**VELEUČILIŠTE U RIJECI**

**Poslovni odjel**

Specijalistički diplomski stručni studij Informacijske tehnologije u poslovnim sustavima Kolegij: Programsko inženjerstvo

Mattea Galich, Valentina Bačić

**HGSS dojavni centar**

Rijeka, 2021.

**VELEUČILIŠTE U RIJECI**

**Poslovni odjel**

Specijalistički diplomski stručni studij Informacijske tehnologije u poslovnim sustavima Kolegij: Programsko inženjerstvo

**HGSS dojavni centar**

MENTOR: STUDENTICE:

dr. sc. Marin Kaluža, prof. visoke škole Mattea Galich

Valentina Bačić

Rijeka, 2021.

# **Opis sustava - Valentina**

Prijaviti nesreću HGSS-u možemo telefonski preko broja 112, no u nekim slučajevima zna biti teško odrediti lokaciju unesrećene osobe te smo se zbog toga odlučile za izradu aplikacije HGSS dojavnog centra, koja preko obrasca koji korisnik ispunjava automatski povlači trenutnu lokaciju te se podaci šalju u dojavni centar.

Sustav za primanje i prikaz dojava o određenoj nezgodi omogućava operateru brzo slanje slobodnog tima na lokaciju gdje je potreban na temelju zaprimljene dojave od strane korisnika.

**Funkcionalnosti**:

1. Korisnik šalje dojavu preko obrasca (lokacija, broj mobitela, opis nezgode i sl.)

2. Operater šalje tim koji je slobodan na lokaciju koju je poslao korisnik (ako nije dano dovoljno podataka operater može nazvati osobu koja je prijavila nesreću, pošto je jedan od podataka broj mobitela)

3. Registracija/login operatera

3. Operater može pregledati sve informacije o svakoj zaprimljenoj dojavi

**Akteri**

- akter 1: Korisnik (civil)

- akter 2: Operater u centru

# **Arhitektura sustava - Mattea**

Tehnologije koje će se koristiti u izgradnji aplikacije su **React** te **Firebase**. Aplikacija će biti izgrađena sa Visual Studio Code-om te će biti spojena sa GitHub-om.

React je open-soruce, Javascript knjižnica za izgradnju korisničkog sučelja ili UI komponenti. Može se koristiti kao osnova u razvoju mobilnih aplikacija

Firebase je platforma za razvoj programa Backend-as-a-Service (BaaS) koja pruža usluge kao što su baza podataka u stvarnom vremenu, pohrane u oblaku, provjera autentičnosti itd. Pomoću Firebase-a, izrada aplikacije bi trebala teći brže jer nema upravljanja infrastrukturom. Firebase već ima funkcionalnosti koje će se koristiti u izradi aplikacije kao što je baza podataka kako bi sve informacije bile prikazane pravovremeno. Time će korisnik imati informacije dostupne odmah nakon promjene ali se i brže kretati kroz aplikaciju.

Web aplikaciji za slanje dojave HGSS-u pristupa korisnik na mobilnom uređaju. Na brz i jednostavan način unosi podatke vezane uz dojavu, a sustav automatski povlači lokaciju uređaja. Korisničkom akcijom se šalje informacija o dojavi na web i database poslužitelj.

Operater u centru za dojave prima informacije o dojavi kroz web aplikaciju na svojoj radnoj stanici. Na temelju primljene dojave operater obavještava slobodan tim o izlasku na teren.

Diagram

Description automatically generated

Izradile: autorice

# **Dijagram akcija - Mattea**

**Graphical user interface, text, application, chat or text message

Description automatically generated**

Izradile: autorice

# **Sekvencijalni dijagram - Mattea**

**A picture containing text

Description automatically generated**

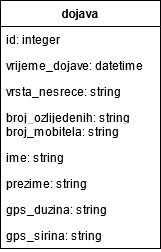
Izradile: autorice

**A picture containing text

Description automatically generated**

Izradile: autorice

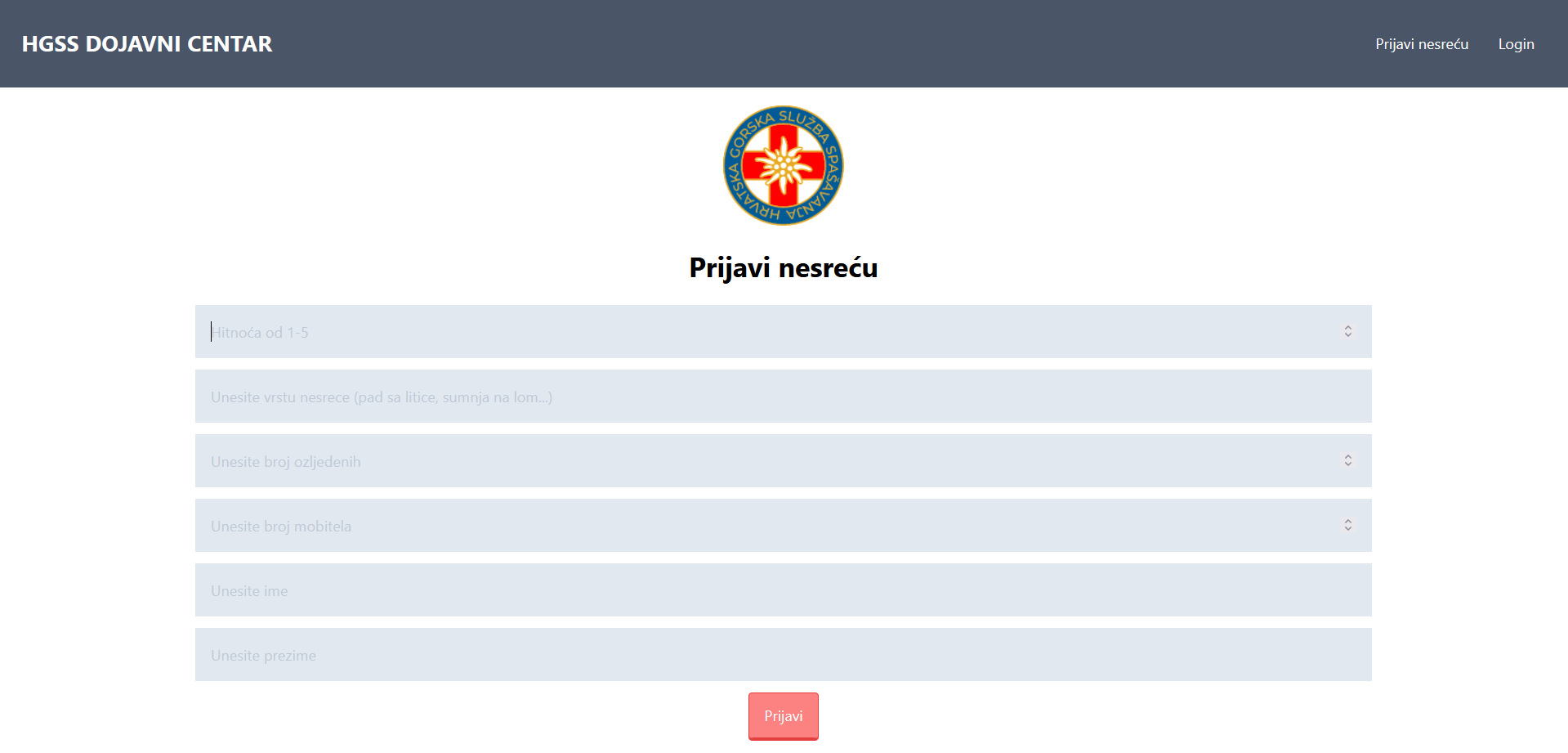
# **Dijagram klasa - Valentina**



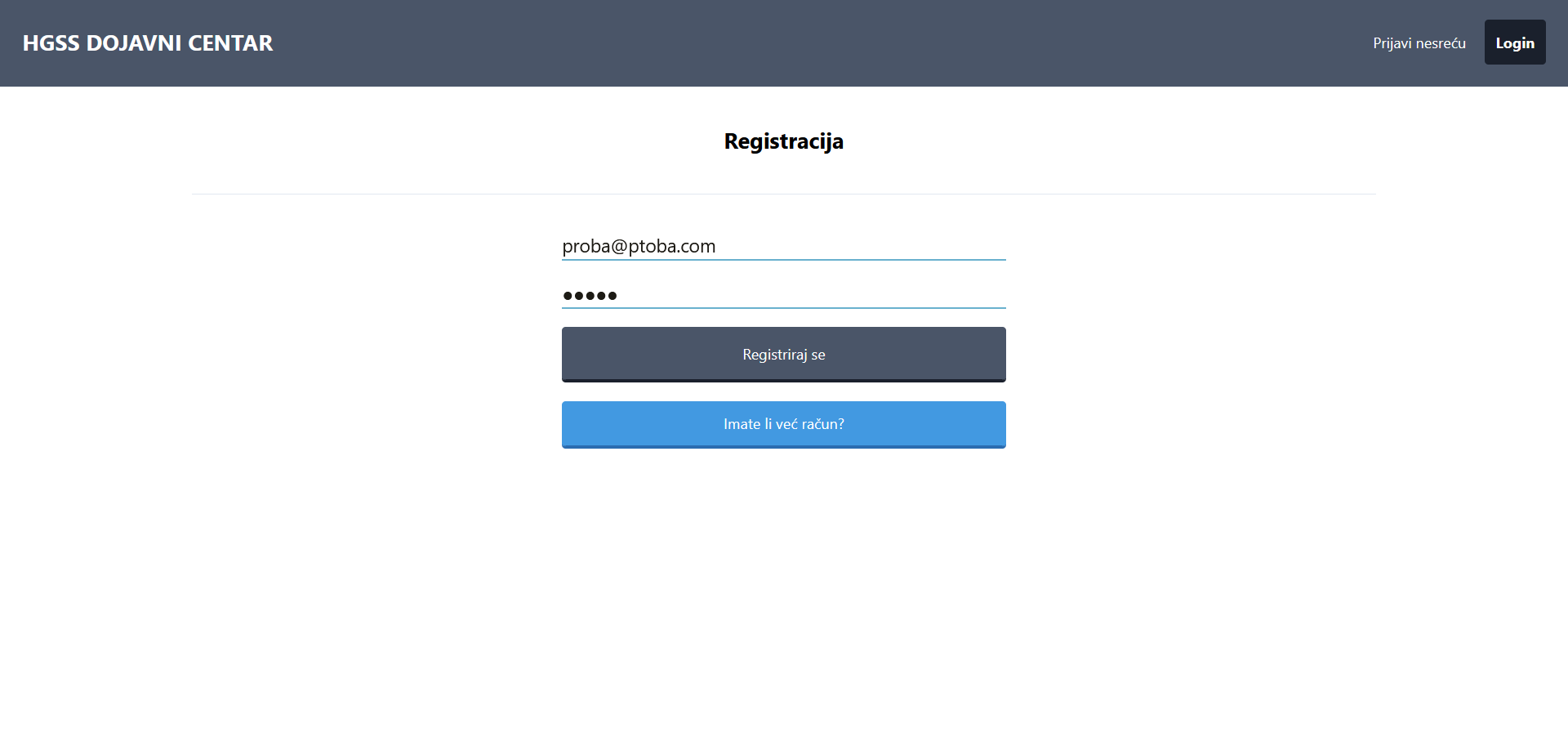
Izradile: autorice

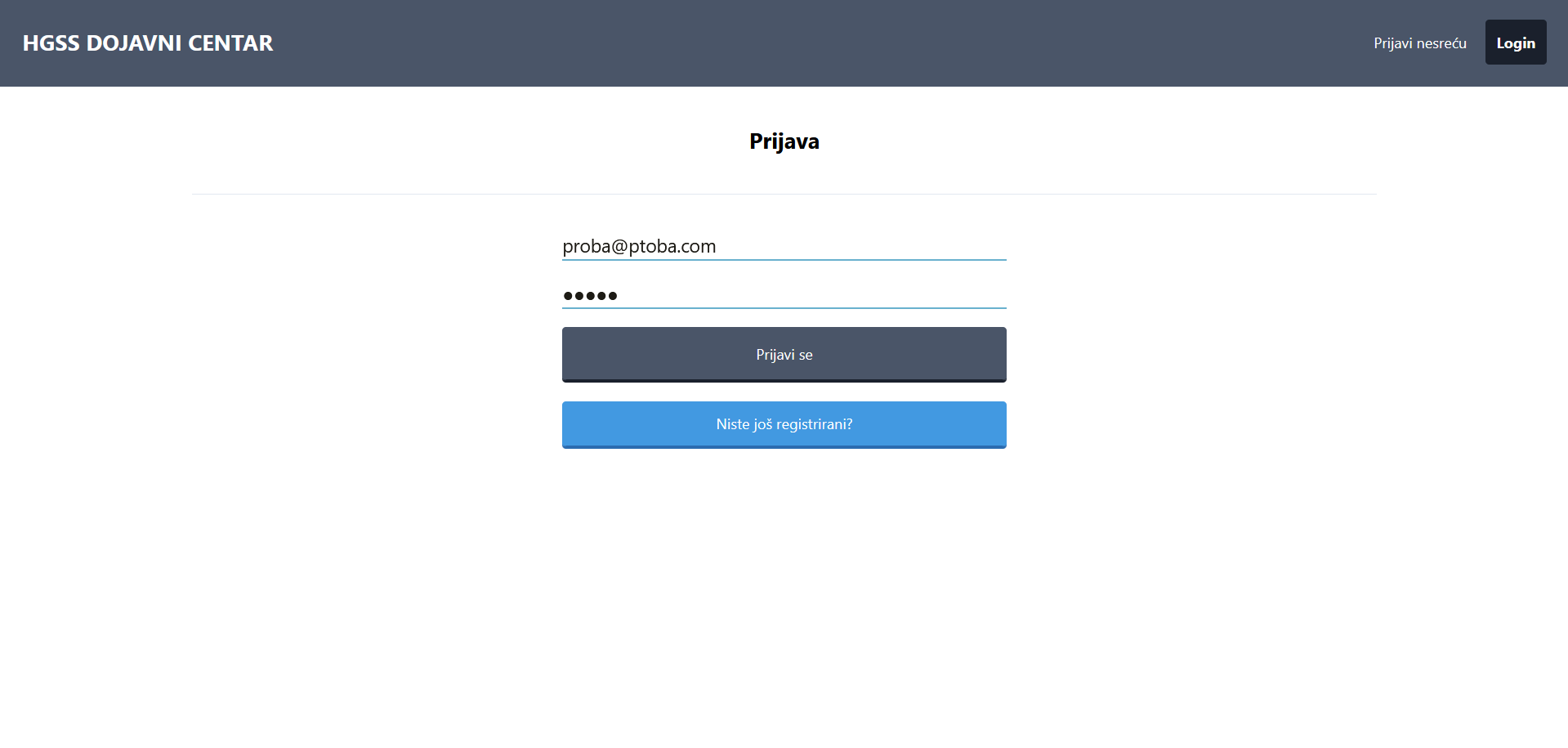
# **DIZAJN KORISNIČKOG SUČELJA - Valentina**

Početna stranica gdje korisnik unosi novu dojavu. U kutu desno se nalazi Login gdje operater ulazi na stranicu za prijavu.



Prozori za registraciju i prijavu za operatera





Korisničko sučelje gdje operater vidi pristigle dojave



# **IZRAĐENE KOMPONENTE - Valentina**

Login.jsx

import React from "react";

import { db, auth } from "../firebase";

import { withRouter } from "react-router-dom";

const Login = (props) => {

  const [email, setEmail] = React.useState("");

  const [pass, setPass] = React.useState("");

  const [msg, setMsg] = React.useState("");

  const [registr, setRegistr] = React.useState(true);

  const registriranjeKorisnika = React.useCallback(async () => {

    try {

      const res = await auth.createUserWithEmailAndPassword(email, pass);

      await db.collection("users").doc(res.user.uid).set({

        email: res.user.email,

        uid: res.user.uid,

      });

      await db.collection(res.user.uid).add({

        name: "primjer",

        fecha: Date.now(),

      });

      setEmail("");

      setPass("");

      setMsg("");

      props.history.push("/admin");

    } catch (error) {

      console.log(error);

      setMsg(error.message);

    }

  }, [email, pass, props.history]);

  const Sesija = React.useCallback(async () => {

    try {

      await auth.signInWithEmailAndPassword(email, pass);

      setEmail("");

      setPass("");

      setMsg("");

      props.history.push("/admin");

    } catch (error) {

      setMsg(error.message);

      return;

    }

  }, [email, pass, props.history]);

  //Funkcija provjere valjanosti podataka

  const provjeraPodataka = (e) => {

    e.preventDefault();

    if (!email.trim()) {

      setMsg("Unesite email adresu");

      return;

    }

    if (!pass.trim()) {

      setMsg("Unesite lozinku");

      return;

    }

    if (pass.length < 6) {

      setMsg("Lozinka mora imati minimalno 6 znakova");

      return;

    }

    setMsg("");

    console.log("Uredu je sve");

    if (registr) {

      registriranjeKorisnika();

    } else {

      Sesija();

    }

  };

  return (

    <div className="container mx-auto mt-10">

      <h3 className="text-center font-bold text-2xl mb-10">

        {registr ? "Registracija" : "Prijava"}

      </h3>

      <hr />

      <form

        onSubmit={provjeraPodataka}

        className="mt-10 w-11/12 md:w-5/6 lg:w-3/5 xl:w-1/2 mx-auto"

      >

        {msg.length > 0 ? (

          <div className="p-4 bg-red-300 text-red-600 block w-3/4 mx-auto mb-4">

            {msg}

          </div>

        ) : null}

        <input

          type="email"

          placeholder="Unesite email adresu"

          name="email"

          className="focus:outline-none bg-transparent text-xl border-b-2 border-blue-400 w-3/4 mx-auto mb-5 block"

          onChange={(e) => setEmail(e.target.value)}

          value={email}

        />

        <input

          type="password"

          placeholder="Unesite lozinku"

          name="password"

          className="focus:outline-none bg-transparent text-xl border-b-2 border-blue-400 w-3/4 mx-auto mb-5 block"

          onChange={(e) => setPass(e.target.value)}

          value={pass}

        />

        <button

          type="submit"

          className="block p-4 border-b-4 bg-gray-700 border-gray-900 rounded text-white w-3/4 mb-5 mx-auto"

        >

          {registr ? "Registriraj se" : "Prijavi se"}

        </button>

        <button

          type="button"

          onClick={() => setRegistr(!registr)}

          className="block p-3 border-b-4 bg-blue-500 border-blue-700 rounded text-white w-3/4 mx-auto"

        >

          {registr ? "Imate li već račun?" : "Niste još registrirani?"}

        </button>

      </form>

    </div>

  );

};

export default withRouter(Login);

UnosDojave.jsx

import React, { useEffect } from "react";

import { db } from "../firebase";

import "moment/locale/es";

import Logo from '../logo.png';

function UnosDojave() {

  const [hitnoca,setHitnoca] = React.useState();

  const [vrsta, setvrsta] = React.useState("");

  const [ozljeden, setozljeden] = React.useState("");

  const [broj, setbroj] = React.useState("");

  const [ime, setime] = React.useState("");

  const [prezime, setprezime] = React.useState("");

  const [lat,setLat]=React.useState();

  const [lng,setLng]=React.useState();

  useEffect (() => {

    navigator.geolocation.getCurrentPosition(function(position) {

      setLat(position.coords.latitude);

      setLng(position.coords.longitude);

      console.log("Latitude is :", position.coords.latitude);

      console.log("Longitude is :", position.coords.longitude);

    });

  })

  const dodavanjedojave = async (e) => {

    e.preventDefault();

    if (!hitnoca.trim()) {

      console.log("Prazno je, popunite polje");

      return;

    };

      var hitnoca\_nesrece = hitnoca

      var vrsta\_nesrece= vrsta

      var broj\_ozljedenih= ozljeden

      var broj\_mobitela= broj

      var ime\_k= ime

      var prezime\_k= prezime

      var today= new Date();

      var dd = String(today.getDate()).padStart(2, '0');

      var mm = String(today.getMonth() + 1).padStart(2, '0');

      var yyyy = today.getFullYear();

      var datum = mm + '/' + dd + '/' + yyyy;

      var geo={latitude: lat,longitude: lng}

      db.collection('test').add({ hitnoca\_nesrece,vrsta\_nesrece,broj\_ozljedenih,broj\_mobitela,ime\_k,prezime\_k,datum,geo });

      setHitnoca("");

      setvrsta("");

      setozljeden("");

      setbroj("");

      setime("");

      setprezime("");

  };

  return (

    <form

            onSubmit={dodavanjedojave}

            className="w-3/4 mx-auto"

          >

<div>

<h1 className="font-bold text-center mx-auto text-3xl my-5">

<a href="http://www.hgss.hr/"><img src={Logo} className="App-logo" alt="Logo" className="text-center mx-auto text-5xl my-5"/> </a>

        Prijavi nesreću

      </h1>

    </div>

      <input

              type="number"

              name="name"

              className="focus:outline-none w-full bg-gray-300 p-4 border mb-3"

              placeholder="Hitnoća od 1-5"

              onChange={(e) => setHitnoca(e.target.value)}

              value={hitnoca}

              autoFocus

            />

      <input

              type="text"

              name="vrsta\_nesrece"

              className="focus:outline-none w-full bg-gray-300 p-4 border mb-3"

              placeholder="Unesite vrstu nesrece (pad sa litice, sumnja na lom...)"

              onChange={(e) => setvrsta(e.target.value)}

              value={vrsta}

            />

<input

              type="number"

              name="broj\_ozljedenih"

              className="focus:outline-none w-full bg-gray-300 p-4 border mb-3"

              placeholder="Unesite broj ozljedenih"

              onChange={(e) => setozljeden(e.target.value)}

              value={ozljeden}

            />

<input

              type="number"

              name="broj\_mobitela"

              className="focus:outline-none w-full bg-gray-300 p-4 border mb-3"

              placeholder="Unesite broj mobitela"

              onChange={(e) => setbroj(e.target.value)}

              value={broj}

            />

<input

              type="text"

              name="ime"

              className="focus:outline-none w-full bg-gray-300 p-4 border mb-3"

              placeholder="Unesite ime"

              onChange={(e) => setime(e.target.value)}

              value={ime}

            />

<input

              type="text"

              name="prezime"

              className="focus:outline-none w-full bg-gray-300 p-4 border mb-3"

              placeholder="Unesite prezime"

              onChange={(e) => setprezime(e.target.value)}

              value={prezime}

            />

              <button

                onClick={dodavanjedojave}

                className="block px-4 py-3 mx-auto text-white rounded bg-red-400 border border-b-4 border-red-600"

              >

                Prijavi

              </button>

          </form>

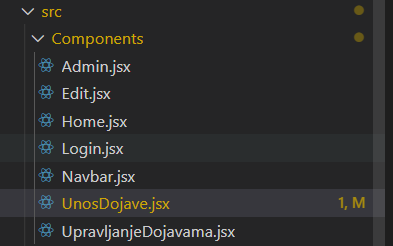
  );

}

export default UnosDojave;

Uz ove dvije gore navedene još su izrađene Admin.jsx gdje se nalazi dio sa upravljanjem dojavama u koji može ući samo ulogirani korisnik. Također su napravljeni Navbar.jsx za navigaciju te Home.jsx na kojoj se nalazi dio iz unosa dojave.

Također je napravljen i Edit.jsx koji je naknadno isključen iz same aplikacije zbog toga što smo zaključile da operater nema zašto mijenjati dojavu korisnika.



# **KORISNIČKE UPUTE - Mattea**

Korisničke upute za korisnika (civila):

1. Potvrditi slanje lokacije uređaja (*lokacija uređaja se šalje u svrhu slanja hitnih službi na lokaciju nesreće. Lokacija uređaja se ne dijeli sa trećim stranama. Lokacija uređaja se šalje samo jednom, prilikom slanja dojave*)
2. Upisati podatke (*Upisani podaci olakšavaju i ubrzavaju proces slanja hitnih službi na lokaciju nesreće. Navedene informacije su obaveze, jer na osnovu poslanih informacija, dežurni operater u centru za obavješćivanje procjenjuje koje hitne službe je potrebno poslati na lokaciju nesreće)*
3. Odabrati akciju *Prijavi*

Korisničke upute za operatera:

1. Registracija/prijava
2. U popisu dojava nalazi se sažetak glavnih informacija o primljenim dojavama: Dojave su sortiranje od najmlađe prema najstarijoj.

# **SIGURNOST SOFTVERA - Valentina**

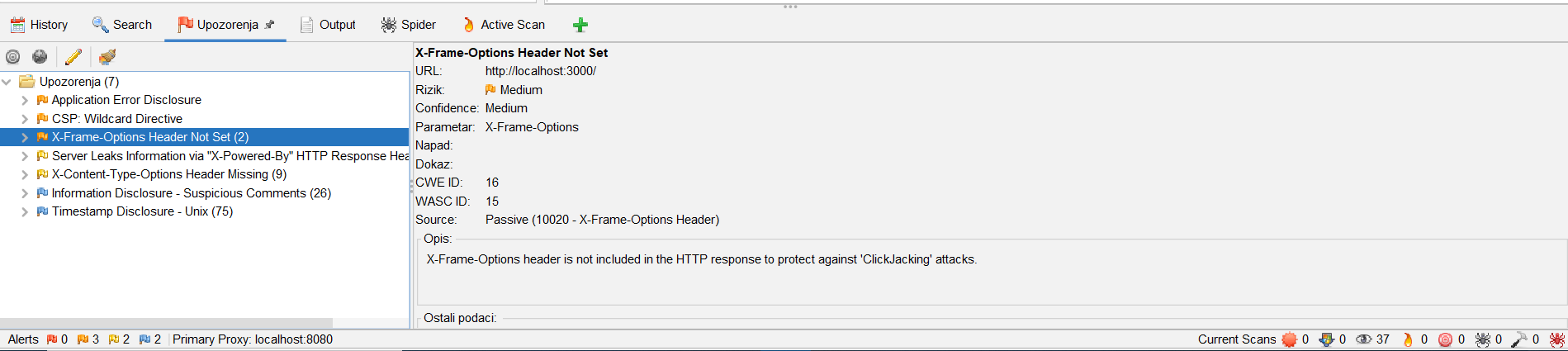
Softverska sigurnost je proces dizajniranja, izrade i testiranja softvera za sigurnost gdje softver sam identificira i uklanja probleme. U osnovi, ideja softverske sigurnosti uključuje proaktivni pristup, koji se odvija unutar faze prije uvođenja. Najbolji postupci utječu na izgradnju jednostavnijeg obrane koda. Nažalost, mnoge tvrtke za razvoj softvera nisu svjesne važnosti najboljih sigurnosnih praksi koje su praćene nepostojanjem znanja o tome kako najbolje razvijati softver temeljen na temeljnim sigurnosnim načelima.

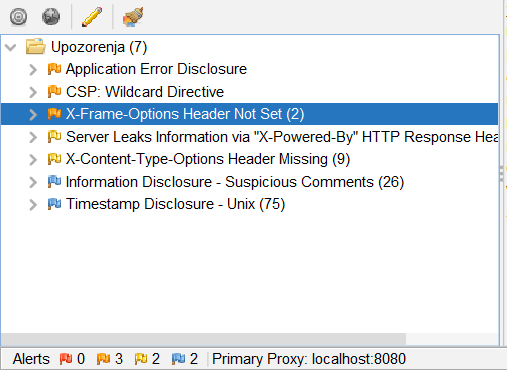
Zbog najbolje prakse sigurnosti, potrebno se usredotočiti na:

1. **Mehanizam provjere autentičnosti i autorizacija**: Ovaj postupak uključuje sigurno dizajniran sustav koji sprječava korisnika da mijenja identitet bez ponovne provjere autentičnosti, višečlanu provjeru autentičnosti, sigurnosni nadzorni mehanizam, dozvole resursa, dozvole datoteke i baze podataka.
2. **Validacija podataka**: U životnom ciklusu razvoja potrebno se uvijek usredotočiti na postupak provjere podataka koji uključuje centralizirane mehanizme provjere, pretvaranje podataka u kanonski oblik, korištenje zajedničkih biblioteka primitivaca za provjeru valjanosti, primjenu tipova jezične razine da bi se zabilježile pretpostavke o valjanosti podataka.
3. **Kriptografija**: Kriptografija je jedan od najvažnijih alata za izgradnju sigurnih sustava. Pravilnim korištenjem kriptografije osigurava se povjerljivost podataka, štite podatci od neovlaštene izmjene i potvrđuje izvor podataka. Kriptografija također može omogućiti i mnoge druge sigurnosne ciljeve.

Testiranje sigurnosti softvera odrađeno je korištenjem besplatnog alata *Zed Attack Proxy*. ZAP je sigurnosti skener s otvorenim kodom. Neke od značajki su: presretanje proxy poslužitelja, tradicionalne i AJAX web *spideri* , automatizirani skener, pasivni skener, prisilno pregledavanje.

Rezultati testiranja:





* **Application Error Disclosure** - sadrži poruku o pogrešci / upozorenju koja može otkriti osjetljive informacije poput lokacije datoteke. Te se informacije mogu koristiti za pokretanje daljnjih napada na web aplikaciju.
* **CSP: Wildcard Directive** - ne postoji nikakav „Content-Security-Policy“ instaliran na web aplikaciji, primarna svrha mu je detekcija i ublažavanje data injection napada
* nije uključen **X-Frame-Options header**, koji štiti od napada nasumičnim klikanjem.
* **Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)-** greška niske razine koja govori kako postoji mogućnost od proboj informacija iz baze podataka
* **X-Content-Type-Options Header Missing** - greška niske razine koja ukazuje da ne postoji nikakav x-content - sprječava neželjeno pretraživanje od strane potencijalnog napadača

Prijetnje identificiranje ZAP-om u ovoj verziji aplikacije nisu riješene.