!: string;
  public varNameTarget!: string;
  public varUnitTarget!: string;
  public Var2Unit: string = "";
  public state: boolean = false;
  ngOnInit(): void {
  }
```

Odsječak programskog kôda 4.31 Uvezene komponente i deklaracija globalnih varijabli dashboard TypeScript datoteke

```
<div class="modal fade" id="uptade" tabindex="-1" role="dialog" aria-labelledby="uptadeLabel" aria-hidden="true" *ngIf="authService.userData as user">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-body">
        <form #updateform="ngForm" (ngSubmit)="updateVarijable(updateform, user.uid)">
          <div class="form-group inviz">
            <input type="text" ngModel name="docName" class="form-control" id="docName">
          </div>
          <div class="form-group inviz">
            <input type="text" ngModel name="updatetarget" class="form-control" id="updatetarget">
          </div>
          <div class="form-group">
            <input type="number" ngModel name="updateValue" class="form-control" id="updateValue" aria-describedby="Update value" value="Insert new value" required>
          </div>


          <div class="modal-footer">
            <button type="button" id="update-dokument-form" class="btn btn-secondary" data-dismiss="modal">Close</button>
            <button [disabled]="updateform.invalid"  type="submit" class="btn btn-secondary">Save changes</button>
          </div>
          </form>

        </div>
      </div>
    </div>
    </div>

<div class="modal fade" id="ChangeVarName" tabindex="-1" role="dialog" aria-labelledby="ChangeVarNameLabel" aria-hidden="true" *ngIf="authService.userData as user">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-body">
          <form #ChangeVarNameForm="ngForm" (ngSubmit)="ChangeVarName(ChangeVarNameForm)">
            <input type="text" ngModel name="naziv" class="form-control" id="Naziv" aria-describedby="InsertTextHelp" placeholder="Insert text here" required>
            <div class="modal-footer">
              <button type="button" id="ChangeVarName-dokument-form" class="btn btn-secondary" data-dismiss="modal">Close</button>
              <button [disabled]="ChangeVarNameForm.invalid"  type="submit" class="btn btn-secondary">Save changes</button>
            </div>
            </form>

          </div>
        </div>
      </div>
    </div>
```

Odsječak programskog kôda 4.32 Četvrti i peti tip elementa

```
<div class="modal fade" id="ChoseUnit" tabindex="-1" role="dialog" aria-labelledby="ChoseUnitLabel" aria-hidden="true" *ngIf="authService.userData as user">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-body">
        <form #ChoseUnitForm="ngForm" (ngSubmit)="ChoseUnit(ChoseUnitForm)">
          <select ngModel name="unit" class="form-control form-control-lg">
            <option>K</option>
            <option>°C</option>
            <option>%</option>
          </select>
        <div class="modal-footer">
          <button type="button" id="ChoseUnit-dokument-form" class="btn btn-secondary" data-dismiss="modal">Close</button>
          <button [disabled]="ChoseUnitForm.invalid"  type="submit" class="btn btn-secondary">Save changes</button>
        </div>
        </form>

      </div>
    </div>
  </div>
</div>
```

Odsječak programskog kôda 4.33 Šesti tip elementa

```
<div class="pickitemtype4 col-sm" >
<button cdkDrag class="btn btn-secondary insertValueButt" (click)="openUpdateModal('var2')">
Change value
</button>
</div>
<div class="pickitemtype4 col-sm">
<button  class="btn btn-secondary insertValueButt" cdkDrag (click)="openChoseUnitModal('var2')">
  Change measurement unit
  </button>
</div>
<div class="pickitemtype4 col-sm" >
  <button  class="btn btn-secondary insertValueButt" cdkDrag (click)="openChangeVarNameModal('var2Button')">
    Change variable name
    </button>
  </div>

  <div class="pickitemtype4 col-sm" *ngIf="authService.userData as user" cdkDrag>
    <form #updateform="ngForm">
      <div class="form-group inviz">···
      </div>
      <div class="form-group inviz">···
    </div>
      <div class="form-group inviz">···
    </div>
      <ngx-slider [(value)]="value" [options]="options" (valueChange)="updateslidervalue(updateform,'var2', user.uid)"></ngx-slider>
    </form>
</div>

<div class="pickitemtype4 col-sm">
  <form #updateform="ngForm">
    <div class="form-group inviz">···
    </div>
    <div class="form-group inviz">···
  </div>
  <div class="form-group inviz">···
</div>
<div *ngIf="authService.userData as user">
  <button class="btn btn-secondary" cdkDrag *ngIf="varijabla.var2 == '1' || varijabla.var2 == '0'" (click)="switcValue('var2', varijabla.var2 , updateform, user.uid)">
    <svg xmlns="http://www.w3.org/2000/svg" width="100%" height="100%" fill="currentColor" class="bi bi-power" viewBox="0 0 16 16">
      <path d="M7.5 1v7h1V1h-1z"/>
      <path d="M3 8.812a4.999 4.999 0 0 1 2.578-4.375l-.485-.874A6 6 0 1 0 11 3.616l-.501.865A5 5 0 1 1 3 8.812z"/>
    </svg>
  </button>
</div>
</form>

</div>
</div>
</div>
```

Odsječak programskog kôda 4.34 Sedmi i osmi tip elementa

```
  public switcValue(position: string, value: string, UpdateForm: NgForm, docName: String){
if(value == "0"){
UpdateForm.value.updateValue = 1;
}
else UpdateForm.value.updateValue = 0;
UpdateForm.value.docName=docName;
UpdateForm.value.updatetarget=position;
this.ArduinoService.updateArduinoVarijable(UpdateForm.value).subscribe(
  (response: any) => {
    this.updatevar = response;
    console.log(response);
    UpdateForm.reset();
  },
  (error: HttpErrorResponse) => {
    alert(error.message);
    UpdateForm.reset();
  }
);
  }

  public openUpdateModal(position: string){
    const container = document.getElementById('main-container');
    const button = document.createElement('button');
    button.type = 'button';
    button.style.display = 'none';
    button.setAttribute('data-toggle', 'modal');
    button.setAttribute('data-target', '#uptade');
    this.updatetarget = position
    container?.appendChild(button);
    button.click();
  }
  public openChangeVarNameModal(ChangeVarName: string){
    const container = document.getElementById('main-container');
    const button = document.createElement('button');
    button.type = 'button';
    button.style.display = 'none';
    button.setAttribute('data-toggle', 'modal');
    button.setAttribute('data-target', '#ChangeVarName');
    this.varNameTarget = ChangeVarName;
    container?.appendChild(button);
    button.click();
  }
  public openChoseUnitModal(ChoseUnitTarget: string){
    const container = document.getElementById('main-container');
    const button = document.createElement('button');
    button.type = 'button';
    button.style.display = 'none';
    button.setAttribute('data-toggle', 'modal');
    button.setAttribute('data-target', '#ChoseUnit');
    this.varUnitTarget = ChoseUnitTarget;
    container?.appendChild(button);
    button.click();

  }
```

Odsječak programskog kôda 4.35 Funkcije za switch gumb, otvaranje forme za ažuriranje vrijednosti, mijenjanja naziva varijable i mijenjanja mjerne jedinice u dashboard TypeScript datoteci

```
127   public updateVarijable(UpdateForm: NgForm, docName: String){
128     document.getElementById('add-dokument-form')?.click();
129     UpdateForm.value.updatetarget=this.updatetarget;
130     UpdateForm.value.docName=docName;
131     this.ArduinoService.updateArduinoVarijable(UpdateForm.value).subscribe(
132       (response: any) => {
133         console.log(response);
134         UpdateForm.reset();
135       },
136       (error: HttpErrorResponse) => {
137         alert(error.message);
138         UpdateForm.reset();
139       }
140     );
141
142   }
143
144   public ChoseUnit(ChangeVarUnit: NgForm){
145     document.getElementById('ChoseUnit-dokument-form')?.click();
146    if(this.varUnitTarget == "var2") this.Var2Unit = ChangeVarUnit.value.unit;
147   }
148
149   ConvertToInt(num: string){
150     return parseInt(num);
151
152
153   }
154
155   public updateslidervalue(UpdateForm: NgForm,Updatetarget: string, docName: String){
156     UpdateForm.value.updatetarget=Updatetarget;
157     UpdateForm.value.docName=docName;
158     UpdateForm.value.updateValue = this.value.toString();
159     this.ArduinoService.updateArduinoVarijable(UpdateForm.value).subscribe(
160       (response: any) => {
161         console.log(response);
162         UpdateForm.reset();
163       },
164       (error: HttpErrorResponse) => {
165         alert(error.message);
166         UpdateForm.reset();
167       }
168     );
169
170   }
171
172
```

Odsječak programskog kôda 4.36 Funkcije za ažuriranje vrijednosti varijable, biranje mjerne jedinice i ažuriranje vrijednosti preko slider elementa u dashboard TypeScript datoteci

4.4 Kreiranje Arduino IDE predložaka

Kako bi se olakšalo pisanje Arduino IDE kôda kreirana su dva predloška koja imaju sav kôd potreban za spajanje na internet i na bazu podataka. Predložak koristi uvezenu biblioteku Firebase Arduino Client Library for ESP8266 and ESP32 by mobizt. Predložak getData (Odsječak programskog kôda 4.39) sadrži kôd za primanje podataka s baze, a sendData (Odsječak programskog kôda 4.40) za slanje podataka na bazu.

```
  #include <WiFi.h>
#elif defined(ESP8266)
  #include <ESP8266WiFi.h>
#endif
#include <Firebase_ESP_Client.h>
#include "addons/TokenHelper.h"
#include "addons/RTDBHelper.h"
#define API_KEY "AIzaSyDsSCZY1KAseOragmEPxZb0SJ_0vsCuYOE"
#define FIREBASE_PROJECT_ID "wine-cellar-monitor-d7139"


// Insert your  credentials
#define WIFI_SSID "Your Wifi WIFI ssid"
#define WIFI_PASSWORD "Your WIFI password"
#define USER_EMAIL "Your Email"
#define USER_PASSWORD "Your password"
#define USER_Uid "Your Uid"


FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;
bool taskCompleted = false;
unsigned long dataMillis = 0;
int count = 0;


void fcsUploadCallback(CFS_UploadStatusInfo info)
{
    if (info.status == fb_esp_cfs_upload_status_init)
    {
        Serial.printf("\nUploading data (%d)...\n", info.size);
    }
    else if (info.status == fb_esp_cfs_upload_status_upload)
    {
        Serial.printf("Uploaded %d%s\n", (int)info.progress, "%");
    }
    else if (info.status == fb_esp_cfs_upload_status_complete)
    {
        Serial.println("Upload completed ");
    }
    else if (info.status == fb_esp_cfs_upload_status_process_response)
    {
        Serial.print("Processing the response... ");
    }
    else if (info.status == fb_esp_cfs_upload_status_error)
    {
        Serial.printf("Upload failed, %s\n", info.errorMsg.c_str());
    }
```

Odsječak programskog kôda 4.37 Kôd za postavljanje koji dijele oba predloška

```
void setup()
{

    Serial.begin(9600);

    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to Wi-Fi");
    while (WiFi.status() != WL_CONNECTED)
    {
        Serial.print(".");
        delay(300);
    }
    Serial.println();
    Serial.print("Connected with IP: ");
    Serial.println(WiFi.localIP());
    Serial.println();

    Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);
    config.api_key = API_KEY;
    auth.user.email = USER_EMAIL;
    auth.user.password = USER_PASSWORD;
    config.token_status_callback = tokenStatusCallback; // see addons/TokenHelper.h

#if defined(ESP8266)
    // In ESP8266 required for BearSSL rx/tx buffer for large data handle, increase Rx size as needed.
    fbdo.setBSSLBufferSize(2048 /* Rx buffer size in bytes from 512 - 16384 */, 2048 /* Tx buffer size in bytes from 512 - 16384 */);
#endif
    fbdo.setResponseSize(2048);
    Firebase.begin(&config, &auth);
    Firebase.reconnectWiFi(true);



//Your setup() code here
```

Odsječak programskog kôda 4.38 Void setup koji dijele oba predloška

```
{
  if (Firebase.ready() && (millis() - dataMillis > 1000 || dataMillis == 0))
    {
        dataMillis = millis();
if (!taskCompleted){
        taskCompleted = true;
        FirebaseJson content;
        String documentPath = "ArduinoVarijable/" +String(USER_Uid);
        String doc_path;
        doc_path += FIREBASE_PROJECT_ID;
        doc_path += "/databases/(default)/documents/ArduinoVarijable/" + String(USER_Uid); // coll_id and doc_id are your collection id and document id
        content.set("fields/docName/stringValue", String(USER_Uid));
        content.set("fields/var2/stringValue", "0");
        content.set("fields/var3/stringValue", "0");
        content.set("fields/var4/stringValue", "0");
        content.set("fields/var5/stringValue", "0");
        content.set("fields/var6/stringValue", "0");
        content.set("fields/var7/stringValue", "0");
        content.set("fields/var8/stringValue", "0");
        content.set("fields/var9/stringValue", "0");
        content.set("fields/var10/stringValue", "0");

        Serial.print("Create a document... ");

        if (Firebase.Firestore.createDocument(&fbdo, FIREBASE_PROJECT_ID, "" /* databaseId can be (default) or empty */, documentPath.c_str(), content.raw()))
            Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
        else
            Serial.println(fbdo.errorReason());

}
        std::vector<struct fb_esp_firestore_document_write_t> writes;
        struct fb_esp_firestore_document_write_t update_write;
        update_write.type = fb_esp_firestore_document_write_type_update;
        FirebaseJson content;
        String documentPath = "ArduinoVarijable/" +  String(USER_Uid);
        content.set("fields/docName/stringValue", String(USER_Uid));
      // your code here

        content.set("fields/var5/stringValue", String(rand()).c_str()); // Send radnom data to var5 example
        update_write.update_document_content = content.raw();
        update_write.update_document_path = documentPath.c_str();
        writes.push_back(update_write);
        if (Firebase.Firestore.commitDocument(&fbdo, FIREBASE_PROJECT_ID, "" /* databaseId can be (default) or empty */, writes /* dynamic array of fb_esp_firestore_document_write_t */, "" /* transaction */))
            Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
        else
            Serial.println(fbdo.errorReason());

    }
}
```

Odsječak programskog kôda 4.39 Void loop sendData predloška

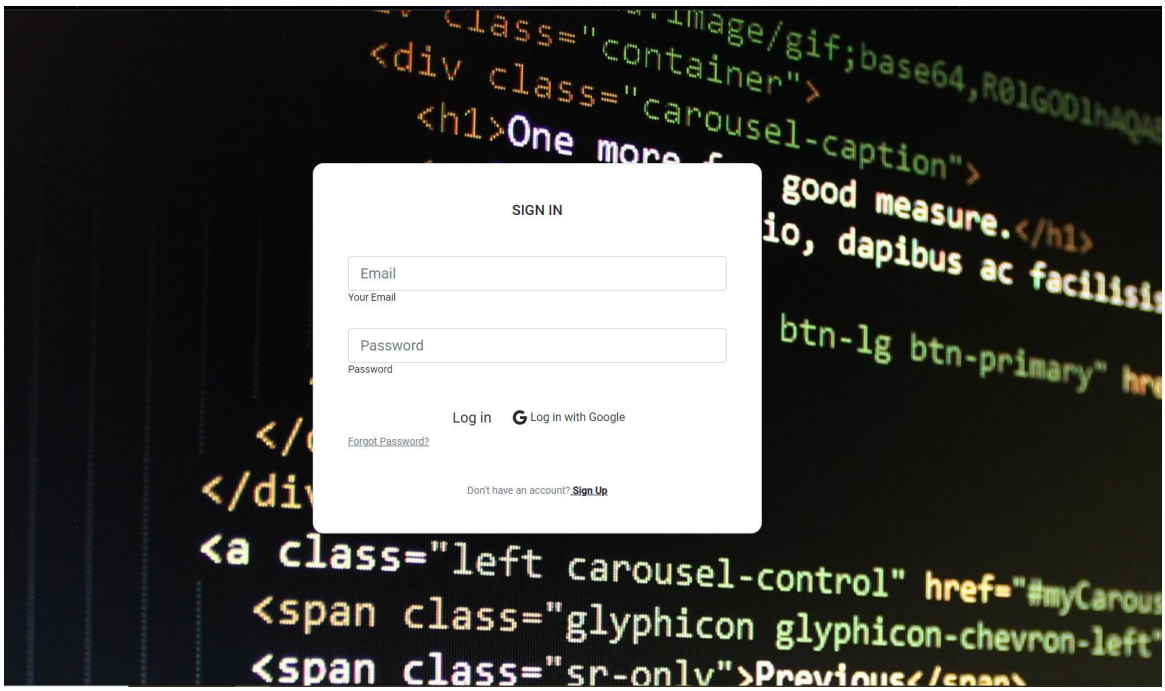
```
    if (Firebase.ready() && (millis() - dataMillis > 60 || dataMillis == 0))
      {
          dataMillis = millis();
    if (!taskCompleted){
          taskCompleted = true;
          FirebaseJson content;
          String documentPath = "ArduinoVarijable/" +String(USER_Uid);
          String doc_path;
          doc_path += FIREBASE_PROJECT_ID;
          doc_path += "/databases/(default)/documents/ArduinoVarijable/" + String(USER_Uid); // coll_id and doc_id are your collection id and document id
          content.set("fields/docName/stringValue", String(USER_Uid));
          content.set("fields/var2/stringValue", "0");
          content.set("fields/var3/stringValue", "0");
          content.set("fields/var4/stringValue", "0");
          content.set("fields/var5/stringValue", "0");
          content.set("fields/var6/stringValue", "0");
          content.set("fields/var7/stringValue", "0");
          content.set("fields/var8/stringValue", "0");
          content.set("fields/var9/stringValue", "0");
          content.set("fields/var10/stringValue", "0");
          count++;

          Serial.print("Create a document... ");

          if (Firebase.Firestore.createDocument(&fbdo, FIREBASE_PROJECT_ID, "" /* databaseId can be (default) or empty */, documentPath.c_str(), content.raw()))
              Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
          else
              Serial.println(fbdo.errorReason());

    }

        String documentPath = "ArduinoVarijable/" + String(USER_Uid);

          // Write code in this if statment to get data from server

          if (Firebase.Firestore.getDocument(&fbdo, FIREBASE_PROJECT_ID, "", documentPath.c_str())){
              FirebaseJson payload;
              payload.setJsonData(fbdo.payload().c_str());
              FirebaseJsonData jsonData;
              payload.get(jsonData, "fields/var2/stringValue", true); //Get var2 data example
              String str = jsonData.stringValue;
              if (str == "1") // Led ON/Of example
              digitalWrite(LED_BUILTIN, HIGH);
              else
              digitalWrite(LED_BUILTIN, LOW);
              }
          else
              Serial.println(fbdo.errorReason());

      }
```

Odsječak programskog kôda 4.40 Void loop getData predloška

4.5 Prikaz funkcionalnosti i izgleda web aplikacije

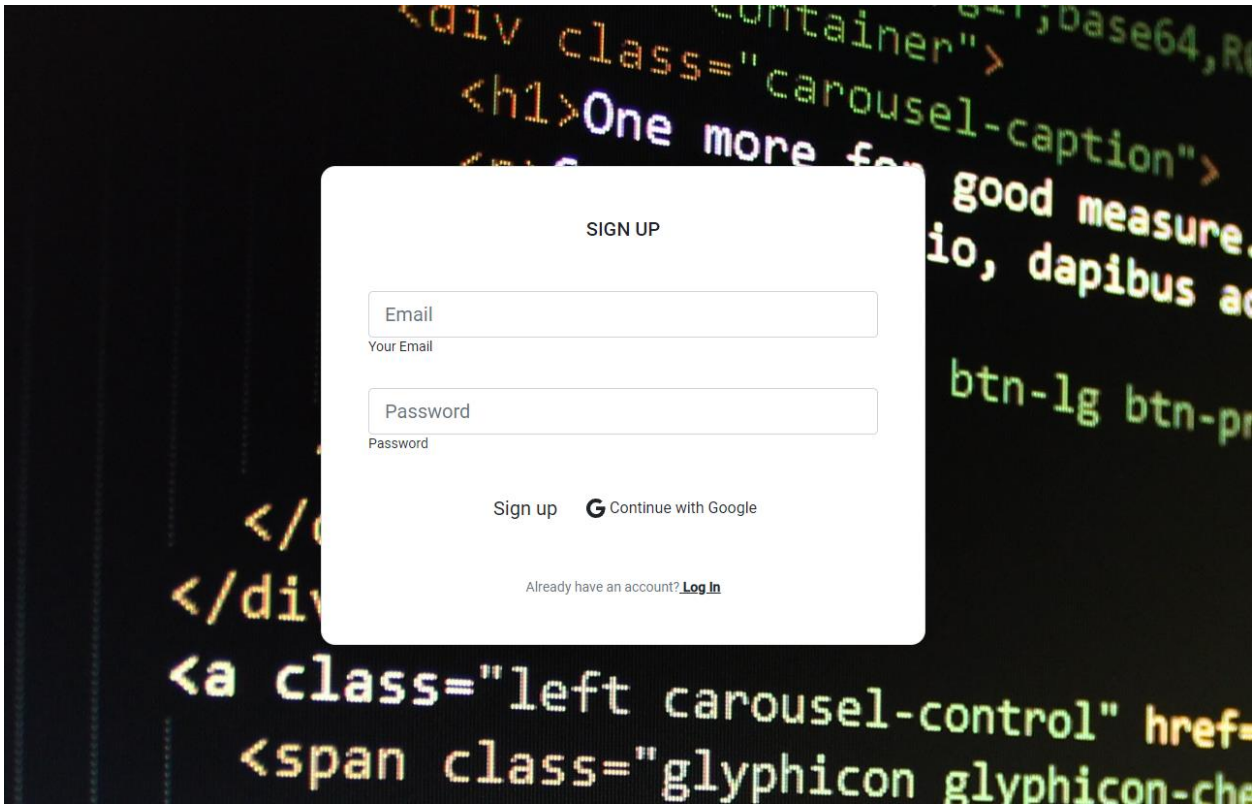
U ovom poglavlju je prikazan proces korištenja web aplikacije zajedno s slikama koje vizualno prikazuju svaki korak procesa.

Prvi prozor koji korisnik vidi nakon ulaska u web aplikaciju je prozor za prijavu (Slika 4.41) koji sadrži formu. U slučaju da korisnik nije registriran može se prijaviti s Google računom ili može kliknuti na Sign Up što preusmjerava korisnika na novi prozor za registraciju (Slika 4.42).

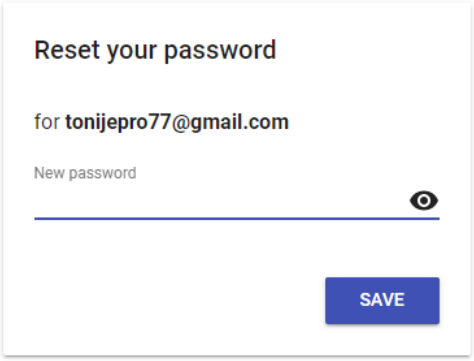


Slika 4.41 Prozor za prijavu

Ako je korisnik zaboravio lozinku može pritisnuti na forgot password što preusmjerava korisnika na novi prozor (Slika 4.43).

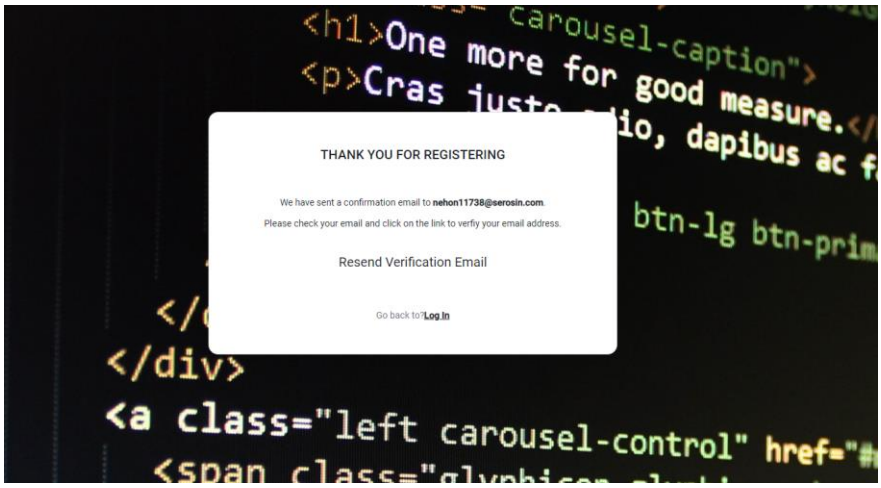


Slika 4.42 Prozor za registraciju

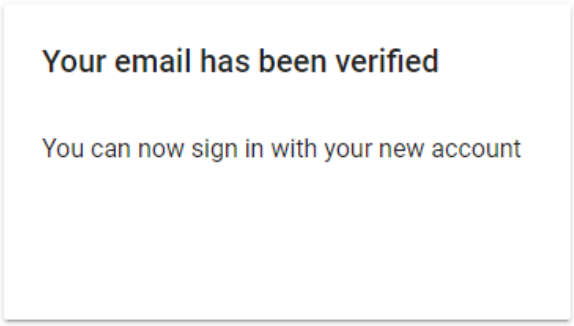


Slika 4.43 Prozor za resetiranje lozinke

Nakon uspješne registracije (Slika 4.44) korisnik mora prvo potvrditi svoj e-mail račun kako bi se mogao prijaviti.

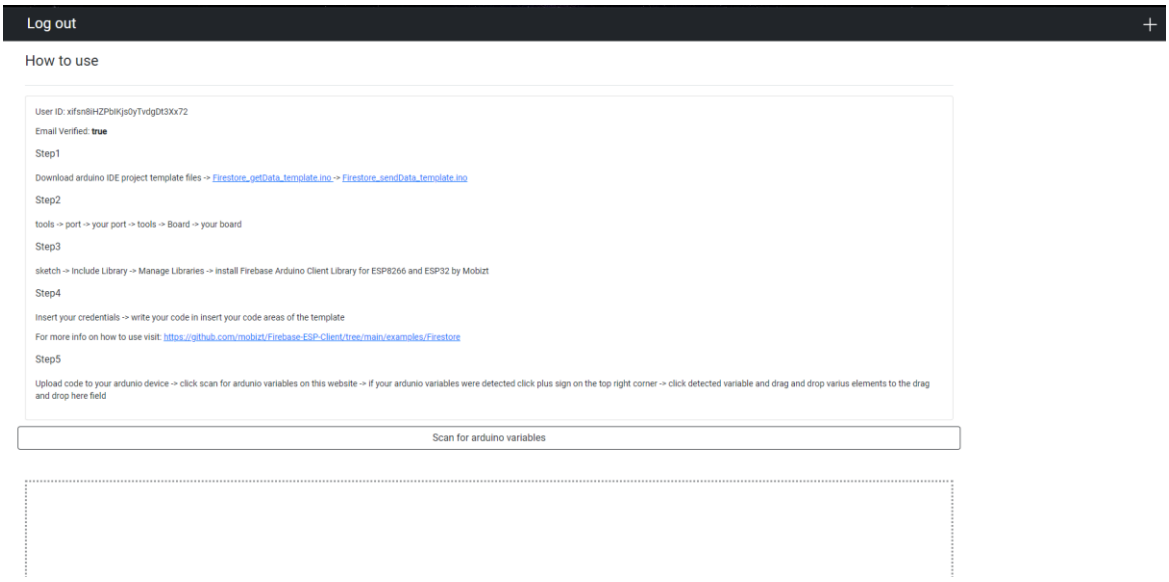


Slika 4.44 prozor nakon uspješne registracije

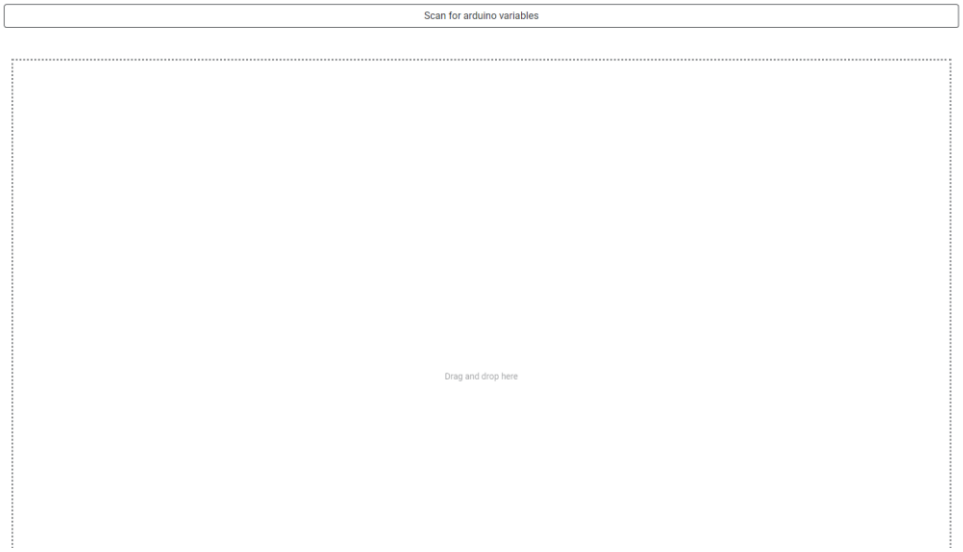


Slika 4.45 Prozor za uspješnu potvrdu e-pošte

Nakon uspješne prijave otvara se glavni prozor (Slika 4.46) koji sadrži navigacijsku traku, upute za korištenje, gumb za dohvaćanje vrijednosti Arduino varijabli te prostor za rad (Slika 4.47).

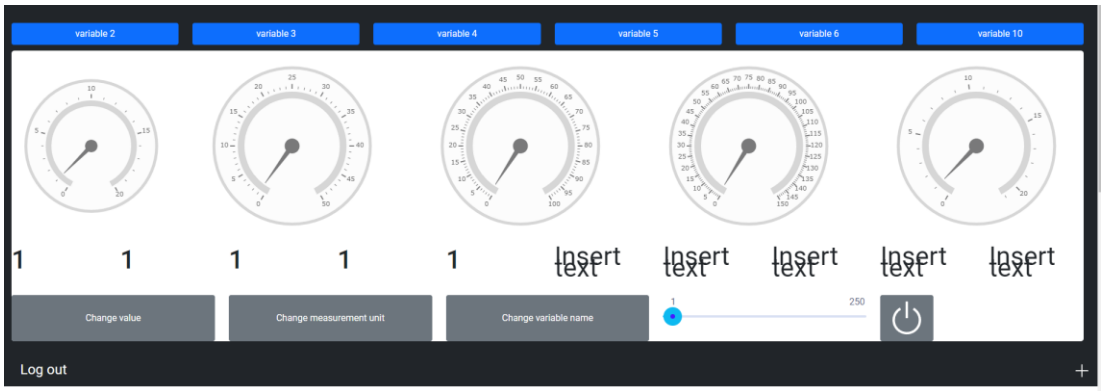


Slika 4.46 Glavni prozor web aplikacije

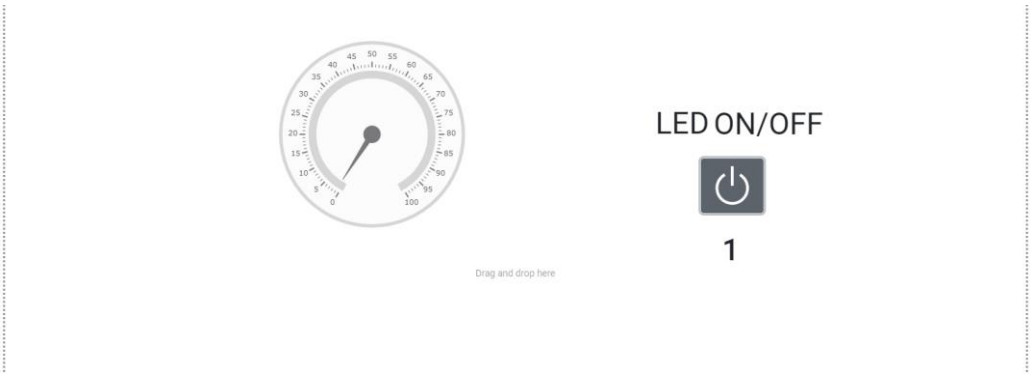


Slika 4.47 Radni prostor

Navigacijska traka se sastoji od gumba za odjavu koji preusmjerava korisnika na prozor za prijavu i „plus“ gumba koji otvara novu listu gumbova koji predstavljaju ne null varijable. Pritiskom na gumb bilo koje varijable otvara se ladica koja sadrži sve tipove elemenata (Slika 4.48) koji korisnik može povući i ispustiti na radni prostor (Slika 4.49).



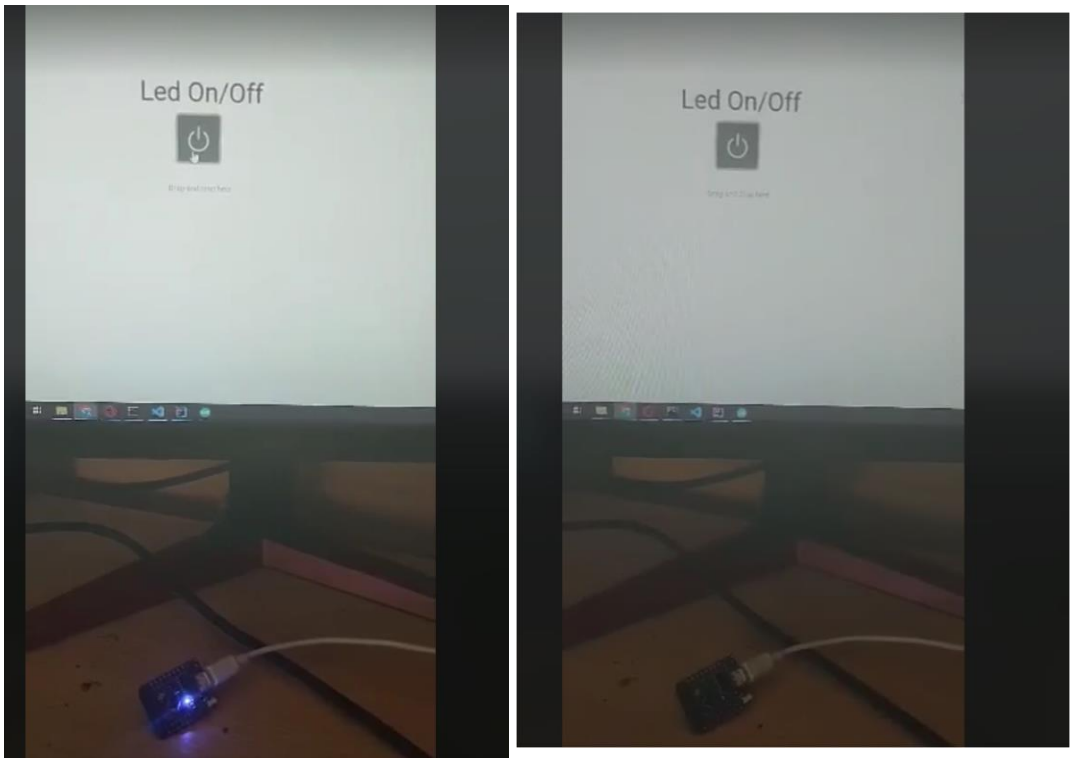
Slika 4.48 Ladica s elementima koji imaju mogućnost povlačenja i ispuštavanja



Slika 4.49 Primjer nekih povućenih elemenata na radnom prostoru

```
  // Write code in this if statment to get data from server

if (Firebase.Firestore.getDocument(&fbdo, FIREBASE_PROJECT_ID, "", documentPath.c_str())){
    FirebaseJson payload;
    payload.setJsonData(fbdo.payload().c_str());
    FirebaseJsonData jsonData;
    payload.get(jsonData, "fields/var2/stringValue", true); //Get var2 data example
    String str = jsonData.stringValue;
    if (str == "1") // Led ON/Of example
    digitalWrite(LED_BUILTIN, HIGH);
    else
    digitalWrite(LED_BUILTIN, LOW);
    }
else
    Serial.println(fbdo.errorReason());
```

Slika 4.50 Primjer Arduino IDE kôda za primanje podataka s baze podataka te paljenje ili gašenje LED diode mikrokontrolera ovisno o vrijednosti primljenih podataka



Slika 4.51 Primjer za paljenje ili gašenje LED diode mikrokontrolera koristeći gumba na web aplikaciji

5. ZAKLJUČAK

Za izradu ovog završnog rada bilo je potrebno koristiti mnoge tehnologije koje su zajedničkim međudjelovanjem omogućile uspješnu realizaciju sustava za autentikaciju i razmjenu podataka između mikrokontrolera i web aplikacije. Za bazu podataka i za sustav autentikacije Firestore se pokazao kao odličan izbor zbog jednostavnosti korištenja i implementiranja u Angular i Spring Boot. Pogodnosti uočene korištenjem Spring Boota za izradu pozadinske aplikacije su jednostavno dodavanje ovisnosti, dovoljna kontrola nad procesima i lagana integracija s bazama podataka. Spring Boot aplikacije koriste malo resursa pri pokretanju što ga čini dobrim Java okvirom za kreiranje manjih web aplikacija na osobnom računalu. Iako je Angular vrlo kompleksan zbog svoje popularnosti postoji mnogo vanjskih biblioteka koje olakšavaju proces izrade Angular aplikacije i proširivaju mogućnosti Angular-a. Upravo zbog svoje fleksibilnosti i mnoštvo korisnih mogućnosti Angular se pokazao kao dobar okvir za izradu sučelja aplikacije. Arduino IDE je program napravljen za rad s Arduino čipovima i puno je jednostavniji i pristupačniji pristup programiranja Arduino čipova od pisanja Assembler kôda. Arduino IDE je program otvorenog kôda i zbog toga postoje mnoge vanjske biblioteke koje znatno proširivaju mogućnosti programa.

LITERATURA

[1] Primjer Blynk korisničkog sučelja, dostupno na: https://blynk.io [pristupljeno 23. lipnja 2022.]

[2] Web aplikacija za izradu sučelja i povezivanje na Arduino uređaj RemoteXY: Arduino control aplikacije dostupno na: https://remotexy.com [pristupljeno 11. kolovoza 2022.]

[3] Introduction to Angular concepts, dostupno na: https://angular.io/guide/architecture [pristupljeno 23. lipnja 2022.]

[4] HTML, dostupno na: https://hr.wikipedia.org/wiki/HTML [pristupljeno 23. lipnja 2022.]

[5] CSS, dostupno na: https://hr.wikipedia.org/wiki/CSS [pristupljeno 23. lipnja 2022.]

[6] Bootstrap (front-end framework), dostupno na: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) [pristupljeno 23. lipnja 2022.]

[7] TypeScript, dostupno na: https://en.wikipedia.org/wiki/TypeScript [pristupljeno 23. lipnja 2022.]

[8] Spring Boot - Introduction, dostupno na: https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.htm [pristupljeno 23. lipnja 2022.]

[9] Spring Boot, dostupno na: https://spring.io/projects/spring-boot#overview [pristupljeno 23. lipnja 2022.]

[10] Spring Initializr, dostupno na: https://start.spring.io/ [pristupljeno 23. lipnja 2022.]

[11] Java_(programming_language), dostupno na: https://en.wikipedia.org/wiki/Java_(programming_language) [pristupljeno 23. lipnja 2022.]

[12] Firebase, dostupno na: https://firebase.google.com [pristupljeno 23. lipnja 2022.]

[13] Arduino Integrated Development Environment (IDE) v1, dostupno na: https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics [pristupljeno 23. lipnja 2022.]

[14] REST, dostupno na: https://en.wikipedia.org/wiki/Representational_state_transfer [pristupljeno 23. lipnja 2022.]

[15] JSON, dostupno na: https://www.json.org/json-en.html [pristupljeno 19. srpnja 2022.]

SAŽETAK

Glavni zadatak ovog završnog rada je bila izrada web aplikacije koja omogućuje korisniku udaljeno upravljanje i nadziranje mikrokontrolera preko interneta. Tehnologije korištene u izradi rada su teorijski objašnjene i njihovo korištenje je prikazano u radu. Izrada rada je podijeljena u više dijelova gdje svaki dio ima različitu ulogu i izrađen je korištenjem drugačijih tehnologija. Za bazu podataka i sustav autentikacije korištena je Firebase platforma. Izgradnja pozadinske aplikacije odvijala se korištenjem Spring Boot okvira u IntelliJ razvojnom okruženju pisanjem Java kôda. Sučelje je izgrađeno u Angular okviru pisanjem HTML, TypeScript i CSS kôda korištenjem Visual Studio Code razvojnog okruženja. Arduino IDE predložak kreiran je pisanjem C kôda. Funkcionalnost i izgled web aplikacije opisom i slikama.

Ključne riječi: Angular, Arduino, Firestore, Spring Boot, web aplikacija

ABSTRACT

The main task of this final paper was the creation of a web application that enables the user to remotely control and monitor the microcontroller via the Internet. The technologies used in the creation of this final paper are theoretically explained and their use is shown in this final paper. The creation of this work is divided into several parts where each part has a different role and is made using different technologies. The Firebase platform was used for the database and authentication system. The backend was built using the Spring Boot framework in the IntelliJ development environment by writing Java code. The frontend was built in the Angular framework by writing HTML, TypeScript, and CSS code in the Visual Studio Code development environment. Arduino IDE template is created by writing C code. The functionality and appearance of the web application are presented with a description and images.

Keywords: Angular, Arduino, Firestore, Spring Boot, web application

PRILOG A: Poveznica na gitHub repozitorij

Link na gitHub repozitoriji koji sadrži izvorni kod web aplikacije:
https://github.com/aomazic/Antonio_Omazic-_Zavrsni