

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli



Fakultet informatike u Puli

TIM 179/22 : TravelHelper

Smjer : Informatika

Kolegij : Programsko inženjerstvo

Mentor : doc. dr. sc. Nikola Tanković

Asistent : Toni Starčić, mag. inf.

Pula, 2023. rujan

Contents

Sadržaj	2
1. Uvod i motivacija	2
2. SWOT analiza	2
Snage (Strengths)	2
Slabosti (Weaknesses)	3
Prilike (Opportunities)	3
Prijetnje (Threats)	3
3. Razrada funkcionalnosti	4
3.1. Use-case dijagram	4
3.2. Prototip sučelja	4
3.2.1. LogIn/Prijava	5
3.2.2. Home	7
3.2.3. Prikaz destinacije – Administrator	8
3.2.4. Prikaz profila	9
3.2.5. Pregled favorita	10
3.3. Class dijagram	11
4. Implementacija	11
4.1. Prijava i registracija korisnika – funkcionalnost	12
4.2. Prikaz i filter svih destinacija- funkcionalnost	13
4.3. Pregled pojedinog putovanja – funkcionalnost	15
4.4. Uređivanje i dodavanje destinacije – funkcionalnost (Administrator)	15
4.5. Micanje i dodavanje destinacija u favorite – funkcionalnost	17
4.6. Uređivanje profila – funkcionalnost	18
4.7. Prikaz favorita - funkcionalnost	20
4.8. Destinacija.vue – komponenta	21

Sadržaj

Ova dokumentacija sadržava sve informacije o izradi web-aplikacije i o samoj aplikaciji TravelHelper. U dokumentaciji će se opisati funkcionalnosti web-aplikacije, koncept aplikacije, kako je došlo do ideje da se napravi takva aplikacija, kako je odabir ciljane publike utjecao na izradu aplikacije i koja je to ciljana publika, razraditi će se UML i Class dijagrami i uz to sve koji su programi pomogli i bili korišteni u izradi aplikacije.

1. Uvod i motivacija

TravelHelper je web aplikacija namijenjena ljubiteljima putovanja. Glavna svrha ove aplikacije je pomoći korisnicima da pronađu idealnu destinaciju za putovanje koja odgovara njihovim osobnim preferencama. Na primjer, ako korisnik želi putovati, ali nema konkretnu destinaciju na umu, aplikacija omogućuje korisniku da odabere određene karakteristike destinacije koje su mu važne.

Korisnik može birati između različitih karakteristika, kao što su npr. specifična kultura i planine. Nakon što unese svoje preferencije, aplikacija će prikazati destinacije koje ispunjavaju te kriterije. Također, ako korisnik ima određenu državu na umu, može dodatno unijeti ime te države kako bi dobio popis destinacija koje odgovaraju njegovim preferencijama unutar te države. Korisnik se još usput može i napraviti račun gdje će biti spremljene odabrane destinacije koje želi posjetiti pa da kasnije si lakše može odabrati pravu destinaciju za putovanje.

Glavni motiv za razvoj ove aplikacije bio je omogućiti korisnicima da pronađu idealnu destinaciju za putovanje temeljem vlastitih osobnih preferencija, umjesto da se ograniče budžetom ili popularnošću destinacije. Naime, mnoge turističke agencije često promoviraju destinacije prema njihovoj popularnosti ili cijeni, što često narušava pravi smisao putovanja ili odmora. Odmor bi trebao biti vrijeme za opuštanje, uživanje i obnovu, kako bi se osoba osjećala osvježeno i spremno za povratak svakodnevnim obvezama i aktivnostima. Svatko ima svoju definiciju odmora i mjesta gdje se želi odmarati, stoga sam razvila aplikaciju u kojoj osoba sama odabire kakvu destinaciju želi i temeljem toga bira gdje će provesti svoj odmor.

2. SWOT analiza

Snage (Strengths)

- Jedinstvena i inovativna ideja koja omogućava korisnicima personalizirano planiranje putovanja.
- User-friendly sučelje.

- Velika baza podataka o različitim destinacijama.

Aplikacija se bazira na tome na kakvu destinaciju korisnik želi otputovati, a ne što je trenutno aktualno ili najjeftinije i ova jedinstvena usluga može privući korisnike koji traže personalizirana iskustva. Namijenjena je da ju svatko može koristiti i zaista je jednostavna za korištenje. U našoj bazi podataka ima dovoljan broj destinacija da korisnik ima što više izbora i opcija za odabir.

Slabosti (Weaknesses)

- Početna faza razvoja.
- Potreba za stalnim ažuriranjem informacija o destinacijama i atrakcijama.

Aplikacija je taman u početnoj fazi razvoja što znači da je teže privući korisnike. Svakim danom se mijenja svijet, a tako i gradovi i destinacije, tako da se uvijek mora istraživati i raspraviti da bi destinacije koje prikazujemo na aplikacije uvijek bile ažurne.

Prilike (Opportunities)

- Rastući interes ljudi za personaliziranim putovanjima.
- Suradnja s turističkim agencijama i partnerstva za promociju.
- Mogućnost dodavanja funkcionalnosti kao što su rezervacije hotela, letova i izleta.
- Globalni rast industrije turizma i putovanja.

Sve veći broj putnika traži jedinstvena iskustva i personalizirane putne planove, što se uklapa u našu aplikaciju. Možemo surađivati s turističkim agencijama tako da kod nas korisnik pronađe željenu destinaciju i mi preporučimo koje turističke agencije mogu koristiti za realizaciju putovanja. Nove funkcionalnosti kao mogućnosti rezervacije hotela, letova i izleta unutar aplikacije može proširiti našu ciljanu publiku i povećati prihode.

Prijetnje (Threats)

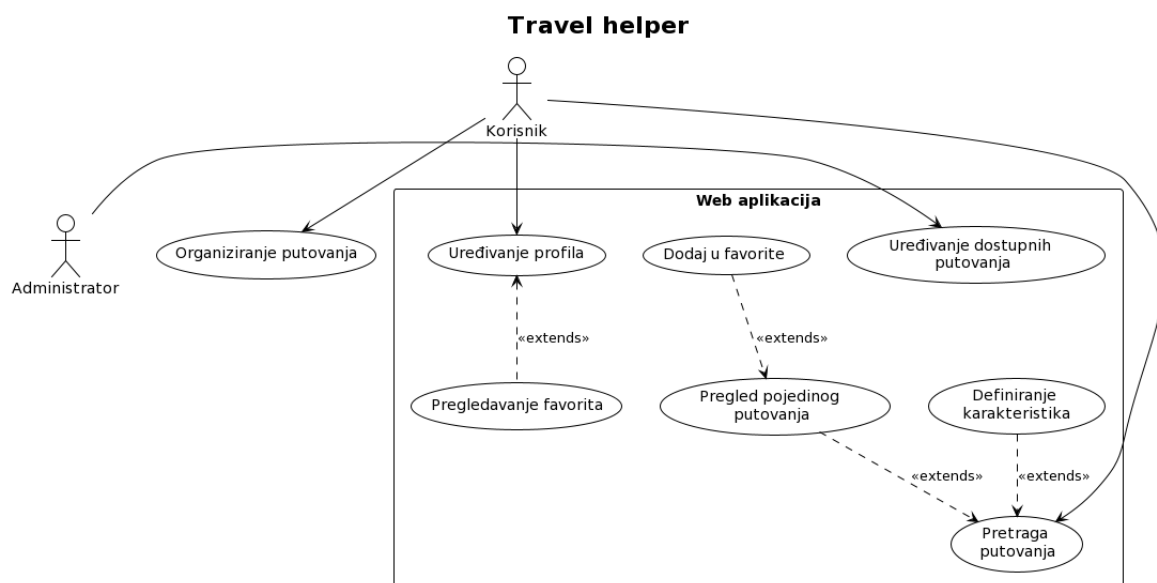
- Snažna konkurencija u industriji aplikacija za putovanja.
- Ekonomska nestabilnost koja može smanjiti putničke budžete.
- Nepredvidljive situacije kao što su prirodne katastrofe koje mogu utjecati na planiranje putovanja.

Tržište aplikacija za putovanja je iznimno konkurentno, a najviše su poznati oni kod kojih se mogu organizirati cijelo putovanje. Za to je potrebno poboljšati aplikaciju sa dodatnim

funkcionalnostima gdje se unutar aplikacije može organizirati putovanje. Nepredvidljive ekonomske situacije, poput recesije ili pandemije, mogu smanjiti putničke budžete i smanjiti potražnju za putovanjima. Također i prirodne katastrofe ili političke nestabilnosti u destinacijama mogu utjecati na planiranje putovanja i sigurnost korisnika.

3. Razrada funkcionalnosti

3.1. Use-case dijagram



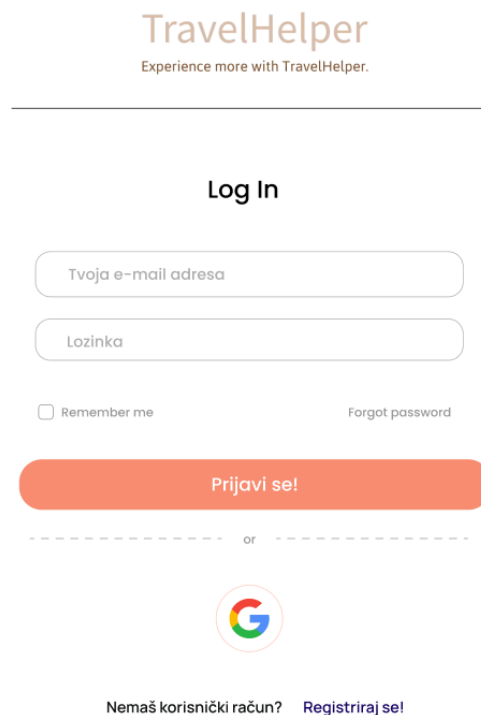
Slika 1: Use-case dijagram

Use-case dijagram prikazuje sudionike i korištenje naše web-aplikacije TravelHelper. Sudionici su korisnik i administrator. Korisnik izvan aplikacije planira i organizira svoje putovanje, dok unutar naše aplikacije prvo može kreirati profil, a i ne mora. Nakon toga može pretraživati destinacije uz pomoć njezinih karakteristika, može ih pretraživati po imenu grada ili države. Korisnik kada pronađe destinaciju koja mu se sviđa može kliknuti na nju i vidjeti specifikacije destinacije. Te iste destinacije može dodavati u favorite i pregledavati sve favorite koje je spremio i micati ih. Korisnik može pregledavati i uređivati svoj profil. S druge strane, administrator ima šire ovlasti. Osim što može obavljati sve radnje koje i korisnik može, administrator ima pristup upravljanju sadržajem aplikacije. To uključuje uređivanje postojećih destinacija i dodavanje novih destinacija u bazu podataka. Kada administrator pregledava destinaciju, ponuđene su mu opcije uređivanja ili brisanja destinacije. Na navigaciji administratoru je prikazan gumb Nova destinacija i klikom na taj gumb može stvoriti novu destinaciju. Nakon uređivanja ili dodavanja destinacije, administrator se vraća na pregled svih destinacija.

3.2. Prototip sučelja

Prototip sučelja je napravljen u Figma i to mi je stvarno pomoglo pri izradi same web-aplikacije. Prvo vizualiziranje u Figma olakšalo mi je razumijevanje kako će aplikacija izgledati, što je onda olakšalo uređivanje aplikacije u kodu. Naravno, web-aplikacija nije potpuno ista kao u Figma, ali kad ih usporedim, vidim neke sličnosti u izgledu.

3.2.1. LogIn/Prijava



The image shows a login form prototype for 'TravelHelper'. At the top, the logo 'TravelHelper' is displayed in a brown font, with the tagline 'Experience more with TravelHelper.' underneath. A horizontal line separates the header from the main content. The title 'Log In' is centered above the form. The form consists of two input fields: 'Tvoja e-mail adresa' and 'Lozinka'. Below these fields are two links: 'Remember me' (with an unchecked checkbox) and 'Forgot password'. A large orange button labeled 'Prijavi se!' is positioned below the links. A dashed line with the word 'or' in the center separates the login section from the Google sign-in section. The Google sign-in section features the Google 'G' logo. At the bottom, there is a link 'Nemaš korisnički račun? Registriraj se!'.

Slika 2: Prototip prijave

Prijava

Upišite svoj e-mail i lozinku!

Email

Lozinka

[Zaboravio si šifru?](#)

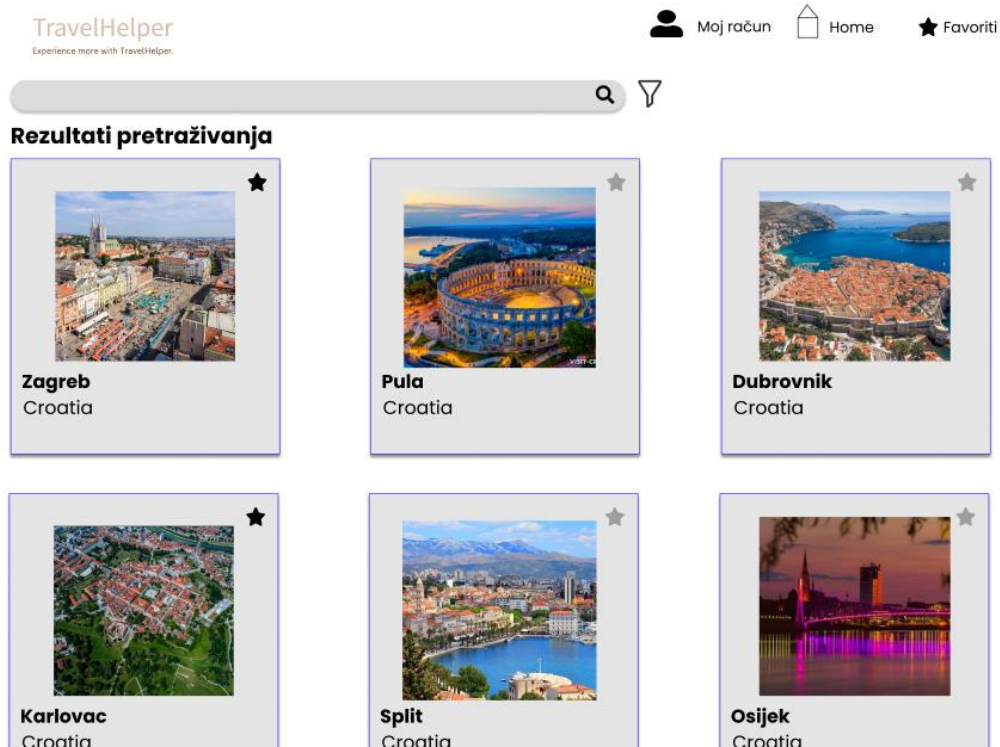
Submit

Nemaš račun? [Registriraj se](#)

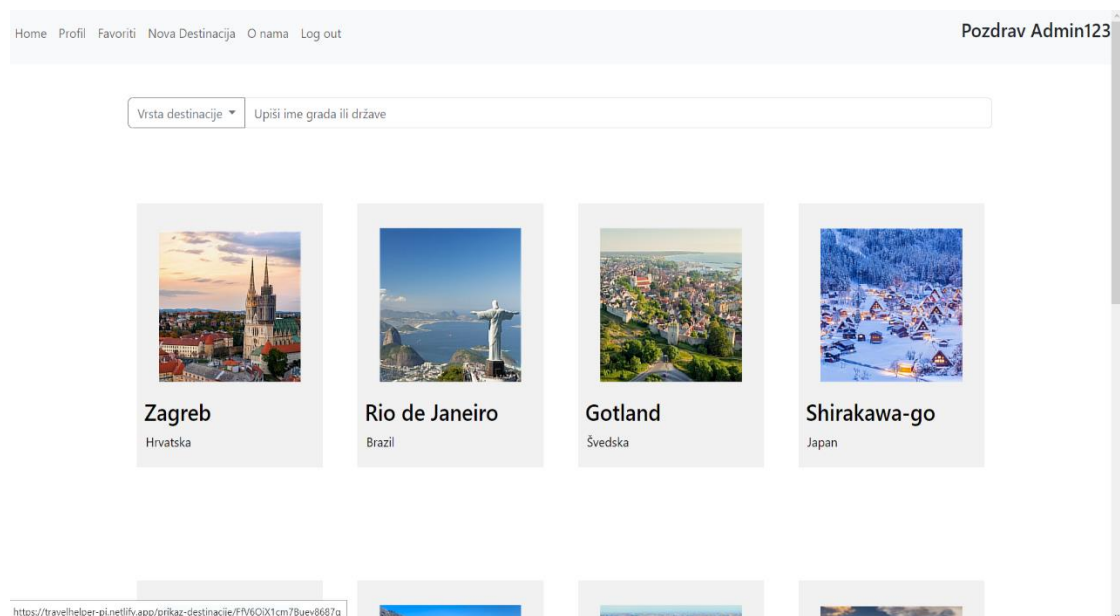
Slika 3: Prijava na web-aplikaciji

Dizajn je znatno drugačiji na prototipu i na finalnoj, ali neki gumbovi kao zaboravljena lozinka i registriraj se su isti, jedino ima samo jedan način prijavljivanja, a to je kroz email i lozinku, dok na prototipu ima opcija još da se može prijaviti i preko Google Accounta. Isto logo aplikacije se samo nalazi na About us stranici, pa mi je bilo malo pretjerano stavljati ga na svaku stranicu kako je i prikazano na prototipu.

3.2.2. Home



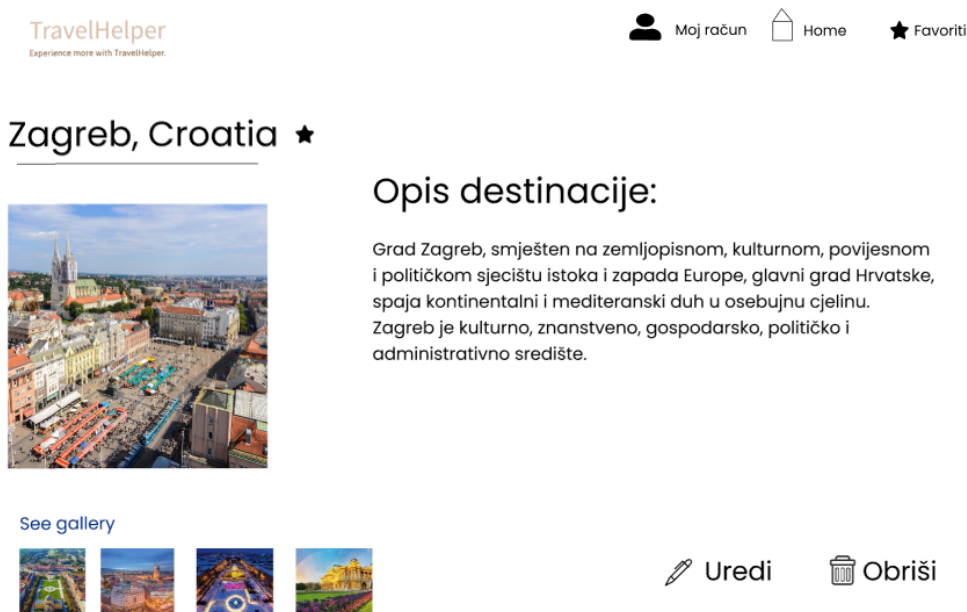
Slika 4: Prototip početne stranice



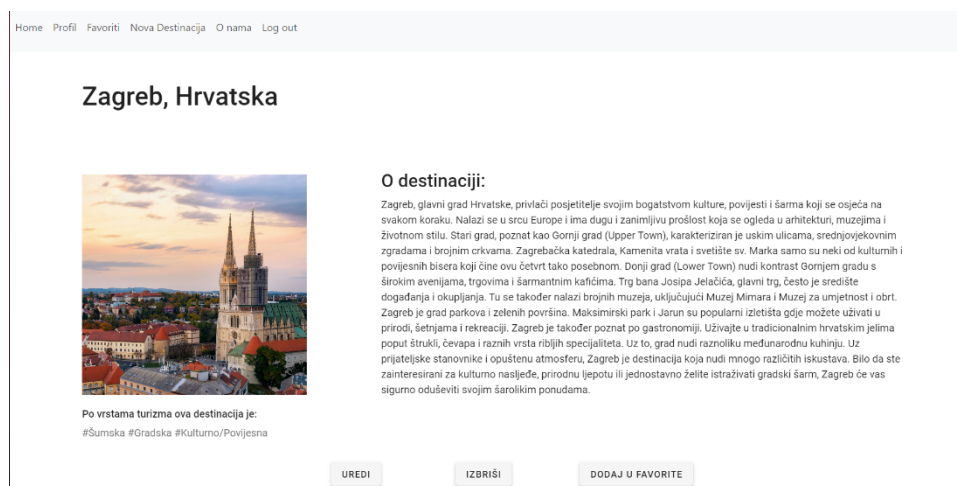
Slika 5: Početna stranica

Početna stranica ili prikaz svih destinacija su dosta slični, jedino nav-bar je dosta drugačiji i to što na svakoj kartici ima gumb za dodavanje u favorite, dok na završnoj aplikaciji se može dodat u favorite jedino kada se stisne na pregled destinacije. Filter na prototipu je zapravo kao što je odabir vrste destinacije na završnoj aplikaciji, i odmah pored toga je pretraga destinacija.

3.2.3. Prikaz destinacije – Administrator



Slika 6: Prototip pregleda destinacije(Administrator)



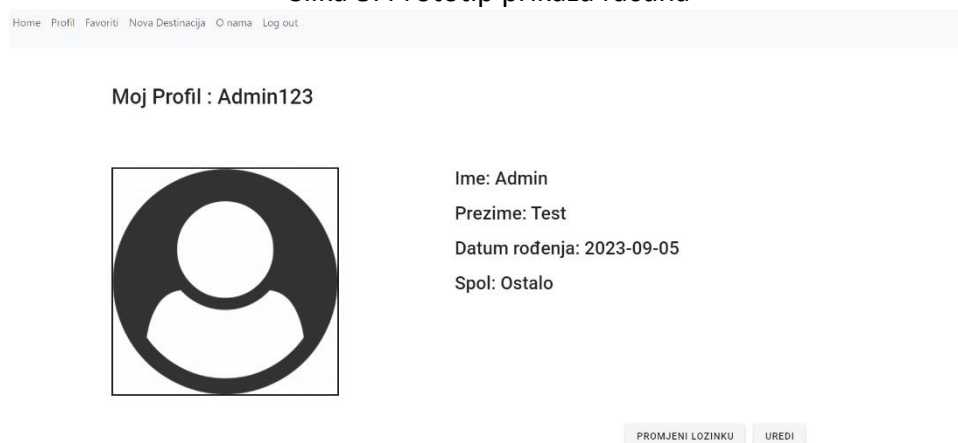
Slika 7: Pregled destinacije (Administrator)

Dizajn je dosta sličan samo se na prototipu vidi moja propala ideja za galerijom slika za svaku destinaciju. Htjela sam da aplikacija cijela bude dosta minimalistička tako da mi se ta galerija jednostavno nigdje nije uklapala na prikazu destinacija. Za dodavanje u favorite je gumb odmah pored uredi i izbriši, dok je na prototipu isto pored imena točno.

3.2.4. Prikaz profila



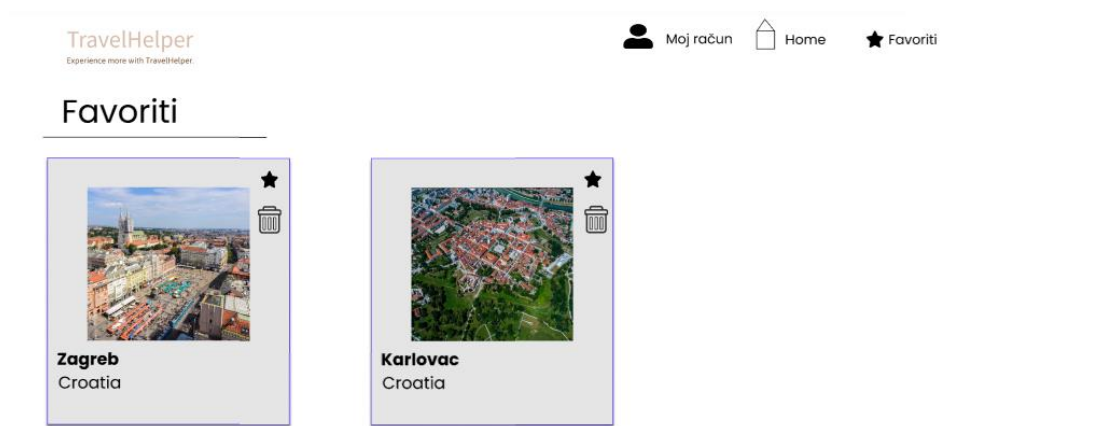
Slika 8: Prototip prikaza računa



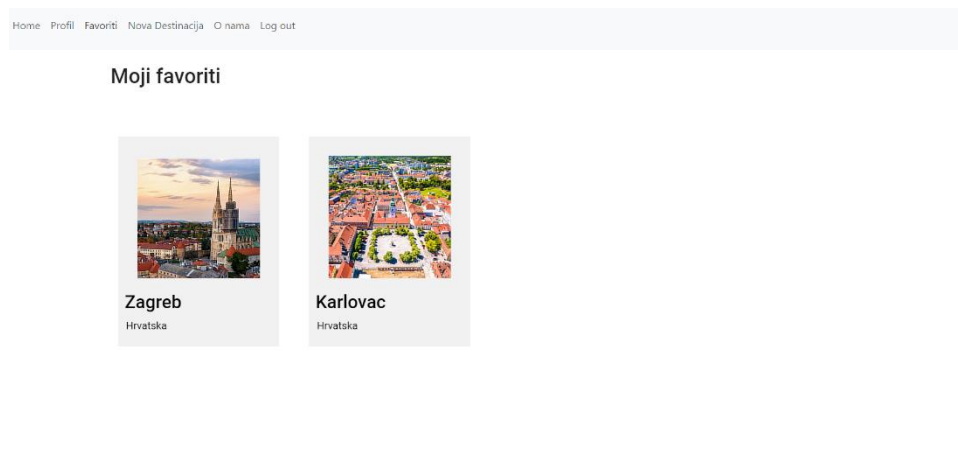
Slika 9: Prikaz profila

Prikaz profila je najbliži od svih stranica Figmi. Jedino dodano je još gumb za promjenu lozinke jer mi to spada pod 'upravljanje i uređivanje profila' pa mi je bilo obavezno da to još stavim.

3.2.5. Pregled favorita



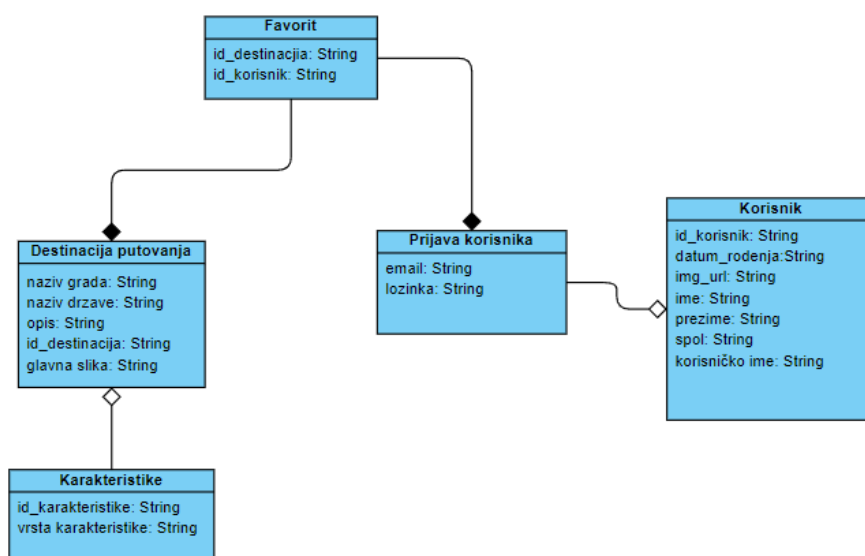
Slika 10: Prototip za pregled favorita



Slika 11: Prikaz favorita korisnika

Pregled favorita je ista dosta sličan završnoj aplikaciji, jedino da se izbriše ili doda destinacija iz favorita i u favorite, mora se ući u pregled destinacije i to obaviti tamo.

3.3. Class diagram



Slika 12: Class dijagram

Ovaj class dijagram pomaže u razumjevanju strukture aplikacije Travel Helper i odnosa između njenih ključnih komponenti. Ove klase zajedno čine osnovu za funkcionalnosti aplikacije, omogućavajući korisnicima da pretražuju destinacije, čuvaju omiljene destinacije i ostvaruju prijavu u aplikaciju. Class dijagram ima 5 klasa. Prva klasa je korisnik, on u sebi sadrži attribute : id, datum rođenja, sliku profila, ime korisnika, prezime korisnika, spol, i korisničko ime. Druga klasa je prijava korisnika, on služi za prijavu i u sebi ima attribute email korisnika i lozinku korisnika. On se veže sa korisnikom na način da ako korisnik izbriše podatke o sebi, još uvijek ostaje račun korisnika spremljen. Treća klasa je favoriti, tamo se spremaju svi favoriti koje prijavljeni korisnik odabere, njegovi atributi su id destinacije i id korisnika. On ako se obriše račun korisnika nestaje jer jedino kroz šta je vezan je kroz korisnika koji sprema te destinacije u svoju bazu, isto ako se obriše spremljena destinacija, spremljeni favorit se također briše. Četvrta klasa je destinacija, tamo se spremaju sve destinacije koje korisnik može vidjeti i spremiti, njezini atributi su: naziv grada, naziv države, opis destinacije, id destinacije, i glavna slika destinacije. I zadnja klasa je karakteristike destinacije, njezini atributi su id karakteristike i naziv karakteristike. Svaka destinacija ima svoje karakteristike i ako se destinacija izbriše, to ne znaci da će se i postojeća karakteristika izbrisati.

4. Implementacija

Projekt se radio preko Visual Studio Code-a. U projektu sam koristila JavaScript, Vue, Html, Css, Firebase, Vuetify, Bootstrap, Croppa i sam hosting preko Netlify-a. Firebase sam koristila za spremanje podataka o destinacijama i korisnicima, koristila sam ga za prijavu i

autentifikaciju i promjenu lozinke, i koristila sam ga isto i za pohranu slika.

4.1. Prijava i registracija korisnika – funkcionalnost

Za prijavu korisnika služila sam se danim funkcijama u firebase-u, ali kada se korisnik registrirao sam još i upisane podatke spremala pod 'korisnici' u bazi podataka.

```
methods: {  
  | signup() {  
    const auth = getAuth();  
    if (this.pass !== this.ponovljenipass) {  
      alert("Lozinke nisu iste, pokušaj ponovno.");  
    } else if (this.pass.length < 6) {  
      alert("Lozinka mora biti duža!");  
    } else {  
      createUserWithEmailAndPassword(auth, this.email, this.pass)  
        .then(async (userCredential) => {  
          // Signed in  
          const user = userCredential.user;  
          const ime = this.ime;  
          const prezime = this.prezime;  
          const datum_rodenja = this.datum_rodenja;  
          const username = this.username;  
          const spol = this.spol;  
          const email = this.email;  
          if (  
            ime !== "" &&  
            prezime !== "" &&  
            datum_rodenja !== "" &&  
            username !== "" &&  
            spol !== ""  
          ) {  
            await setDoc(doc(db, "korisnici", email), {  
              Ime: ime,  
              Prezime: prezime,  
              DatumRodenja: datum_rodenja,  
              Username: username,  
              Spol: spol,  
              Email: email,  
              Favoriti: "",  
            });  
            console.log("Uspjesno dodano ");  
            alert("Uspješna registracija");  
          }  
        })  
    }  
  }  
}
```

Slika 13: Prikaz koda za sign-up (SignUp.vue)

Prvo sam davala uvjete pod kojima se može izvršiti registracija, i nakon što su ti uvjeti bili zadovoljeni sam koristila firebase createUserWithEmailAndPassword funkciju za dodavanje korisnika na Authentication i nakon toga sam koristila funkciju setDoc da svi upisani podaci se spremaju u bazu podataka, da se lakše mogu dohvatiti.

```

methods: {
  login() {
    signInWithEmailAndPassword(auth, this.email, this.pass)
      .then(() => {
        alert("Uspješna prijava! :");
        this.$router.replace("/home"); // u pushu mzes ici nazad
      })
      .catch((error) => {
        alert(error.message);
      });
  },
  sendPassReset() {
    if (this.email == "") {
      alert("Upiši email adresu prvo!");
    } else {
      const auth = getAuth();
      sendPasswordResetEmail(auth, this.email)
        .then(() => {
          this.$router.replace("/pass-reset-msg");
        })
        .catch((error) => {
          const errorCode = error.code;
          const errorMessage = error.message;
          // ..
        });
    }
  },
},
},

```

Slika 14: Prijava korisnika(LogIn.vue)

Za login/prijavu sam koristila isto funkciju `signInWithEmailAndPassword` iz firebase i uz nju sam imala `sendPassReset` funkciju koja šalje na mail link za promjenu lozinke ukoliko ju je korisnik zaboravio.

4.2. Prikaz i filter svih destinacija- funkcionalnost

```

methods: {
  async dohvatiSveDestinacije() {
    this.kartice = [];
    const querySnapshot = await getDocs(collection(db, "destinacije"));
    querySnapshot.forEach((doc) => {
      const data = doc.data();

      this.kartice.push({
        id: doc.id,
        glavnaSlika: data.ImageUrl,
        drzava: data.Drzava,
        grad: data.Grad,
        vrstadestinacije: data.VrstaDestinacije,
      });
    });
  },
},

```

Slika 15: Prikaz svih destinacija(Home.vue)

Sa funkcijom `getDocs` sam dohvaćala sve destinacije iz baze podataka i spremala ih u array `kartice`.

```

filtriraneDestinacije() {
  let kartice2 = [];
  for (let destinacija of this.kartice) {
    let counterTRUE = 0;
    let counterFALSE = 0;
    for (let element of this.vrstedestinacije) {
      if (destinacija.vrstedestinacije.includes(element) == false) {
        counterFALSE = counterFALSE + 1;
      } else {
        counterTRUE = counterTRUE + 1;
      }
    }
    if (counterTRUE == this.vrstedestinacije.length) {
      kartice2.push(destinacija);
    }
  }
  let termin = this.store.searchTerm.toLocaleLowerCase();
  let noveKartice = [];

  for (let kartica of kartice2) {
    if (
      kartica.grad.toLocaleLowerCase().indexOf(termin) >= 0 ||
      kartica.drzava.toLocaleLowerCase().indexOf(termin) >= 0
    ) {
      // sta radi index of? on u arrayu kartice trazi koji je index od t
      noveKartice.push(kartica);
    }
  }
  return noveKartice;
},

```

Slika 16: Filter za destinacije(Home.vue)

U funkciji `filtriraneDestinacije` sam napravila 2 filtera : filter po vrsti destinacije i filter po pretrazi. Za filter po vrsti destinacije sam napravila novi array `kartice2` u kojeg sam spremala

destinaciju po destinaciju ako je zadovoljio uvjet da se u destinaciji nalaze svi elementi iz arraya vrstedestinacije. A Za filter po pretrazi sam napravila ponovno novi array noveKartice u kojem su one završne destinacije koje se nakon svih filtera prikazuju. Uspoređivala sam ga sa varijablom termin u koji se spremala vrijednost koju je korisnik pretraživao. I na kraju ako zadovoljava uvjet se u noveKartice sprema destinacija iz kartice2.

4.3. Pregled pojedinog putovanja – funkcionalnost

```
methods: {
  async dovatiDestinaciju() {
    const idDestinacije = this.$route.params.id;
    console.log(idDestinacije);
    this.prikazDestinacije = {};
    const querySnapshot = await getDocs(collection(db, "destinacije"));
    querySnapshot.forEach((doc) => {
      const data = doc.data();
      if (doc.id == idDestinacije) {
        this.prikazDestinacije = {
          id: doc.id,
          glavnaSlika: data.ImageUrl,
          drzava: data.Drzava,
          grad: data.Grad,
          opis_destinacije: data.OpisDestinacije,
        };
        this.vrstadestinacije.push(data.VrstaDestinacije);
      } else {
        return;
      }
    });
  }
};
```

Slika 17: Prikaz odabrane/pojedine destinacije (PrikazDestinacije.vue)

Za pregled pojedinog putovanja sam morala spremati id destinacije u rutu za prikaz pojedine destinacije. I nakon tog sam koristila funkciju getDocs da spremim sve podatke o toj destinaciji ali samo uz uvijet da se id rute podudara sa id destinacije u bazi.

4.4. Uređivanje i dodavanje destinacije – funkcionalnost (Administrator)

Prijavom korisničkog računa Admin@Test.com (koji se identificira kao administrator), možemo uređivati destinacije.


```

methods: {
  async dovatiDestinaciju() {
    const idDestinacije = this.$route.params.id;
    console.log(idDestinacije);
    this.prikazDestinacije = {};
    const querySnapshot = await getDocs(collection(db, "destinacije"));
    querySnapshot.forEach((doc) => {
      const data = doc.data();
      if (doc.id == idDestinacije) {
        this.ime_drzave = data.Drzava;
        this.ime_Grada = data.Grad;
        this.opis_destinacije = data.OpisDestinacije;

        this.odabrane_vrste = data.VrstaDestinacije;
      } else {
        return;
      }
    });
  }
};

```

Slika 18: Dohvaćanje podataka o destinaciji koju uređujemo (UrediDestinaciju.vue)

Prvo pomoću `getDocs` spremamo podatke za željeni id, tako da je administratoru lakše urediti podatke.

```

async UrediPodatke() {
  const idDestinacije = this.$route.params.id;
  this.pravaSlika.generateBlob((blobData) => {
    let Blobby = null;
    Blobby = blobData; // size blob ide u Blobby

    let imageName = (
      "destinacije/" +
      this.ime_Grada +
      this.ime_drzave +
      "/" +
      "GLAVNA.png"
    ).replace(/s/g, ""); // generiranje imena za svaku sliku
    console.log(imageName);

    const storageRef = ref(storage, imageName);
    uploadBytes(storageRef, Blobby).then((snapshot) => {
      getDownloadURL(storageRef).then(async (url) => {
        console.log("Javni link", url);
        const nazivGrada = this.ime_Grada;
        const nazivDrzave = this.ime_drzave;
        const opis_Destinacije = this.opis_destinacije;
        const vrstaDestinacija = this.odabrane_vrste;
        const imgUrl = url;
        console.log(imgUrl);
        if (
          nazivGrada != "" &&
          nazivDrzave != "" &&
          opis_Destinacije != "" &&
          vrstaDestinacija.length > 0 &&
          Blobby != null
        ) {
          if (opis_Destinacije.length > 1650) {
            alert(
              "Maksimalan broj znakova za opis destinacije je 1650 znakova!"
            );
          } else {

```

Slika 19: Prvi dio funkcije `urediPodatke` (UrediDestinaciju.vue)

```

    ) {
      if (opis_Destinacije.length > 1650) {
        alert(
          "Maksimalan broj znakova za opis destinacije je 1650 znakova!"
        );
      } else {
        await setDoc(doc(db, "destinacije", idDestinacije), {
          imageUrl: imgUrl,
          Grad: nazivGrada,
          Drzava: nazivDrzave,
          OpisDestinacije: opis_Destinacije,
          VrstaDestinacije: vrstaDestinacija,
        });
        console.log("Uspješno uređivanje ");
        this.$router.replace("/prikaz-destinacije/" + idDestinacije);
      }
    } else {
      alert("Ispuni sva polja!!");
    }
  });
};

```

Slika 20: Drugi dio funkcije urediPodatke (UrediDestinaciju.vue)

Prvo sam generirala sliku destinacije i spremala je u firebase storage,, nakon toga sam dobila url slike i ako su svi podaci uspješno uneseni i uvjeti zadovoljeni pomoću setDocs se ažurira destinacija i sprema se nova verzija destinacije.

4.5 Micanje i dodavanje destinacija u favorite – funkcionalnost

Spremanje destinacija u favorite se izvršava tako da korisnik ode na pregled pojedine destinacije i tamo ispod opisa i svega o destinacije ima gumb za dodavanje ili micanje destinacije iz favorita.

```

async DodajUFavorite() {
  const idDestinacije = this.$route.params.id;
  const KorisniciRef = doc(db, "korisnici", store.currentUser);

  await updateDoc(KorisniciRef, {
    Favoriti: arrayUnion(idDestinacije),
  });
  window.location.reload();
},
async IzbrisiIzFavorita() {
  const idDestinacije = this.$route.params.id;
  const KorisniciRef = doc(db, "korisnici", store.currentUser);

  await updateDoc(KorisniciRef, {
    Favoriti: arrayRemove(idDestinacije),
  });
  window.location.reload();
},
StoreIncludes() {
  if (
    store.prikazFavorita != undefined &&
    store.prikazFavorita.includes(this.prikazDestinacije.id)
  ) {
    return true;
  } else return false;
},
StoreDoesntIncludes() {
  if (
    store.prikazFavorita == undefined ||
    !store.prikazFavorita.includes(this.prikazDestinacije.id)
  ) {
    return true;
  } else return false;
},

```

Slika 21: Dodavanje i brisanje favorita (PrikazDestinacije.vue)

Ako destinacija nije u favoritima pomoću funkcije StoreDoesntIncludes, prikazujemo gumb 'Dodaj u favorite', a ako je onda pomoću funkcije StoreIncludes onda prikazujemo samo gumb 'Izbriši iz favorita'.

Za dodavanje sam koristila funkciju updateDoc gdje sam koristila arrayUnion da se doda id destinacije u array Favoriti i kada se to izvrši se refresha cijeli zaslon i promjeni se gumb 'Dodaj u Favorite' u 'Izbriši iz favorita'.

A za micanje iz favorita sam koristila isto funkciju updateDoc, ali sa arrayRemove, gdje briše id destinacije koji je već pohranjen u arrayu Favoriti.

4.6. Uređivanje profila – funkcionalnost

```

methods: {
  async DohvatiPodatkeKorisnika() {
    const emailKorisnika = this.$route.params.id;
    const docRef = doc(db, "korisnici", emailKorisnika);
    const docSnap = await getDoc(docRef);

    if (docSnap.exists()) {
      const data = docSnap.data();
      this.ime = data.Ime;
      this.prezime = data.Prezime;
      this.datum_rodenja = data.DatumRodenja;
      this.email = data.Email;
      this.username = data.Username;
      this.spol = data.Spol;
      console.log("Document data:", docSnap.data());
    } else {
      // docSnap.data() will be undefined in this case
      console.log("No such document!");
    }
  },

```

Slika 22: Spremanje podataka korisnika iz baze u varijable (UrediProfil.vue)

Prvo sam dohvatila sve korisničke podatke sa getDoc funkcijom da korisnik lakše uređuje već postojeće podatke.

```
async UrediPodatke() {
  const idKorisnika = this.$route.params.id;
  this.slikakorisnika.generateBlob((blobData) => {
    let Blobby = null;
    Blobby = blobData; // size blob ide u Blobby

    let imageName = (
      "korisnici/" +
      this.ime +
      this.prezime +
      "/" +
      "GLAVNA.png"
    ).replace(/\\/s/g, ""); // generiranje imena za svaku sliku
    console.log(imageName);

    const storageRef = ref(storage, imageName);
    uploadBytes(storageRef, Blobby).then((snapshot) => {
      getDownloadURL(storageRef).then(async (url) => {
        console.log("Javni link", url);
        const imeKorisnika = this.ime;
        const prezimeKorisnika = this.prezime;
        const usernameKorisnika = this.username;
        const DatumRodenjaKorisnika = this.datum_rodenja;
        const spolKorisnika = this.spol;
        const imgUrl = url;
        console.log(imgUrl);
        if (
          imeKorisnika !== "" &&
          prezimeKorisnika !== "" &&
          usernameKorisnika !== "" &&
          DatumRodenjaKorisnika !== "" &&
          spolKorisnika !== ""
        ) {
          await setDoc(doc(db, "korisnici", idKorisnika), {
            ImageUrl: imgUrl,
            Ime: imeKorisnika,
            Prezime: prezimeKorisnika,
```

Slika 23: Prvi dio spremanja korisničkih podataka (UrediProfil.vue)

```
await setDoc(doc(db, "korisnici", idKorisnika), {
  ImageUrl: imgUrl,
  Ime: imeKorisnika,
  Prezime: prezimeKorisnika,
  Spol: spolKorisnika,
  DatumRodenja: DatumRodenjaKorisnika,
  SlikaProfila: imgUrl,
  Username: usernameKorisnika,
}));
console.log("Uspješno uređivanje ");
this.$router.replace("/profil-korisnika/" + idKorisnika);
} else {
  alert("Ispuni sva polja!!");
}
});
```

Slika 24: Drugi dio spremanja korisničkih podataka (UrediProfil.vue)

Prvo sam stvorila profilnu sliku i pohranila je u Firebase Storage. Nakon toga, dobila sam URL slike. Ako su svi podaci ispravno uneseni i ispunjeni svi uvjeti, koristim setDocs za ažuriranje korisničkih podataka i nakon spremanja svih podataka te odvede ponovno u ažurirani profil.

4.7. Prikaz favorita - funkcionalnost

```
async dohvatiSveDestinacije() {
  this.kartice = [];
  if (store.prikazFavorita !== undefined) {
    const querySnapshot = await getDocs(collection(db, "destinacije"));
    querySnapshot.forEach((doc) => {
      const data = doc.data();
      if (store.prikazFavorita.includes(doc.id)) {
        this.kartice.push({
          id: doc.id,
          glavnaSlika: data.ImageUrl,
          drzava: data.Drzava,
          grad: data.Grad,
          vrstadestinacije: data.VrstaDestinacije,
        });
      }
    });
  }
}
```

Slika 25: Prikaz svih destinacija (PrikazFavorita.vue)

U App.vue odmah čim se korisnik prijavi spremaju se svi korisnički podaci u posebne varijable da bi se mogle koristiti bilogdje u aplikaciji. Ova funkcija radi samo ako u store.prikazFavorita su spremljeni id destinacija koje su u favoritima u korisniku. Izvlači podatke destinacija i sprema ih u array kartice. Iz tog arraya se pomoću komponente 'Destinacije.vue' prikazuje svaka destinacija spremljena u favoritima.

4.8. Destinacija.vue – komponenta

```
<template>
  <div class="city-wrapper">
    <div class="container" style="margin-top: 55px">
      <router-link
        :to="'/prikaz-destinacije/' + info.id"
        style="color: black; text-decoration: none"
      >
        <div class="city-container">
          
          <h3 style="margin-top: 20px; margin-left: -20px">{{ info.grad }}</h3>
          <p style="margin-left: -18px; margin-bottom: -10px">
            {{ info.drzava }}
          </p>
        </div>
      </router-link>
    </div>
  </div>
</template>

<script>
export default {
  props: ["info"],
  name: "Destinacija",
};
</script>

<style>
.city-container {
  display: inline-block;
  background-color: #f1f1f1;
  padding: 30px;
}
```

Slika 26: Prikaz svake destinacije (Destinacija.vue)

Pomoću ove komponente se generiraju sve kartice destinacija koje koristimo u PrikazFavorita.vue i Home.vue. U svakoj od tih komponenta prvo u <style> moramo importati Destinacija from Destinacija.vue, staviti pod components i staviti ime Destinacija i pomoću v-for petlje unutar <Destinacija> elementa se generiraju i prikazuju sve destinacije.