UNAPRJEĐENJE POSTOJEĆEG SUSTAVA AŽURNOSTI ZA UDRUGU POMOĆU PRILAGOĐENE WEB APLIKACIJE

Mikulec, Iva

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Algebra University College / Visoko učilište Algebra**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:225:503767

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-12-01**

ALGEBRA SVEUČILIŠTE

Repository / Repozitorij:

Algebra Univerity - Repository of Algebra Univerity

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

UNAPRJEĐENJE POSTOJEĆEG SUSTAVA AŽURNOSTI ZA UDRUGU POMOĆU PRILAGOĐENE WEB APLIKACIJE

Iva Mikulec

Zagreb, veljača 2023.

Student vlastoručno potpisuje Završni rad na prvoj stranici ispred Predgovora s datumom i oznakom mjesta završetka rada te naznakom:

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada."

U Zagrebu, 16. veljače 2023.

Predgovor

Zahvaljujem svomu mentoru mr. sc. Mariu Periću na podršci, stručnoj pomoći, strpljivosti te korisnim savjetima i sugestijama tijekom izrade ovoga završnoga rada. Također, želim zahvaliti svojim kolegama na prekrasnim zajedničkim trenutcima koje sam proživjela na Visokome učilištu Algebra.

Zahvaljujem magistrici hrvatskog jezika i književnosti, Maji Stokić, na lekturi ovoga završnoga rada.

Najviše zahvaljujem svojim roditeljima i sestri Petri, koji su me bodrili ove tri godine studiranja i koji su uvijek vjerovali u mene.

Nakraju želim zahvaliti svim profesorima i asistentima Visokoga učilišta Algebra na znanju koje sam primila i svim savjetima kojima su me izgradili u osobu koja sam danas.

A Gradišćanska 24, HR-10000 Zagreb
T (01) 2222 182
E student@algebra.hr
www.algebra.hr

ALGEBRA

Temeljem članka 8. Pravilnika o završnom radu i završnom ispitu na preddiplomskom studiju Visokog učilišta Algebra sačinjena je ova

Potvrda o dodjeli završnog rada

kojom se potvrđuje da studentica Iva Mikulec, JMBAG 0058210920, OIB 83136037178 u šk. godini 2021./2022., studij: Multimedijsko računarstvo - Preddiplomski studij, od strane povjerenstva za provedbu završnog ispita, dana 23.02.2022. godine, ima odobrenu izradu završnog rada

s temom: **Unaprjeđenje postojećeg sustava ažurnosti za udrugu pomoću prilagođene web aplikacije**

i sažetkom rada: Ovaj završni rad sadrži detaljan opis razvoja korisničkoga sučelja web aplikacije koja je izrađena za potrebe korisnika. Korisnik je web aplikacije udruga RODA (RODitelji u Akciji). Portal udruge RODA jedino je mjesto na kojem trudnice u Hrvatskoj mogu dobiti ažurne podatke potrebne za planiranje poroda. Ovom web aplikacijom zdravstvenim djelatnicima omogućava se unos podataka i fotografija svojih rodilišta. Za izradu korisničkoga sučelja korištene su sljedeće moderne web tehnologije: biblioteka React te Material UI knjižnica React komponenti. Tijekom procesa izrade učvrstila se suradnja udruge i rodilišta, a time su podatci o njihovim uslugama kvalitetniji i dostupniji budućim roditeljima.

Mentor je: Mario Perić.

Odobrenjem završnog rada studentici je omogućen upis kolegija "Izrada završnog projekta/Praksa" te je sukladno članku 8. Pravilnika o završnom radu i završnom ispitu dužan najkasnije do početka nastave ljetnog semestra u sljedećoj školskoj godini, uspješno obraniti završni rad uspješnim polaganjem završnog ispita.

U protivnom studentica može zatražiti novog mentora/icu i temu te ponovo upisati kolegij "Izrada završnog projekta/Praksa" budući da rad koji nije predan i obranjen na završnom ispitu u roku određenom Pravilnikom završnom radu i završnom ispitu prestaje vrijediti. Izrada novog završnog rada se izvodi sukladno rokovima određenima za školsku godinu u kojoj je studentici određen novi mentor/ica i dodijeljen novi završni rad.

Potpis studentice: Potpis mentora: Potpis predsjednika povjerenstva:

Ova potvrda izdaje se u 4 (četiri) primjerka od kojih 3 (tri) idu kao prilog završnom radu.

Sažetak

Ovaj završni rad sadrži detaljan opis razvoja korisničkoga sučelja *web* aplikacije koja je izrađena za potrebe korisnika. Korisnik je *web* aplikacije udruga RODA (RODitelji u Akciji). Portal udruge RODA jedino je mjesto na kojem trudnice u Hrvatskoj mogu dobiti ažurne podatke potrebne za planiranje poroda. Ovom *web* aplikacijom zdravstvenim djelatnicima omogućava se unos podataka i fotografija svojih rodilišta. Za izradu korisničkoga sučelja korištene su sljedeće moderne *web* tehnologije: biblioteka React te Material UI knjižnica React komponenti. Tijekom procesa izrade učvrstila se suradnja udruge i rodilišta, a time su podatci o njihovim uslugama kvalitetniji i dostupniji budućim roditeljima.

Ključne riječi: *web* aplikacija, podatci, React biblioteka, udruga RODA, korisničko sučelje, Material UI

Abstract

This paper contains detailed description of front-end development of the web application created for the RODA association (Parents in Action). Portal of the RODA association is the only place where pregnant women in Croatia can get up-to-date information needed for birth planning. This application allows healthcare professionals to enter data and photos of their maternity hospitals. The following modern web technologies were used for its creation: React library and Material-UI component library. During the development process, the cooperation between association and the maternity hospital was strengthened and data of their services are more accessible to future parents.

Keywords: web application, data, React library, RODA association, front-end development, Material UI

Sadržaj

1. Uvod 1

2. Vizija i strategija 2

2.1. Vizija 2

2.2. Ciljane skupine 3

2.3. Potrebe 6

2.4. Proizvod 7

2.5. Opis funkcionalnosti sustava 7

3. Dizajn *web* aplikacije 11

3.1. Dijagram *web* aplikacije 11

3.2. Žičani okvir 13

3.3. Dizajn-sistem 23

3.3.1. Boja 24

3.3.2. Tipografija 25

3.3.3. Raspored 26

3.3.4. Ikone i komponente 28

3.4. Konačan dizajn 30

4. Izrada *web* aplikacije 41

4.1. Odabrane tehnologije 41

4.2. Implementacija pojedinih tehnologija 43

4.2.1. Baza podataka 44

4.2.2. Biblioteka React 47

4.2.3. Typescript 49

4.2.4. Material UI 51

4.2.5. Yarn 54
4.2.6. Create React App 55
4.2.7. *useState* i *useEffect* 55
4.2.8. *useContext* 57
4.2.9. React Router 59
4.2.10. Formik 61
4.2.11. Yup 63
4.2.12. React Testing biblioteka 63
4.3. Korisničko testiranje 65
Zaključak 72
Popis kratica 73
Popis slika 74
Popis tablica 76
Popis kôdova 77
Literatura 78
Prilog 79

1. Uvod

Web aplikacija softversko je rješenje kojem se pristupa *web* preglednikom. Rastom uporabe interneta, raste i potreba za sofisticiranim i inovativnim *web* aplikacijama te se one koriste u svakodnevnom životu.

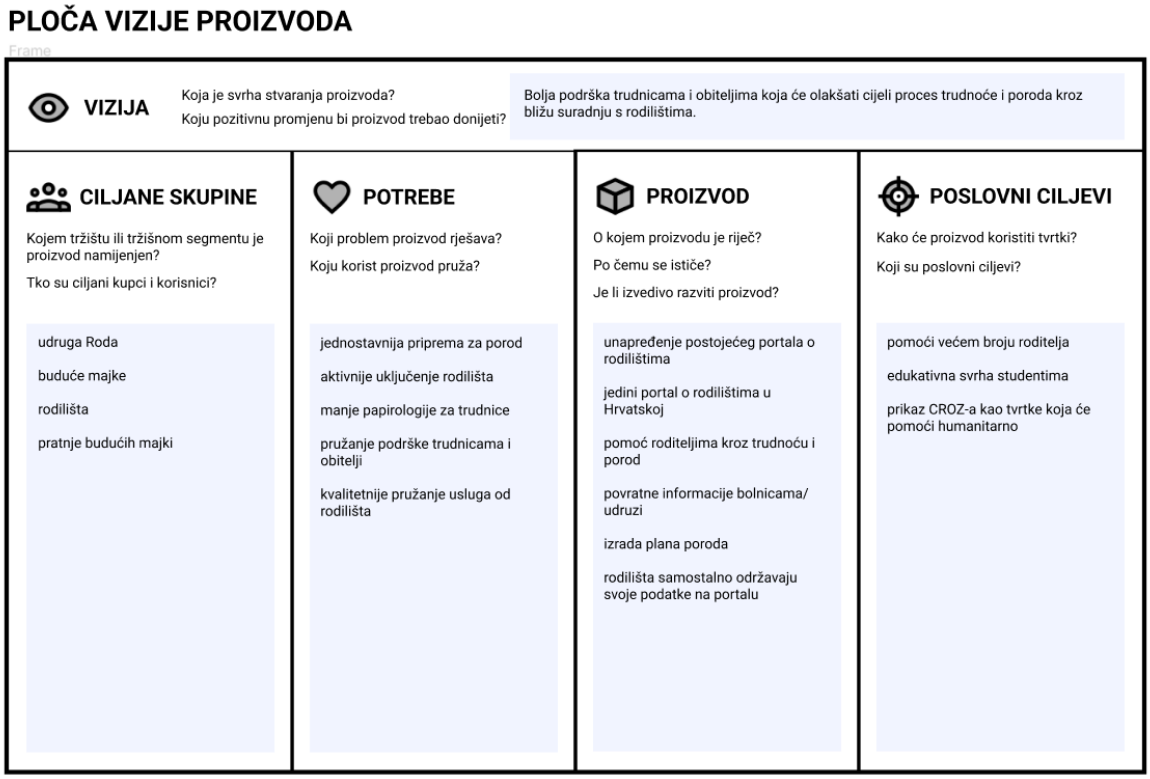
U ovome završnome radu objašnjen je razvoj korisničkoga sučelja *web* aplikacije (engl. *front-end development*) za udrugu RODA (RODitelji u Akciji). Udruga RODA bavi se pravima djece i roditelja u Hrvatskoj i trenutačno je jedina udruga koja na svome portalu ima informacije o svim rodilištima u Hrvatskoj.

Cilj je ovoga završnoga rada unaprjeđenje postojećega portala *web* aplikacijom koja bi proces prikupljanja i objavljivanja podataka o rodilištima učinila što bezbolnijim i za udrugu i rodilišta te kako bi podatci bili što ažurniji za krajnje korisnike, odnosno buduće roditelje.

Dizajn aplikacije napravljen je pomoću Figma alata dok se za njegovu implementaciju koristi moderna React biblioteka. React je trenutačno pri samome vrhu najkorištenijih tehnologija kada je u pitanju implementacija korisničkoga sučelja *web* aplikacije. Prvo poglavlje daje uvid u viziju i strategiju izrade, sljedeće opisuje proces dizajna, a sama implementacija opisana je u zadnjem poglavlju.

2. Vizija i strategija

Prilikom izrade ovakvoga tipa proizvoda najbolje je krenuti od vizije, a kasnije i strategije razvoja proizvoda. Nakon inicijalnoga sastanka s korisnikom sastavlja se tzv. ploča vizije proizvoda (engl. *The Product Vision Board*) koja služi definiranju ciljane skupine, korisničkih potreba, ključnih funkcionalnosti i poslovnih ciljeva. Ploča vizije proizvoda sastoji se od pet odjeljaka: vizije, ciljane skupine, njezinih potreba, strategije proizvoda i poslovnih ciljeva.



Slika 2.1 Ploča vizije proizvoda

2.1. Vizija

Vizijom se opisuje konačan cilj ili svrha proizvoda, odnosno pozitivna promjena koja se proizvodom želi postići. Bitno je da ju podupiru razvojni tim i interesne skupine (engl. *stakeholders*). Vizija treba biti jasna, kratka, jezgrovita, razumljiva i inspirativna. Vizija proizlazi iz prethodno spomenutih stavki, a u konkretnome slučaju ona glasi: „Bolja podrška

trudnicama i obiteljima koja će olakšati cijeli proces trudnoće i poroda bližom suradnjom s rodilištima.” S obzirom na ciljane skupine, razlikuju se dvije vizije:

a. vizija za trudnice
b. vizija za rodilišta.

Vizija za trudnice glasi: „rodilista.roda.hr je *web* aplikacija za trudnice koje žele da im porod prođe što jednostavnije, koja olakšava cijeli plan i proces poroda. Za razliku od trenutačne aplikacije ovaj će proizvod biti poboljšan i jedini postojeći u Hrvatskoj.” Vizija za rodilišta glasi: „rodilista.roda.hr je *web* aplikacija za rodilišta koja žele poboljšati svoje usluge, koja omogućava lakše predstavljanje i prikupljanje povratnih informacija. Za razliku od trenutačnoga procesa, ovaj proizvod bit će jedini u Hrvatskoj.”

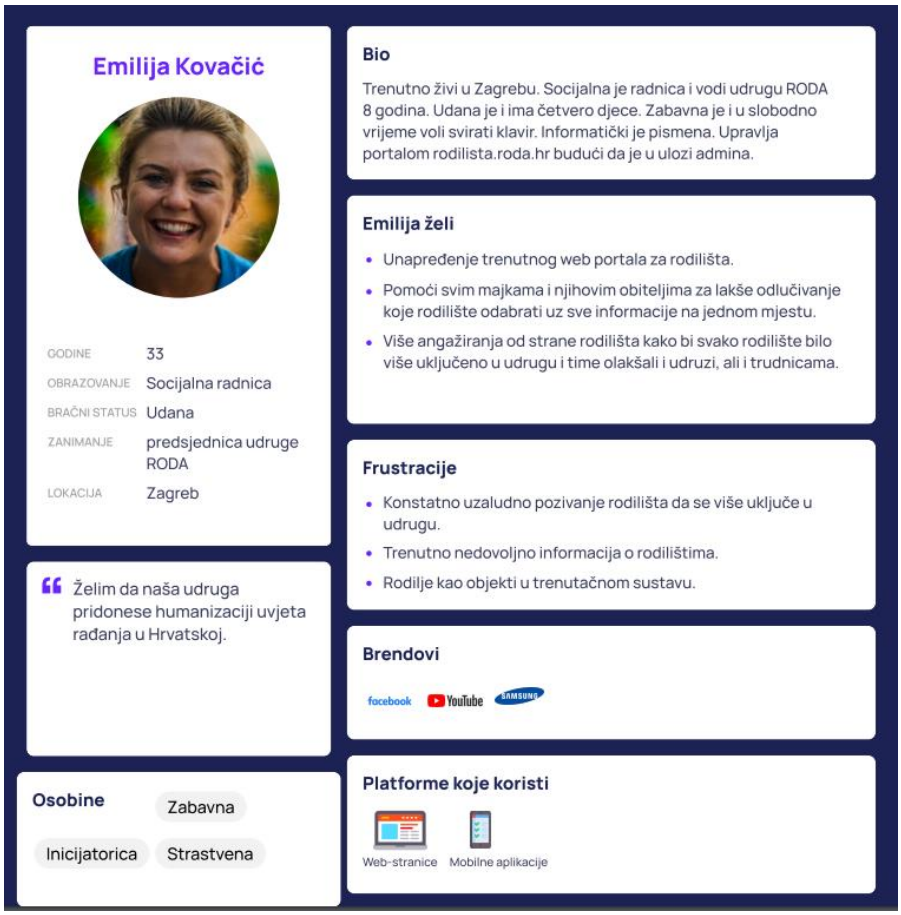
2.2. Ciljane skupine

Ciljana skupina predstavlja skup potencijalnih korisnika kojoj je namijenjen krajnji proizvod. Treba obratiti pozornost na to da ciljana skupina bude homogena i jasna. Nakon inicijalnoga sastanka s korisnikom i određivanja temeljnih funkcionalnosti opisanih u potpoglavlju 2.5., mogu se definirati četiri ciljane skupine:

1. udruga RODA
2. buduće majke
3. rodilišta
4. pratnje budućih majki.

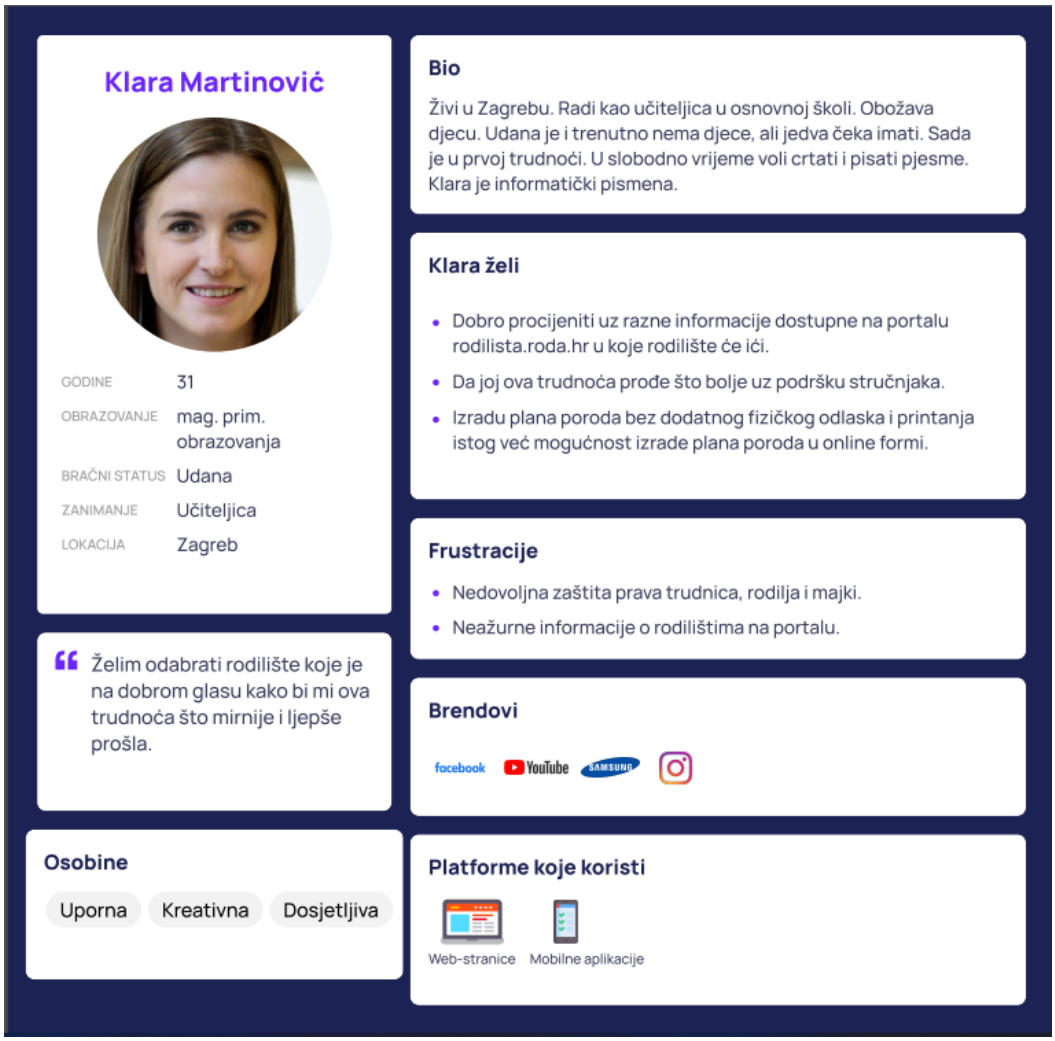
Za prve tri navedene ciljane skupine izrađene su persone. Ciljana skupina razlikuje se od persone u tome što predstavlja najširu sliku ciljane publike, dok je persona najuži prikaz ciljane publike. Persona je izmišljen lik kojim se predstavlja ciljana skupina. Proces stvaranja persona pomaže pri razumijevanju potreba, iskustava i ponašanja korisnika. Različiti ljudi imaju različite potrebe i očekivanja. Persone olakšavaju proces dizajna time što jasnije opisuju dobro korisničko iskustvo za ciljanu skupinu.

Persona iz ciljane skupine udruge RODA predsjednica je udruge RODA – Emilija Kovačić. Na slici se (Slika 2.2) nalazi prikaz čime se točno Emilija bavi, što želi, koje frustracije joj prolaze glavom, kakve ju osobine krase te kakve digitalne platforme najčešće koristi. Emilija želi unaprjeđenje trenutačnoga *web* portala za rodilišta i nezadovoljna je time što neprestano poziva rodilišta da se više uključe u udrugu.



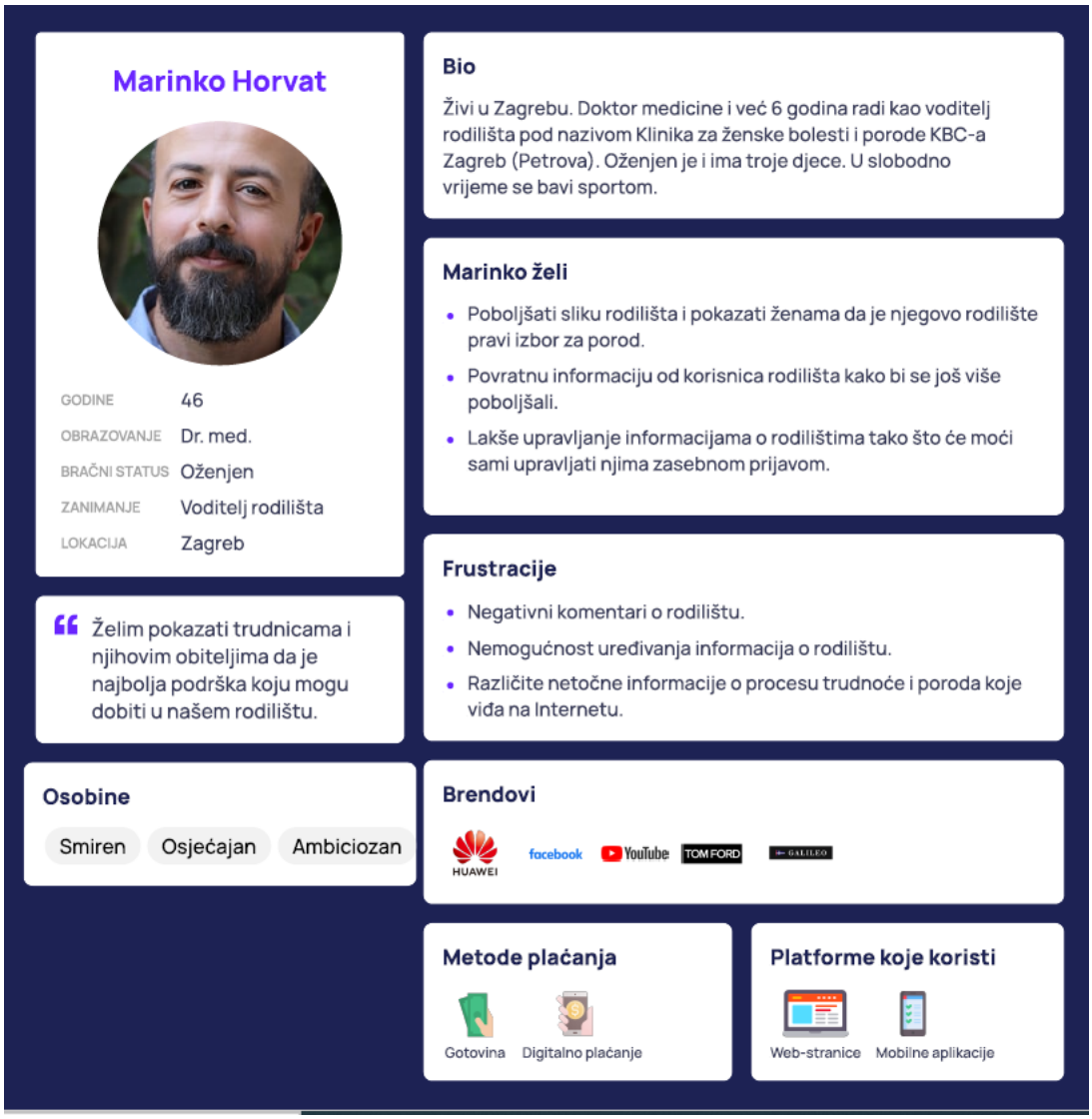
Slika 2.2 Persona – predsjednica Udruge RODA

Persona druge ciljane skupine je trudnica Klara. Iz predstavljene analize na slici (Slika 2.3) proizlazi kako je Klara frustrirana neažurnim informacijama o rodilištima na portalu i nedovoljnom zaštitom prava trudnica, rodilja i majki. Trenutačno je u prvoj trudnoći i želi da joj ova trudnoća prođe što ljepše i mirnije. Također, Klara bi htjela izraditi plan poroda bez dodatnoga fizičkoga odlaska i njegova ispisa, s mogućnošću njegove izrade u *online* formi. Usto, na temelju informacija dostupnih na portalu rodilista.roda.hr, htjela bi dobro procijeniti u koje će rodilište ići. Koristi se digitalnim plaćanjem, mobilnim i *web* aplikacijama što ju čini i više nego primjerenom korisnicom ovoga proizvoda.



Slika 2.3 Persona – trudnica / buduća majka

Sljedeća su ciljana skupina rodilišta te njih predstavlja voditelj rodilišta Marinko Horvat. Marinka frustrira nemogućnost uređivanja informacija o rodilištu kao i netočne informacije o procesu trudnoće i poroda koje vidi na internetu. Želio bi poboljšati sliku rodilišta i pokazati ženama da je njegovo rodilište pravi izbor za porod. Osim toga, htio bi lakše upravljati informacijama o rodilištu time što će ih moći sam unositi, a ne sve rješavati pozivima s administratorima udruge RODA.



Slika 2.4 Persona – voditelj rodilišta

2.3. Potrebe

U dijelu „Potrebe“ sadržana je vrijednost proizvoda. Ovdje je opisan glavni problem koji proizvod rješava, odnosno primarne vrijednosti koje će donijeti. Navedenim se nudi jasnoća korištenja samim proizvodom. Razgovorom s korisnikom i stvaranjem persona, ustanovljene su osnovne potrebe korisnika:

- jednostavnija priprema za porod
- aktivnije uključenje rodilišta
- manje papirologije za trudnice
- pružanje podrške trudnicama i obitelji

- kvalitetnije pružanje usluga rodilišta.

2.4. Proizvod

Dio o proizvodu sažima tri do pet ključnih funkcionalnosti proizvoda. Odabrane funkcionalnosti trebale bi korelirati s potrebama identificiranima u odjeljku „Potrebe“. Iz prethodno definiranih potreba proizlaze sljedeće ključne karakteristike proizvoda:

- unaprjeđenje postojećega portala o rodilištima
- jedini portal o rodilištima u Hrvatskoj
- pomoć roditeljima tijekom trudnoće i poroda
- povratne informacije rodilištima/udruzi
- izrada plana poroda
- rodilišta samostalno održavaju svoje podatke na portalu.

Ono što trenutačno postoji na portalu rodilista.roda.hr glavne su informacije o rodilištima. Problem je trenutačne aplikacije to što udruga RODA sama dodaje i objavljuje informacije o rodilištima u Hrvatskoj. Trenutačno ne postoji galerija slika ni za koje rodilište, a rodilišta nemaju mogućnost ažuriranja vlastitih podataka. Stoga ova *web* aplikacija omogućava bolnicama zasebnu prijavu u sustav kako bi one samostalno mogle ispunjavati i ažurirati svoje podatke koji zatim odlaze na pregled udruzi i bivaju odobreni ili odbijeni. Detaljan opis funkcionalnosti nalazi se u sljedećem potpoglavlju.

2.5. Opis funkcionalnosti sustava

Općeniti opis funkcionalnosti sustava RODA-e podijeljen je prema slučajevima uporabe:

- Prijava

  Prijava omogućava prijavu članovima udruge RODA i medicinskim tehničarima u aplikaciju. Funkcionalnost je potrebna kako bi se korisnici mogli jedinstveno identificirati unutar aplikacije.

- Pregled informacija o rodilištu (medicinski tehničar)

  Pregled informacija o rodilištu omogućava medicinskomu tehničaru prikaz informacija o rodilištu kojem pripada. Funkcionalnost je potrebna kako bi medicinski

tehničar na jednome mjestu mogao vidjeti sve potrebne informacije vezane uz rodilište u kojem trenutačno radi.

- Pregled informacija o rodilištu (RODA)

  Pregled informacija o rodilištu omogućava članu udruge RODA prikaz informacija o određenome rodilištu. Funkcionalnost je potrebna korisnicima udruge kako podatke o rodilištima ne bi trebali skupljati u obliku različitih papira ili *excel* tablica, nego kako bi točne podatke koje je unio medicinski tehničar imali na jednome zaslonu.

- Administracija medicinskih tehničara

  Administracija medicinskih tehničara omogućava članu udruge RODA pregled medicinskih tehničara koji su podijeljeni po rodilištima. Također omogućava dodavanje novoga medicinskog tehničara, uređivanje postojećih medicinskih tehničara te deaktivaciju njihovih računa. Funkcionalnost je potrebna kako bi članovi udruge mogli jednostavnije upravljati korisničkim računima novih i postojećih medicinskih tehničara.

- Administracija članova udruge

  Administracija članova udruge omogućava članu udruge RODA pregled članova udruge, dodavanje novoga člana udruge, uređivanje postojećih članova udruge te deaktivaciju njihovih računa. Funkcionalnost je potrebna kako bi članovi udruge mogli jednostavnije upravljati korisničkim računima novih i postojećih članova udruge.

- Administracija statističkih podataka

  Administracija statističkih podataka omogućava članu udruge RODA pregled i unos statističkih podataka o rodilištima. Funkcionalnost je potrebna kako bi član udruge mogao unijeti statističke podatke za pripadajuće rodilište koji trenutačno ne postoje na portalu te će mu biti omogućen njihov pregled.

- Unos podataka o rodilištu

  Unos podataka o rodilištu omogućava medicinskomu tehničaru unos podataka o rodilištu. Funkcionalnost je potrebna kako taj posao ne bi trebali odrađivati članovi

udruge i za svaku promjenu komunicirati telefonom, nego bi postojalo jedno mjesto na kojem rodilište unosi potrebne podatke koje udruga također može vidjeti.

- Administracija galerije slika

  Administracija galerije slika omogućava medicinskomu tehničaru pregled galerije slika rodilišta, dodavanje slika te brisanje postojećih slika. Funkcionalnost je potrebna kako bi vanjski korisnici na portalu, uz podatke o rodilištu, mogli vidjeti i slike iz pojedinih rodilišta. Funkcionalnost je dodijeljena medicinskim tehničarima kako članovi udruge ne bi trebali ručno dodavati fotografije za svako rodilište, nego medicinski tehničar administrira fotografije samo za svoje rodilište.

- Administracija objave

  Administracija objave omogućava članu udruge RODA odobravanje i odbijanje podataka koji će se objaviti o rodilištu. Funkcionalnost je potrebna članovima udruge da, ako se na prikazu podataka nalazi neki pogrešan podatak ili eksplicitna slika, mogu upravljati tim dijelom i u tome slučaju odbiti podatak, odnosno poslati ga medicinskim tehničarima na ispravak.

- Promjena lozinke

  Promjena lozinke omogućava medicinskomu tehničaru i članu udruge promjenu lozinke. Funkcionalnost je potrebna ako medicinski tehničar ili član udruge zaboravi trenutačnu lozinku.

- Slanje podsjetnika

  Slanje podsjetnika omogućava provjeru sustava koliko dugo podatci o rodilištu nisu mijenjani. Ako podatci nisu mijenjani određeno vrijeme, sustav šalje podsjetnik medicinskomu tehničaru. Funkcionalnost je potrebna kako podatci na javnome portalu ne bi bili zastarjeli.

- Odjava

  Odjava omogućava članu udruge RODA i medicinskomu tehničaru odjavu iz aplikacije. Funkcionalnost je potrebna kako bi se korisnik mogao odjaviti iz aplikacije u bilo kojem trenutku.

- Potvrda ažurnosti podataka

Potvrda ažurnosti podataka omogućava medicinskomu tehničaru potvrdu ažurnih podataka. Funkcionalnost je potrebna kako članovi udruge ne bi morali svake godine usmeno provjeravati s rodilištima jesu li podatci ispravni, nego medicinski tehničar jednim klikom potvrdi ažurnost podataka, što udruzi daje do znanja da su podatci valjani za trenutačnu godinu.

- Izrada plana poroda

  Izrada plana poroda omogućava neregistriranomu korisniku izradu plana poroda. Funkcionalnost je potrebna kako bi trudnice mogle unaprijed napraviti personalizirani plan poroda s kojim bi mogle doći u bolnicu, umjesto da na licu mjesta odlučuju o tim stvarima.

Na temelju poslovnih zahtjeva, trenutačno bez arhitekturalnih pitanja, prepoznatim i gore opisanim slučajevima uporabe određeni su sljedeći prioriteti:

Kritični:

- prijava
- pregled informacija o rodilištu (medicinski tehničar)
- unos podataka o rodilištu
- administracija galerije slika
- odjava.

Važni:

- administracija članova udruge
- pregled informacija o rodilištu (RODA)
- administracija medicinskih tehničara
- potvrda ažurnosti podataka
- slanje podsjetnika
- promjena lozinke
- administracija objave
- administracija statističkih podataka.

Ostali:

- izrada plana poroda.

3. Dizajn *web* aplikacije

Dizajn *web* stranice ili *web* aplikacije važna je faza u izradi *web* aplikacije. Dobro dizajnirane *web* stranice nude mnogo više od same estetike. One privlače posjetitelje i pomažu ljudima razumjeti proizvod, tvrtku i robnu marku nizom pokazatelja koji obuhvaćaju vizuale, tekst i korisničke interakcije. Faza dizajna obuhvaća nekoliko različitih aspekata: dizajn korisničkoga sučelja (engl. *User Interface*, skraćeno UI), dizajn korisničkoga iskustva (engl. *User Experience*, skraćeno UX), pisanje sadržaja i grafički dizajn.

Dizajn korisničkoga sučelja predstavlja sve ono s čim korisnik može ostvariti interakciju, kao što su boje, pisma, gumbi, navigacija, responzivni dizajn itd. Dobar UI dizajn privlači korisnike i jamči izvrsno korisničko iskustvo. On isto tako promiče kvalitete *branda*, osiguravajući pouzdanost dizajna i njegovo stilsko zadovoljavanje. Cilj je dizajnera korisničkoga sučelja napraviti uvjerljivo korisničko sučelje koje potiče emocionalnu reakciju korisnika. Smisao UI dizajna proizlazi iz izrade instinktivnoga susreta koji od korisnika ne zahtijeva pretjerano razmišljanje[1].

Dizajn korisničkoga iskustva, s druge strane, usredotočen je na iskustvo korištenja aplikacije. On je u teoriji kognitivna znanost koja se koristi pretežito u digitalnoj industriji. Ključ je UX prakse to što počinje identificiranjem ljudskih potreba, odnosno potreba korisnika sustava[2]. Neka od pitanja koja si UX dizajner postavlja su: je li aplikacija komplicirana za korištenje, je li spora, je li korisnik razočaran prilikom korištenja njom? UX dizajn nije strogo vezan uz vizuale, nego uz cjelokupni osjećaj korisničkoga iskustva.

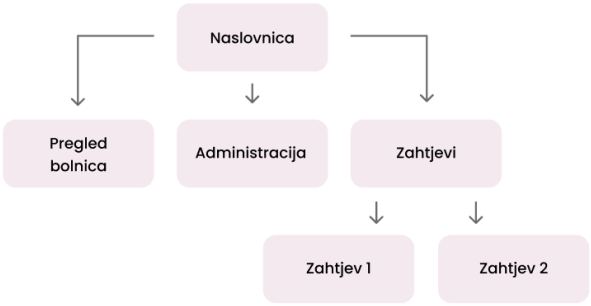
3.1. Dijagram *web* aplikacije

Prikaz organizacije stranica na *web* sjedištu najčešće se prikazuje dijagramom stabla *web* stranice, odnosno *web* aplikacije. Nadređene stranice prikazane su iznad podređenih, a najčešće je korištena nadređena stranica naslovnica (engl. *Home Page*). Ispod nje nalaze se podređene stranice ukazujući na međuodnos preostalih podstranica i načina na koji im se pristupa. Dijagrami stabla *web* sjedišta mogu biti jednostavni ili složeni, ovisno o veličini

[1] UX / UI Design: Introduction Guide To Intuitive Design And User-Friendly Experience, stranica 20.
[2] Lean UX: Designing Great Products with Agile Teams, stranica 8.

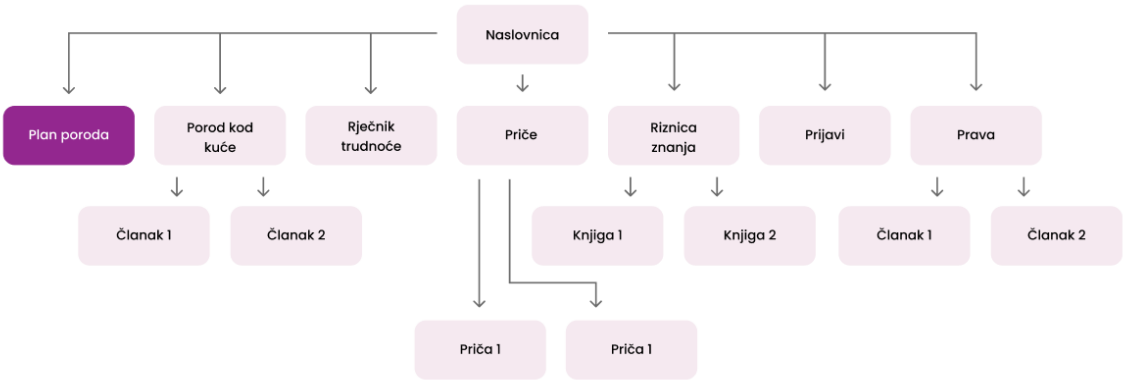
web sjedišta i potrebama razvojnoga tima. Također, oni pomažu posjetiteljima u kretanju velikim i kompleksnijim *web* sjedištima prikazujući njihovu cjelokupnu strukturu.

Za tri definirane ciljane skupine razlikuju se i tri dijagrama *web* sjedišta. Na slici je (Slika 3.1Slika 2.1) prikazan dijagram *web* aplikacije za korisnika Udruge. Na vrhu se nalazi naslovnica koja ujedno služi filtriranju bolnica i podataka o bolnicama. S naslovnice se je moguće kretati u dvije zasebne stranice: stranicu *Administracija* i stranicu *Zahtjevi*. Sa stranice *Zahtjevi* moguć je pregled pojedinačnoga zahtjeva te su oni međusobno odvojeni.



Slika 3.1 Dijagram *web* aplikacije za korisnika Udruge

Sljedeća su ciljana skupa buduće majke. Dijagram sa slike (Slika 3.2) prikazuje strukturu već postojećega portala Udruge RODA. Dio koji je ostvaren u sklopu ovoga završnog rada izrada je plana poroda, označen tamnoljubičastom bojom na dijagramu, kojemu se portalom može javno pristupiti.



Slika 3.2 Dijagram portala Udruge RODA

Treća su ciljana skupina rodilišta. Za rodilišta ne postoji hijerarhijska struktura, nego je prisutna jedino naslovnica, na kojoj je moguće obaviti sve potrebne izmjene podataka.

3.2. Žičani okvir

Žičani je okvir (engl. *Wireframe*) shema ili skica zaslona koja pomaže dizajnerima i programerima prilikom komunikacije o generalnoj strukturi softvera. Jedan zaslon u aplikaciji može se napraviti na mnogo načina, ali samo neka od rješenja prenijet će ispravnu poruku korisniku i olakšati korištenje aplikacijom. Upravo iz toga razloga izrada žičanoga okvira nije zanemariv dio procesa, već neophodan korak prilikom dizajniranja *web* aplikacije.

Za ovu *web* aplikaciju *Wireframe* i dizajn izrađivali su se za tri korisničke uloge, odnosno persone. Uloge su sljedeće:

1. član Udruge RODA
2. medicinski tehničar
3. neregistrirani korisnik.

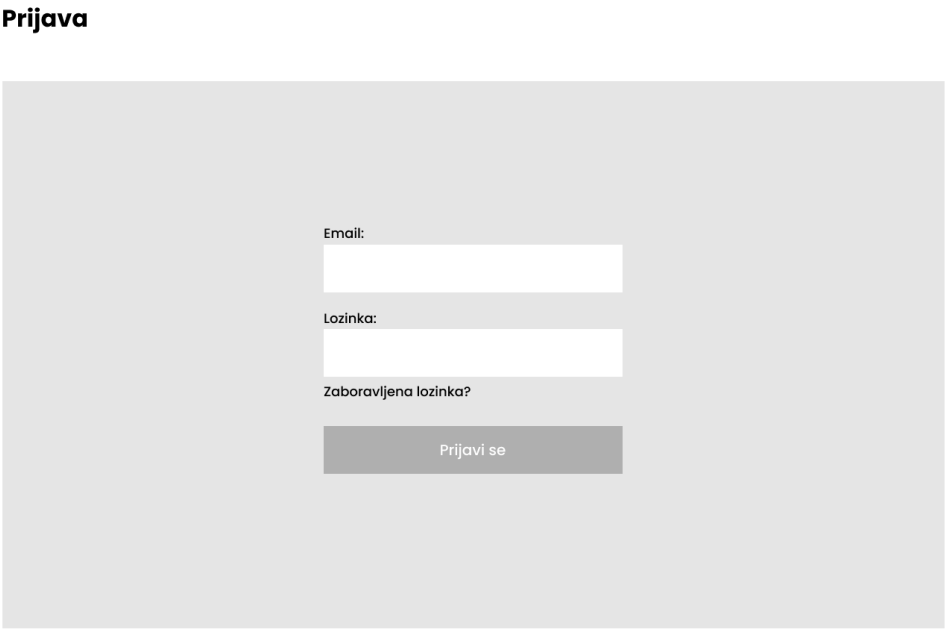
Član udruge RODA primarni je akter. Prijavljuje se korisničkim imenom i lozinkom. On vrši registraciju drugoga aktera, odnosno medicinskoga tehničara. Član udruge RODA registrira nova rodilišta te šalje korisničke podatke za rodilište. Članu udruge RODA prikazan je pregled svih medicinskih tehničara po rodilištu te može administrirati njima kao i ostalim članovima udruge RODA. Također, ima pristup statističkim podatcima koje može unositi i mijenjati. Prije objave odobrava podatke koje unese medicinski tehničar.

Medicinski tehničar drugi je akter. Može samostalno uređivati podatke o rodilištu kojemu pripada i pregledati te podatke. Također može pregledavati, dodavati i brisati slike rodilišta.

Neregistrirani korisnik može izraditi plan poroda i mijenjati ga. Također može pristupiti pregledu statističkih podataka o rodilištima.

Član Udruge RODA i medicinski tehničar mogu se ulogirati u aplikaciju. Budući da se radi o prijavi, na sredini zaslona nalazi se forma za prijavu koju sačinjavaju: polje za unos *e-maila*, polje za unos lozinke i akcijski gumb (engl. *Call To Action*, skraćeno CTA), odnosno u ovome slučaju *Prijavi se*. Kako bi se korisnik mogao prijaviti, potrebno je da bude registriran u sustavu, odnosno da posjeduje podatke za prijavu. Sustav tada utvrđuje jesu li

podatci ispravni i preusmjerava korisnika na početnu stranicu. Tijekom dizajna forme za prijavu u sustav važno je imati na umu i poruke greške koje se javljaju prilikom neispravnoga unosa podataka.



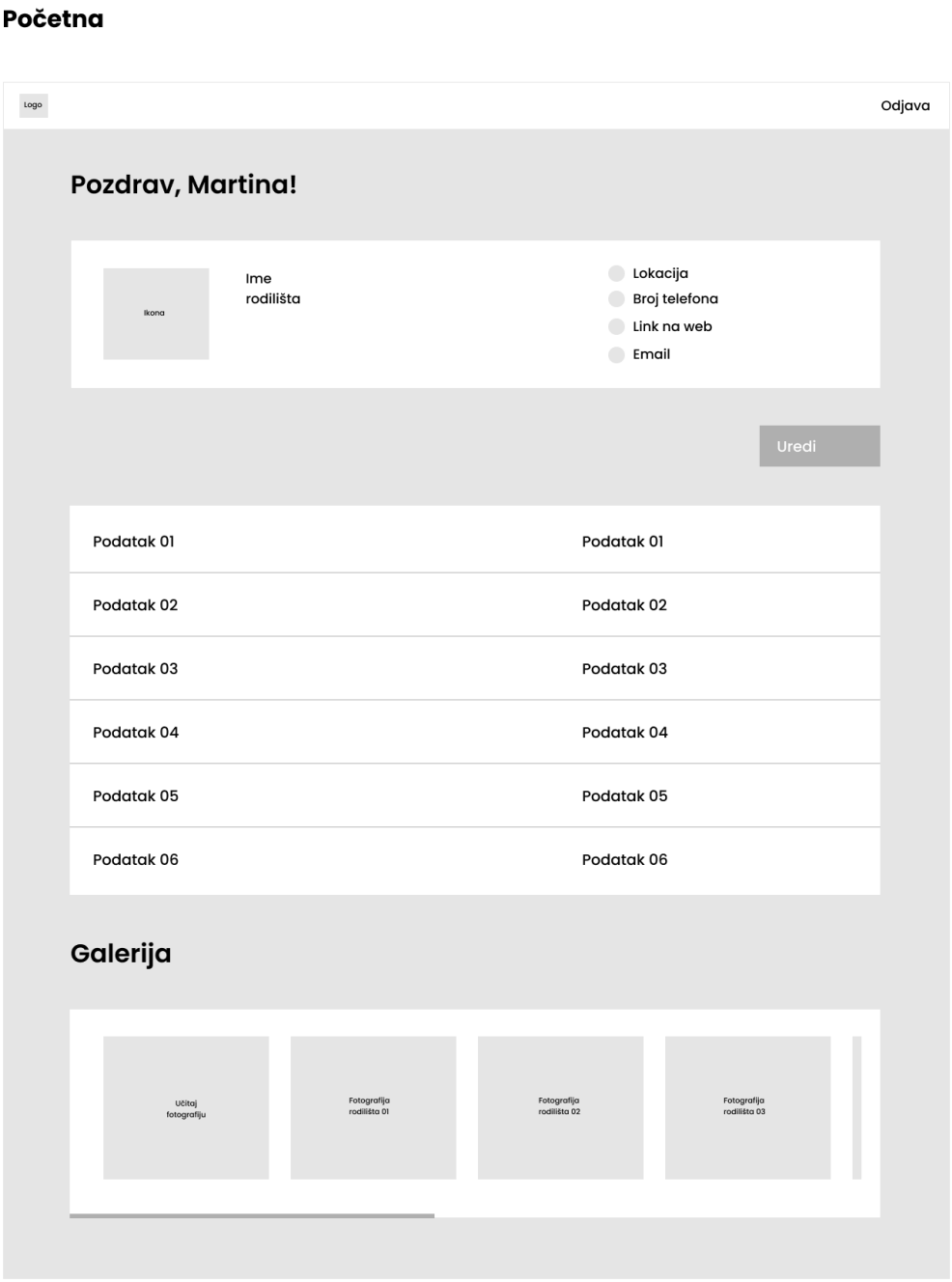
Slika 3.3 *Wireframe* prijave u desktop verziji

Nakon uspješne prijave medicinskoga tehničara u aplikaciju, sustav provjerava kojemu rodilištu korisnik pripada. Zatim ga sustav preusmjerava na početnu stranicu s informacijama o rodilištu kojemu pripada. Stranica s podatcima o rodilištu sastoji se od sljedećih elemenata:

- navigacijske trake
- pozdravne poruke
- panela s detaljima o rodilištu
- gumba *Uredi*
- tablice s prikazom ankete
- galerije fotografija rodilišta.

Navigacijska traka sadržava logo Udruge s lijeve strane te se klikom na njega, na bilo kojem mjestu u aplikaciji, korisnika preusmjerava na početnu stranicu. Osim logotipa, na desnoj strani navigacijske trake nalazi se poveznica *Odjava*, klikom na koju se korisnik odjavljuje iz aplikacije. Ispod navigacijske trake prikaza pozdravna je poruka koja se sastoji od

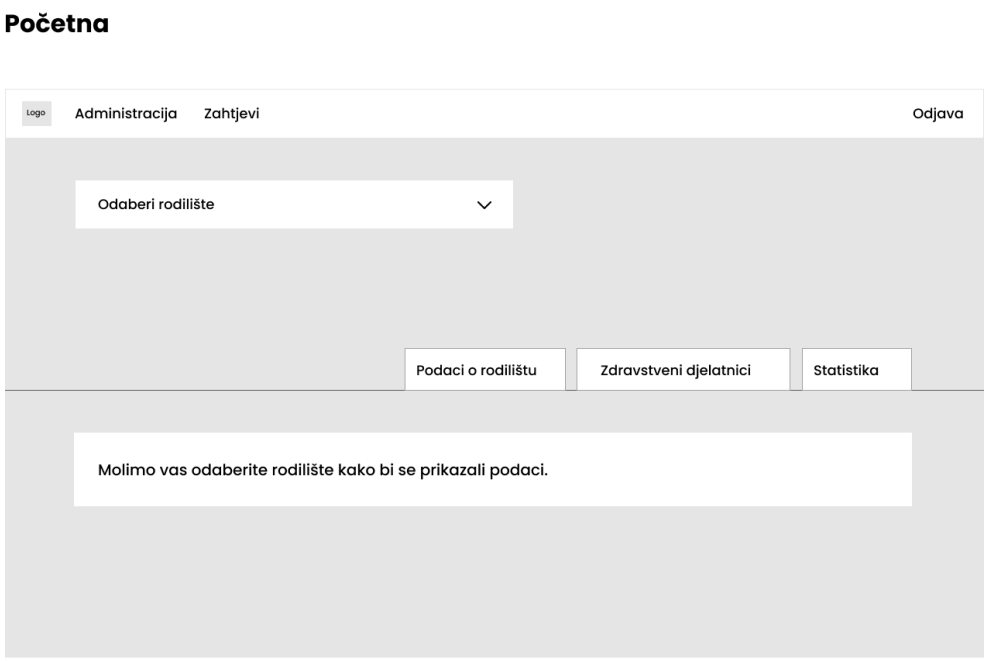
pozdrava i imena medicinskoga tehničara prijavljenoga u sustav. Sljedeći je panel s detaljima o rodilištu u kojem se nalazi ime rodilišta u kojem radi prijavljeni medicinski tehničar i informacije o njemu s lijeve strane panela. Gumb *Uredi* nalazi se neposredno ispod panela te se klikom na njega otvara prozor u kojem je moguće urediti podatke u tablici. Tablica se sastoji od dvaju stupaca teksta, prostire se cijelom dužinom zaslona i glavni je element ovoga prikaza. Ispod tablice nalazi se galerija s mogućnošću dodavanja fotografija i pregledom dodanih fotografija. Galerija sadrži horizontalni klizač čijim se pomicanjem mogu vidjeti sve dodane fotografije u galeriji.



Slika 3.4 *Wireframe* početne stranice medicinskoga tehničara u desktop verziji

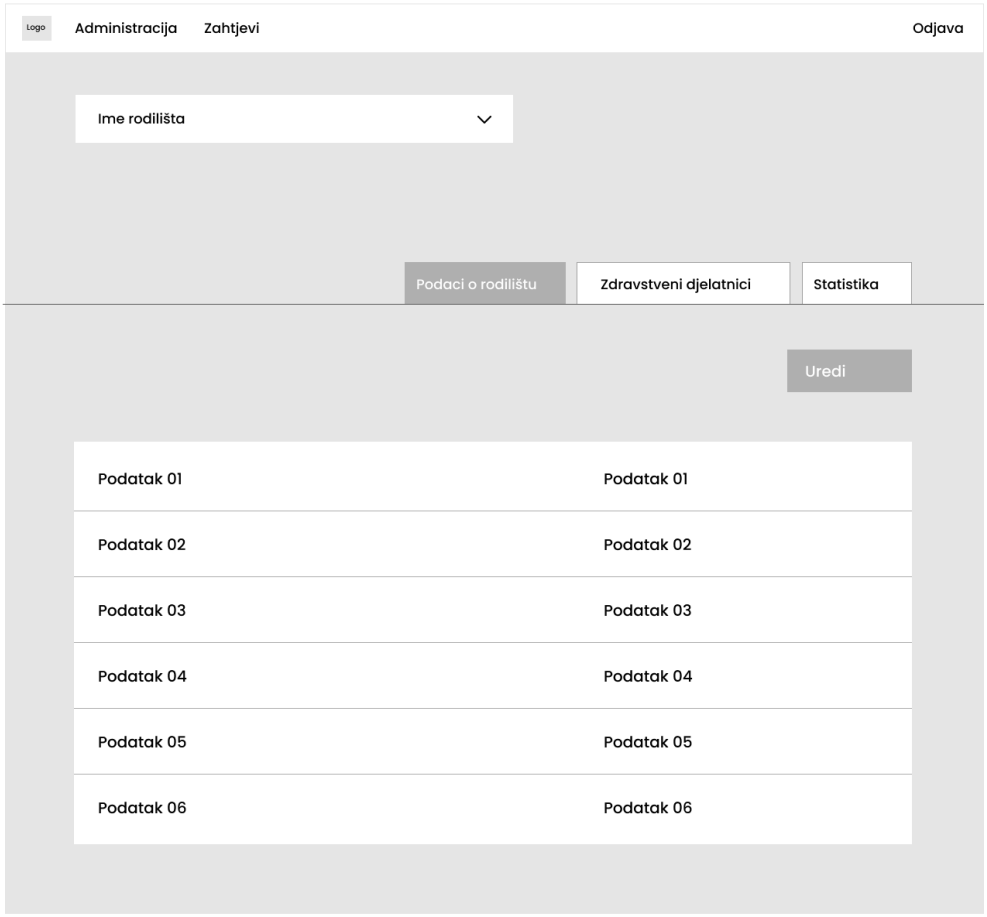
Administrator udruge, nakon uspješne prijave, nalazi se na početnoj stranici. Sustav prikazuje stranicu s osnovnim elementima i aktivnom karticom *Podatci o rodilištu*. U kartici *Podatci o rodilištu* prikazuje se poruka „Nema podataka. Molimo Vas da odaberete rodilište iz padajućega izbornika na vrhu stranice.” Korisnik tada odabire rodilište iz padajućega izbornika koji se nalazi ispod navigacijske trake. Nakon odabira rodilišta na zaslonu se

prikazuju podatci o rodilištu. Za svako rodilište administrator može pregledavati njegove podatke, popis zdravstvenih djelatnika / medicinskih tehničara koji su ondje zaposleni te statističke podatke odabrana rodilišta. Spomenuti dijelovi informacija izvedeni su u obliku kartica za pregledniji prikaz. U svakoj kartici nalazi se tablica s informacijama i CTA gumb ovisno o pripadajućoj akciji. Osim CTA gumba, u kartici *Statistika*, nalazi se padajući izbornik s mogućnošću odabira godine na koju se odnose statistički podatci prikazani u tablici. Navigacijska traka za administratora Udruge sastoji se od logotipa, dviju poveznica koje vode na zasebne stranice i gumba *Odjava*. Klikom na logotip korisnik se preusmjerava na početnu stranicu aplikacije, a klikom na gumb *Odjava* odjavljuje ga se iz aplikacije.

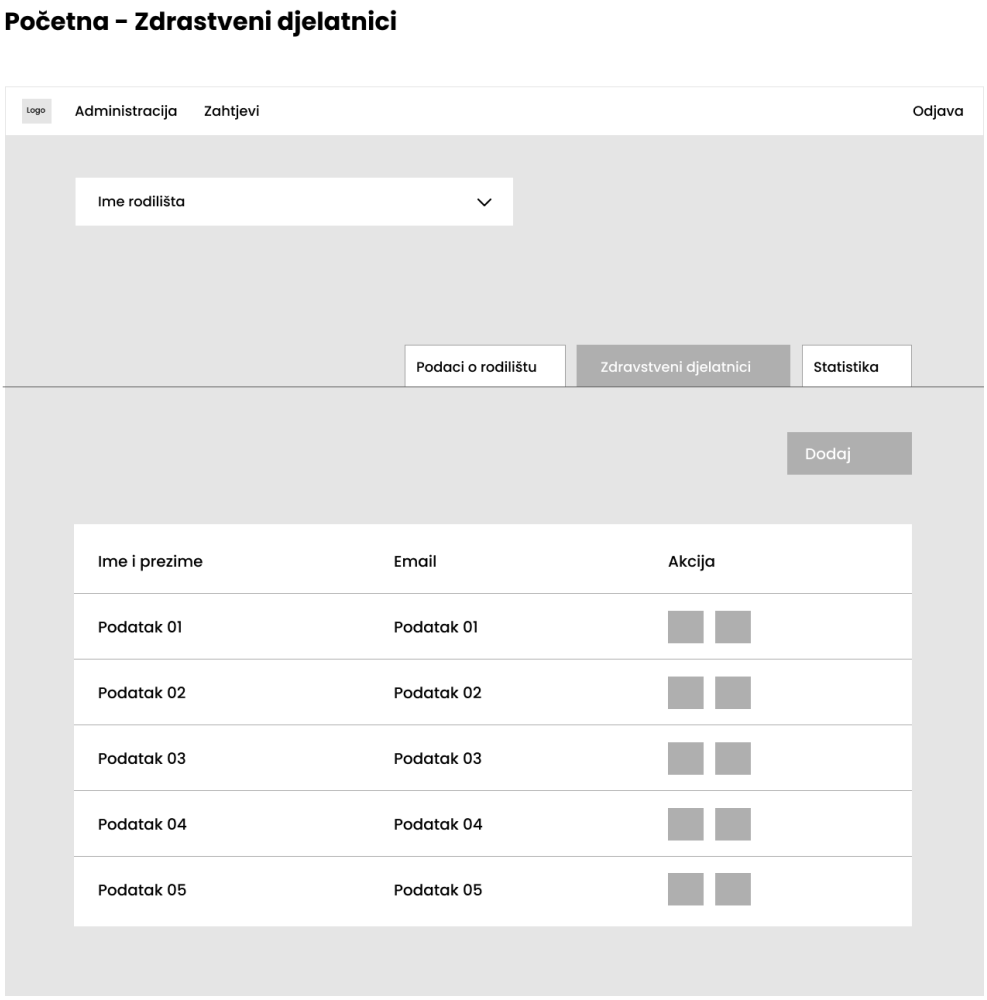


Slika 3.5 *Wireframe* početne stranice administratora Udruge RODA u desktop verziji

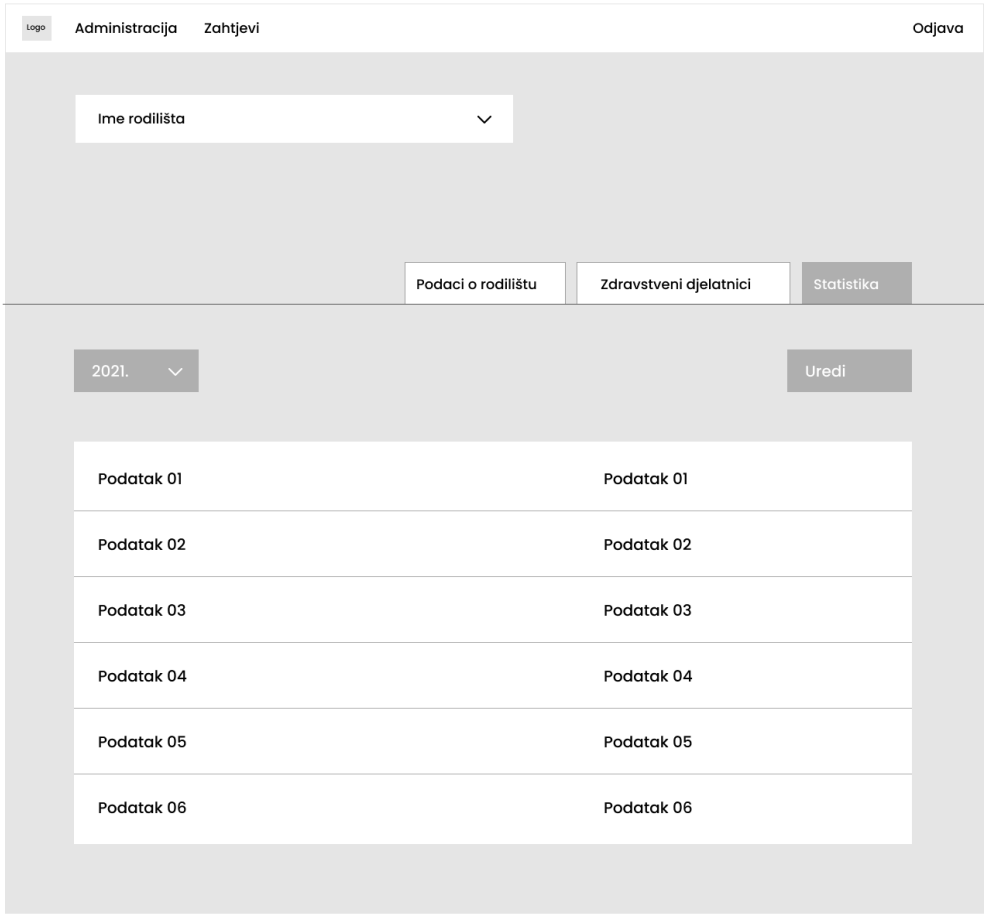
Početna - Podaci o rodilištu



Slika 3.6 *Wireframe* početne stranice s odabranim rodilištem u desktop verziji



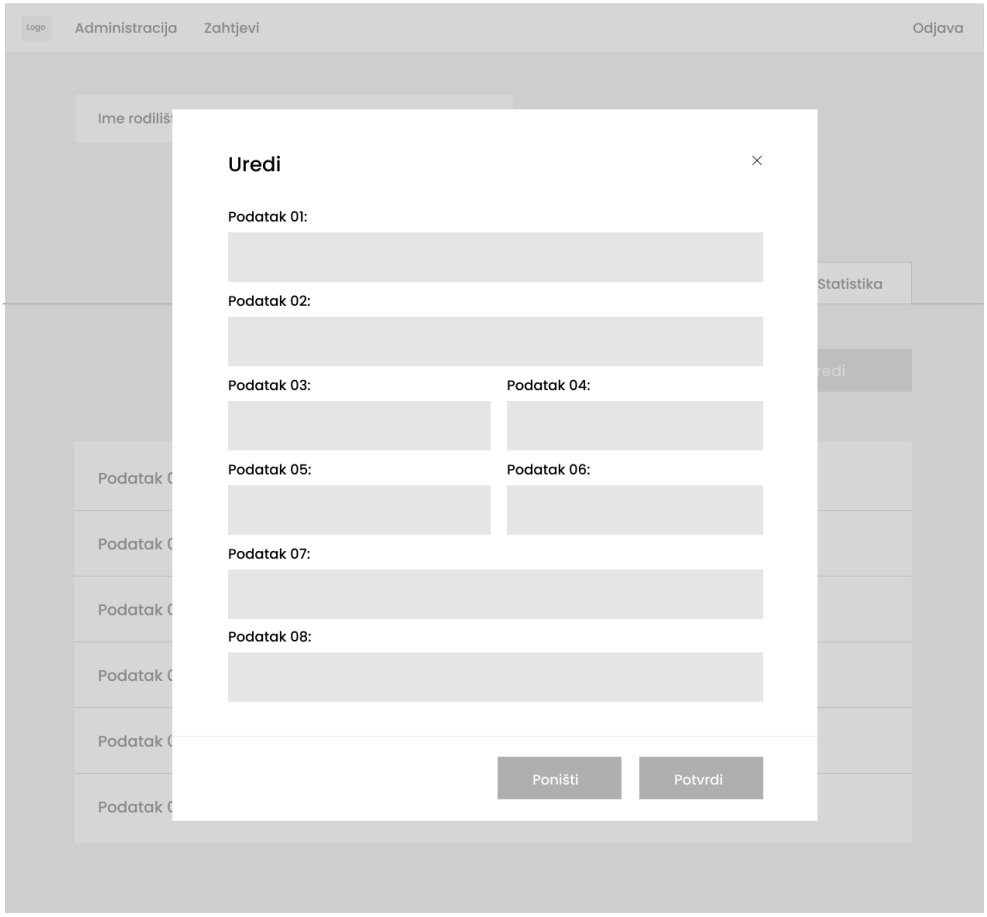
Slika 3.7 *Wireframe* početne stranice s označenom karticom *Zdravstveni djelatnici* u desktop verziji



Slika 3.8 *Wireframe* početne stranice s označenom karticom *Statistika* u desktop verziji

Ako se administrator udruge nalazi na kartici *Podatci o rodilištu*, klikom na CTA gumb *Uredi* otvara se skočni prozor koji omogućava uređivanje informacija o rodilištu. Skočni prozor sadržava naslov i formu koju sačinjavaju polja za unos, odnosno u ovome konkretnom slučaju, promjenu podataka. Raspored polja za unos napravljen je u skladu s dužinom informacije za unos te je svako polje udaljeno od drugoga za određenu vrijednost kako bi se postigla preglednost i urednost forme. U gornjem lijevom kutu postavljena je ikona klikom na koju se zatvara skočni prozor. Na dnu skočnoga prozora nalaze se CTA gumbi za poništavanje, odnosno potvrđivanje promjene podataka.

Početna - Uredi statističke podatke

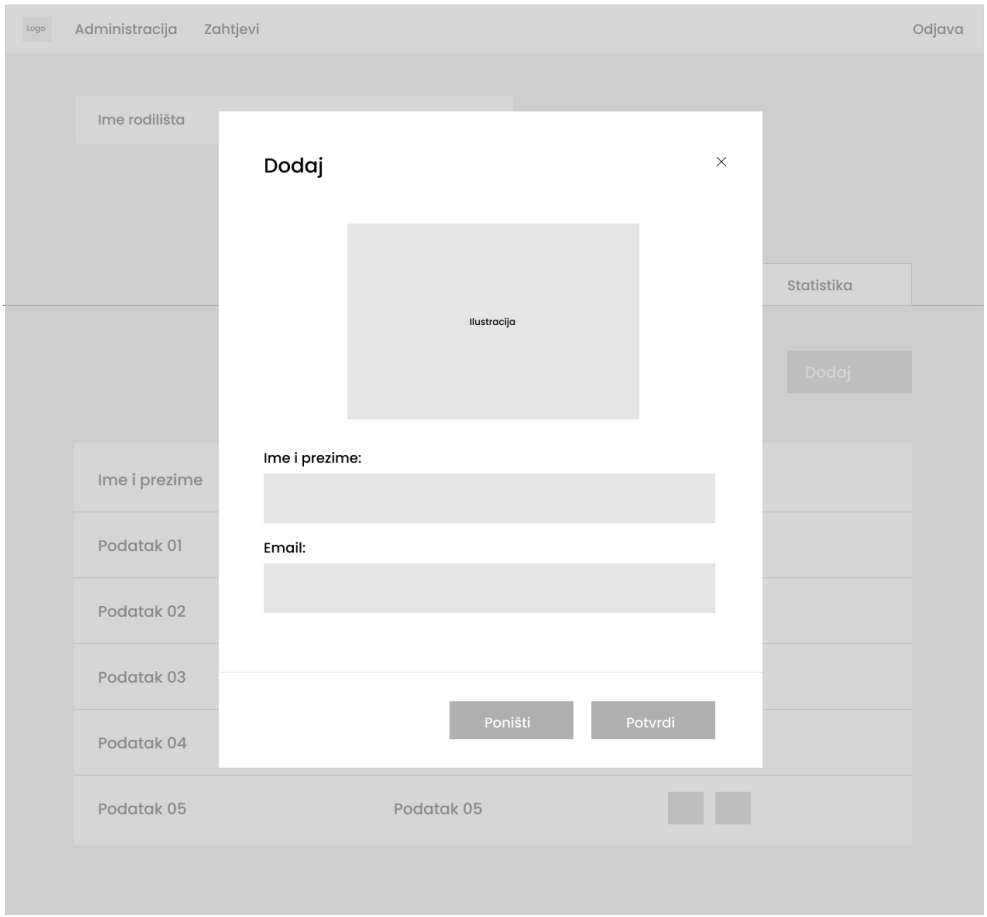


Slika 3.9 *Wireframe* početne stranice s označenom karticom *Podatci o rodilištu* i pripadajućim skočnim prozorom u desktop verziji

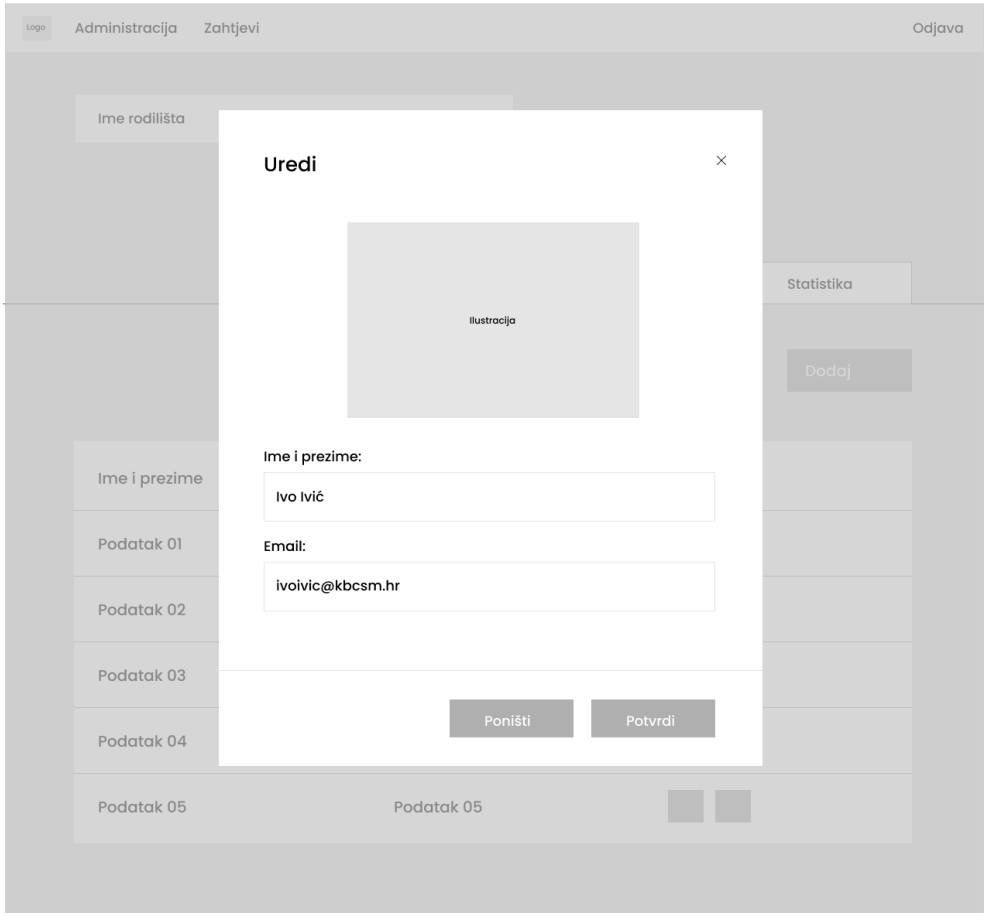
Administracija medicinskih tehničara riješena je u kartici *Zdravstveni djelatnici*. Sustav prikazuje listu medicinskih tehničara koji upravljaju računom odabrana rodilišta. Tablica koja prikazuje popis medicinskih tehničara sastoji se od triju stupaca. Informacije su dovoljno razmaknute kako bi bile što čitljivije i kako bismo postigli preglednost. Iznad tablice nalazi se CTA gumb, koji služi dodavanju medicinskoga tehničara. Klikom na njega otvara se skočni prozor za dodavanje novoga medicinskog tehničara. Spomenuti skočni prozor sadrži naslov, prikladnu ilustraciju te dva polja za unos podataka. Forma je jednostavna, sadržajno prihvatljiva i vrlo intuitivna. Gornji lijevi kut sadrži ikonu za izlaz iz skočnoga prozora, ali korisnik može izaći i ako klikne bilo gdje izvan zaslona. Ispod polja za unos podataka nalaze se dva CTA gumba, kojima se potvrđuje dodani medicinski tehničar

ili se odustaje od radnje. Opisani scenarij sadrži dvije iznimke. Ako korisnik upiše već postojeću *e-mail* adresu, prikazuje mu se poruka „Već postoji korisnički račun s navedenom *e-mail* adresom." Polje za unos *e-maila* u tome slučaju poprima obrub crvene boje, iste kao i poruka koja se ispisuje ispod polja za unos. Ako korisnik klikne na gumb *Poništi*, sustav ne sprema novoga medicinskog tehničara i scenarij završava. S lijeve strane tablice nalaze se CTA gumbi koji služe uređivanju i deaktivaciji medicinskoga tehničara. Klikom na lijevi gumb otvara se skočni prozor kao iz prethodnoga scenarija, samo s već popunjenim vrijednostima. Korisnik uređuje podatke te klikom na gumb *Potvrdi* sustav sprema promjene i preusmjerava korisnika na stranicu pregleda medicinskih tehničara. Ako korisnik klikne na gumb *Poništi*, sustav ne sprema informacije o medicinskome tehničaru. Desni gumb služi za deaktivaciju medicinskoga tehničara.

Početna - Dodaj zdravstvenog djelatnika



Slika 3.10 *Wireframe* sa skočnim prozorom za dodavanje zdravstvenoga djelatnika



Slika 3.11 *Wireframe* sa skočnim prozorom za uređivanje zdravstvenoga djelatnika

3.3. Dizajn-sistem

Dizajn-sistem skup je pravila koja se odnose na grafičke elemente uključene u aplikaciju. On ujedno predstavlja dokumentaciju koja objašnjava vizualni aspekt elemenata korisničkoga sučelja i osnovu za dizajnerske i implementacijske odluke koje utječu na aplikaciju. Kvalitetan dizajn-sistem treba biti detaljan, informativan, organiziran, prilagodljiv za buduće scenarije, ažuran i jednostavan za korištenje. Održavanje grafičke dosljednosti u aplikaciji igra ključnu ulogu, stoga je u stilskome vodiču nužno definirati sve grafičke varijante elemenata i komponenti. S obzirom na to da se ova *web* aplikacija naslanja na već postojeći portal Udruge RODA, eksperimentiranje novim bojama ovdje nije bio

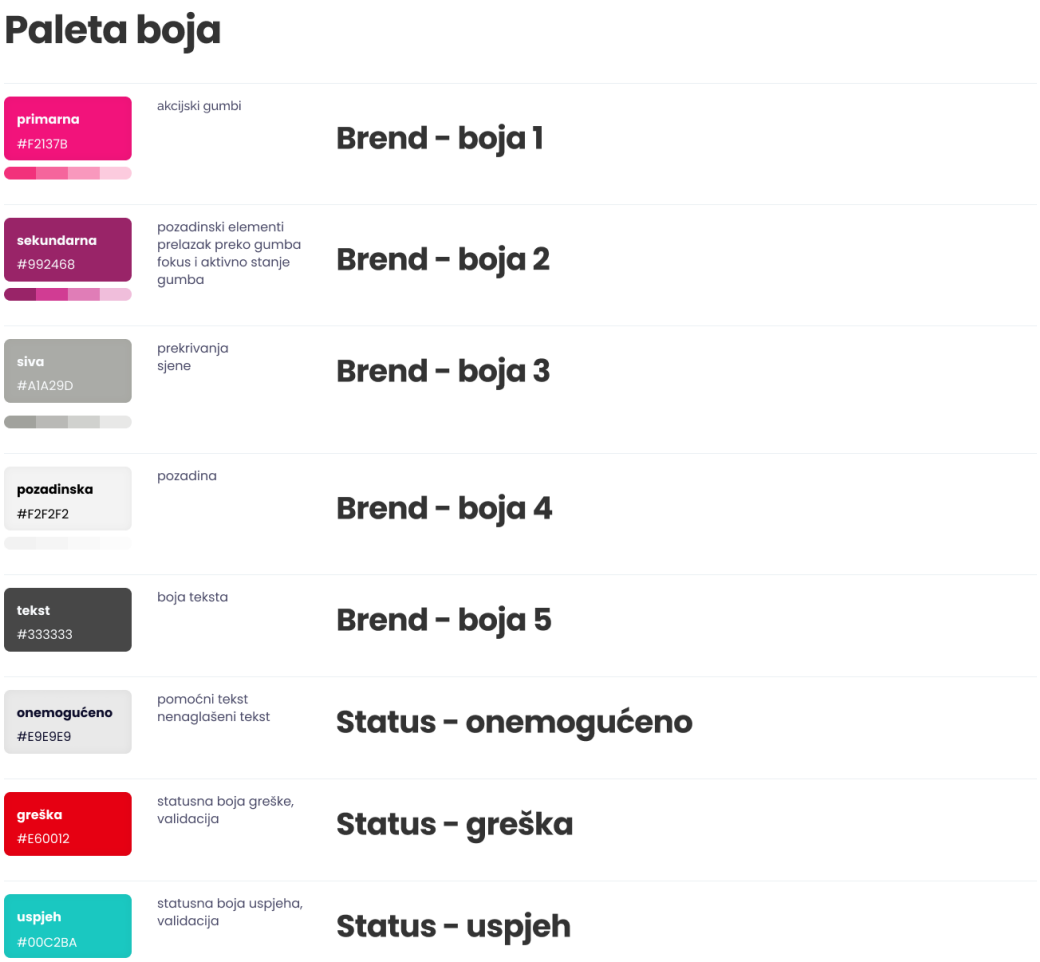
slučaj, nego su one u skladu s portalom već jasno definirane. Elementi koji su se uzeli u obzir prilikom izrade stilskoga vodiča su boja, tipografija, raspored (engl. *layout*), ikone i komponente.

Softverski alat korišten za izradu stilskoga vodiča je Figma. Figma je moćan dizajnerski alat u kojem se s lakoćom mogu stvoriti vizualne reprezentacije *web* stranica i aplikacija, ali i mnoštvo ostalih elemenata grafičkoga dizajna poput logotipa i postova za društvene mreže. Također, Figma je *web* aplikacija, što znači da se njom možemo koristiti iz *web* preglednika.

3.3.1. Boja

Boje, između ostaloga, mogu potaknuti prodaju dodirujući podsvjesne ljudske emocije. Cilj je skoro svake *web* stranice privući pozornost posjetitelja i izazvati određene reakcije i emocije s ciljem povećanja broja klikova, pretplata, registracije i/ili kupnje. Iz toga razloga postoji znanstvena grana koja se bavi proučavanjem utjecaja boja na ljudsko ponašanje, a zove se psihologija boja. Psihologija boja proučava usku povezanost vrijednosti, emocija, čak i fizioloških reakcija s određenim bojama. Ona je moćan alat u *web* dizajnu koji pomaže u poticaju korisničkoga angažmana na *web* stranici, ali i izvan nje.

Odabrano je devet glavnih boja koje će se koristiti u aplikaciji. Osim devet glavnih boja u nekim slučajevima odabrani su i podtonovi primarne, sekundarne i sive boje. Primarna boja ima heksadekadsku vrijednost #F2137B te se najčešće koristi za CTA gumbe i naslove u skočnim prozorima. Sekundarna boja, s heksadekadskom vrijednošću #992468, najčešće prikazuje stanje gumba u trenutku prelaska preko njega, aktivna i fokusirana stanja polja za unos teksta u formama, a njezine svjetlije nijanse koriste se za pozadinske elemente. Treća odabrana boja ima heksadekadsku vrijednost #A1A29D i koristi se kod preklopnih elemenata i sjena. Sljedeća boja, #333333, koristi se za običan tekst u aplikaciji dok se boja #E9E9E9 koristi za pomoćne i nenaglašene tekstove. Preostale su boje statusne, čija je uporaba najznačajnija u formama za unos podataka, a to su crvena, s heksadekadskom vrijednošću #E60012, i zelena, s heksadekadskom vrijednošću #00C2BA. Statusne boje važne su prilikom priopćavanja razine ozbiljnosti o informacijama korisnicima. Različite boje omogućuju korisnicima bržu procjenu i identifikaciju statusa i u skladu s tim odgovarajući odgovor.



Slika 3.12 Dizajn-sistem – paleta boja

3.3.2. Tipografija

Sljedeći je element u stilskome vodiču tipografija. Mnogo toga treba uzeti u obzir prilikom odabira pisma za *web* aplikaciju. Iako ne postoji formula, dizajneri se mogu koristiti sljedećim pravilima kako bi pronašli pravo pismo za ugodno iskustvo čitanja:

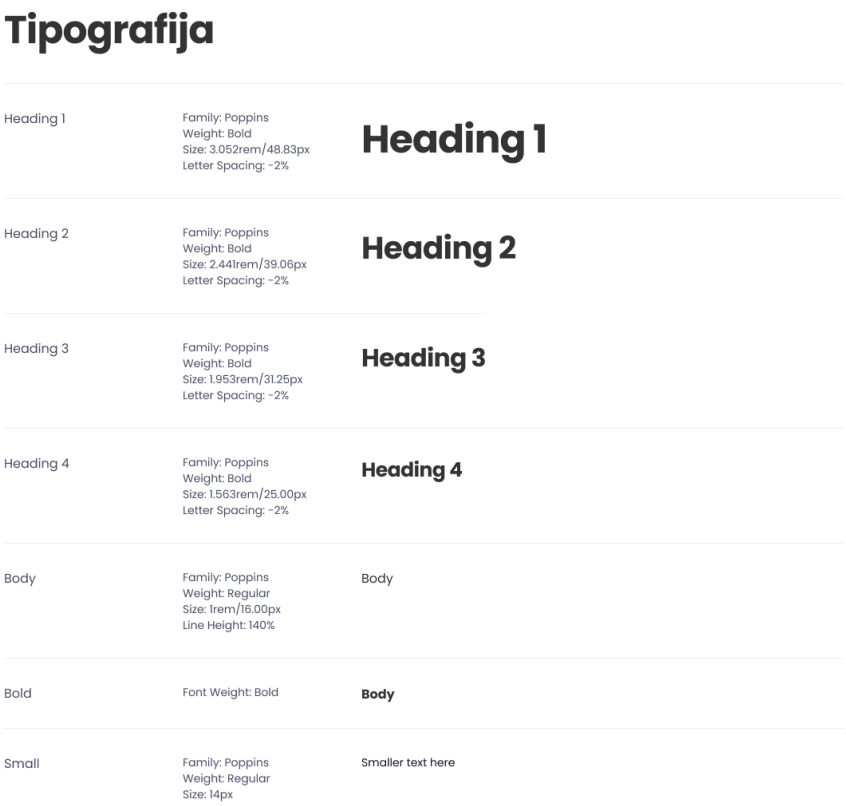
- odabir jasnoga i prepoznatljivoga oblika slova

Tekst je najčitljiviji onda kada oko može lako razlikovati slova. Neka su slova dizajnirana tako da budu glatka, ritmična i ugodna oku. Oblici koji se previše ponavljaju mogu učiniti tekst nečitljivim pa bi svako slovo trebalo imati poseban oblik. Slova bi trebala biti prozračna i otvorena.

- formatiranje teksta za najvišu razinu čitljivosti

Formatiranje teksta na zaslonu od jednake je važnosti kao i odabir slova. Odluke poput boja i veličine pisma pridonose iskustvu čitanja. Testiranje dizajna na različitim vrstama zaslona ključno je za osiguravanje ugodnoga i čitljivoga iskustva čitanja. Za razmak između redova teksta postoji pravilo: što je pismo manje, to je razmak između redova uži.

Pismo koje se koristi u cijeloj aplikaciji je Poppins. Poppins je sans serifno pismo s podrškom za latinski sustav pisanja. Diferencijacija važnosti tekstovnih elemenata postignuta je različitom veličinom i debljinom pisma.



Slika 3.13 Dizajn-sistem – tipografija

3.3.3. Raspored

Mreža *web* stranice (engl. *website grid*) sustav je kojim se organizira sadržaj na stranici kako bi se postigli red i usklađenost. Mreža čini kostur korisničkoga sučelja i pomaže u upravljanju omjerima između svakoga elementa, a ne samo cijelim izgledom stranice. Mrežni sustav poravnava elemente stranice na temelju uzastopnih stupaca i redaka. Nakon postavljanja toga strukturnog okvira postavljaju se tekst, slike i ostali elementi na uredan način unutar sučelja. Stupci su okomiti blokovi koji se protežu po visini zaslona i što ih je

više, mreža je fleksibilnija. Širina stupca varira od šezdeset do osamdeset piksela. Za ovu aplikaciju odabrana je mreža od dvanaest stupaca na stolnome računalu, odnosno četiri stupca na mobilnome uređaju. Osim dvanaest stupaca u mreži, za stolna računala definirane su bočne margine od 165 piksela i razmak između stupaca (engl. *gutter*) od dvadeset piksela. Na mobilnim uređajima bočne margine široke su 45 piksela dok je razmak među stupcima isti kao i kod mreže za stolna računala.

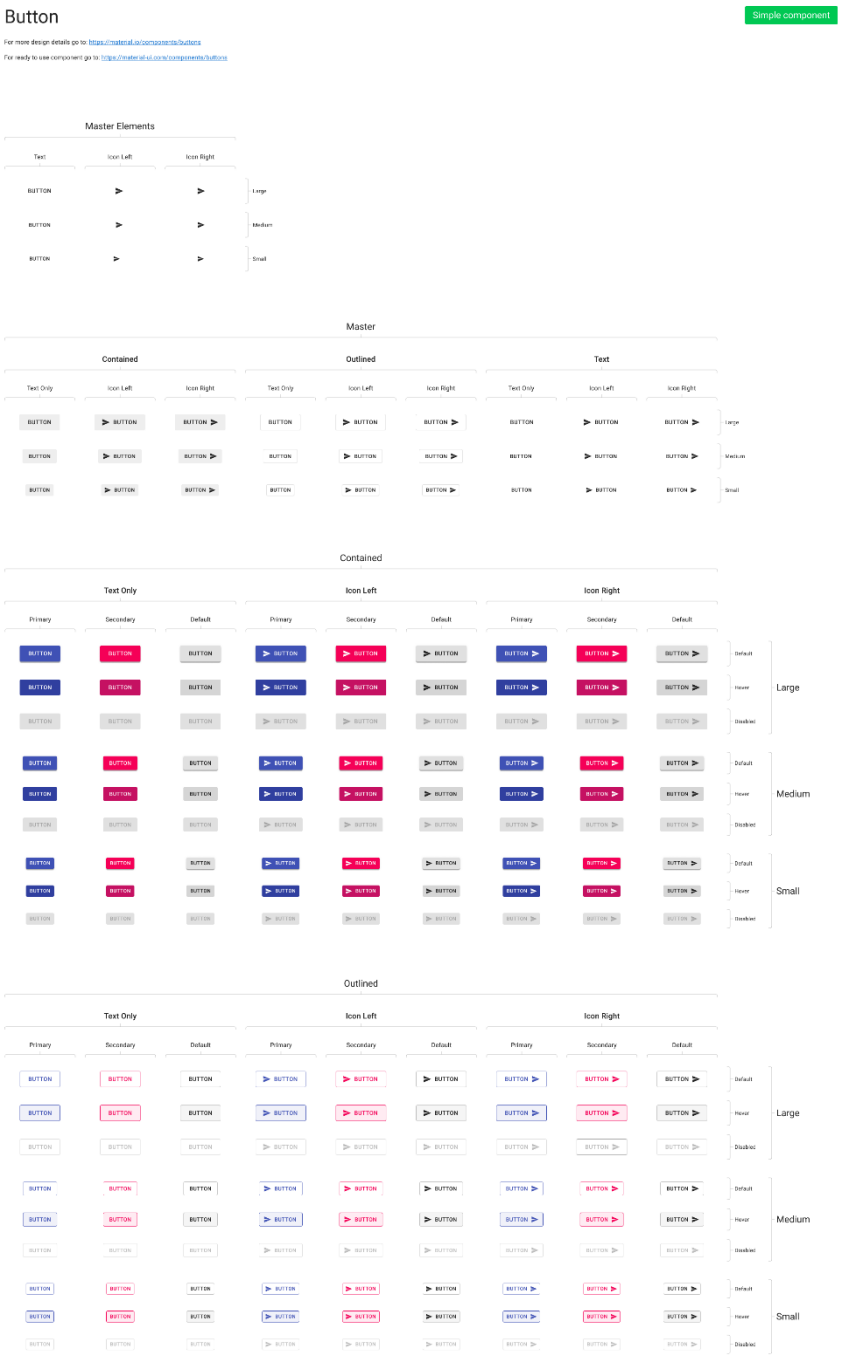
Slika 3.14 Dizajn-sistem – *Website grid* za stolna računala

Slika 3.15 Dizajn-sistem – *Website grid* za mobilne uređaje

3.3.4. Ikone i komponente

Ikone i komponente za ovu *web* aplikaciju koristile su se iz popularne, moderne i prilagodljive knjižnice React komponenti naziva Material UI (skraćeno MUI). Kao što i samo ime sugerira, Material UI poklapa se s *Googleovim* materijalnim dizajnom (engl. *Material Design*). Uz komponente Material UI nudi i jedinstveni set ikona koje se vrlo lako implementiraju u dizajn. Material UI tema potpuno je prilagodljiva i može joj se pristupiti globalno, što znači da ju prate sve komponente unutar aplikacije. Tema služi za definiranje svojstava poput boje, razmaka i prijelomnih točaka. Knjižnica se redovito ažurira s ispravcima grešaka i korisnim nadogradnjama. Službena dokumentacija je organizirana i lako dostupna.

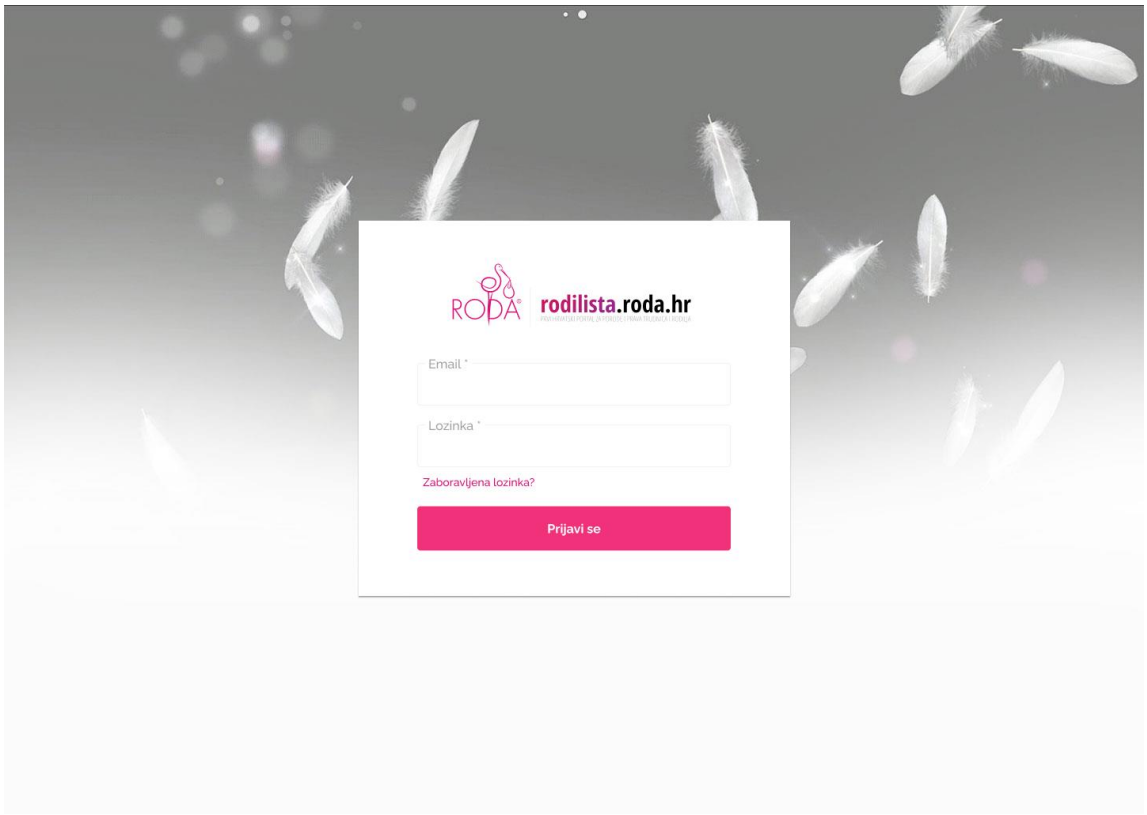
Slika 3.16 Dizajn-sistem – ikone iz knjižnice Material UI



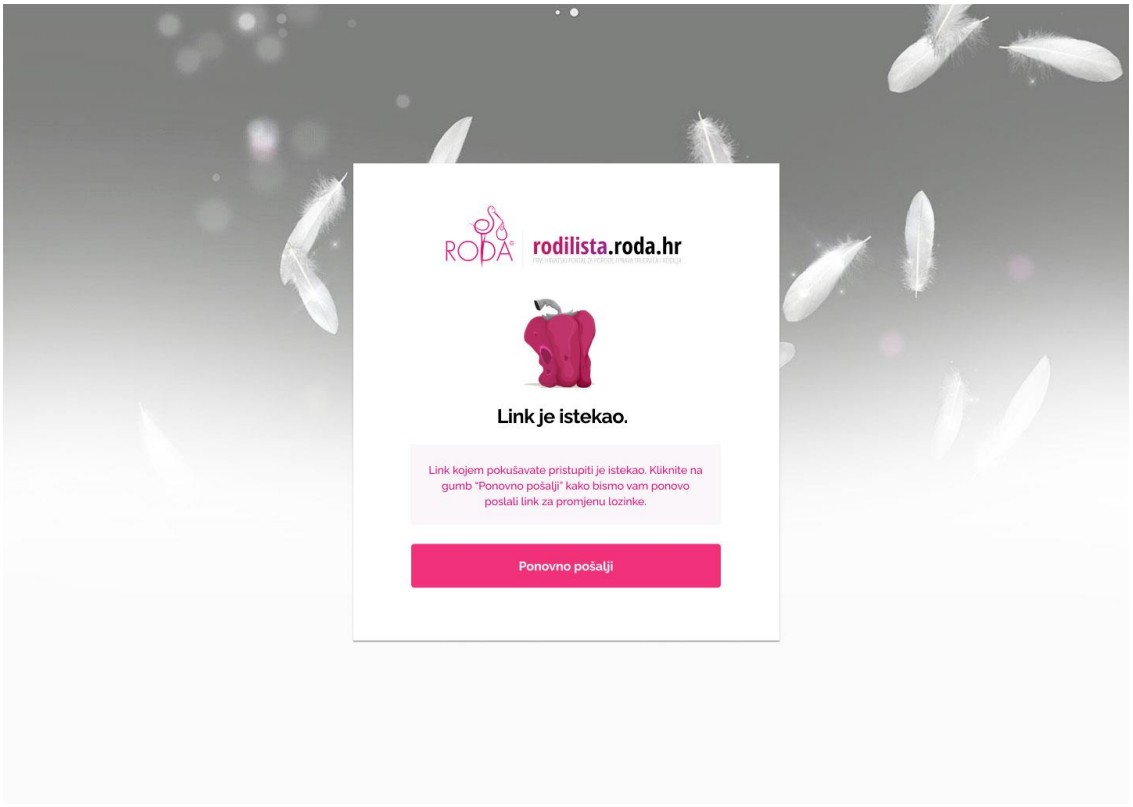
Slika 3.17 Dizajn-sistem – komponenta gumba iz knjižnice Material UI

3.4. Konačan dizajn

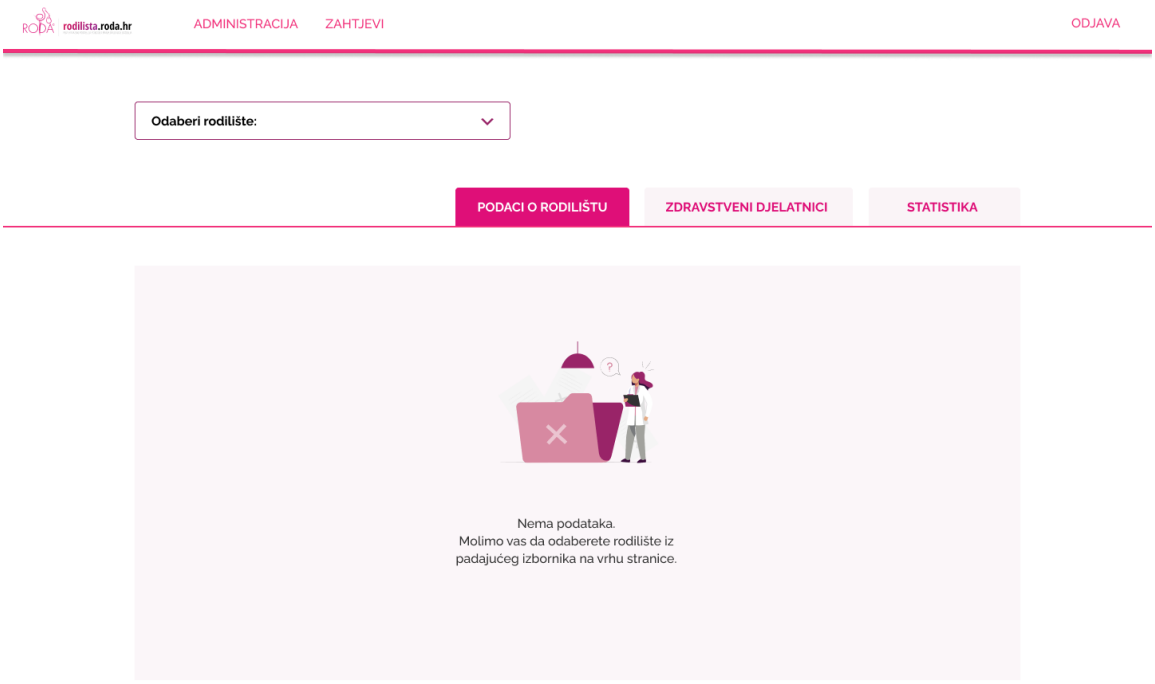
Na temelju definiranoga žičnog sustava i smjernica stilskoga vodiča napravljen je konačan dizajn aplikacije.



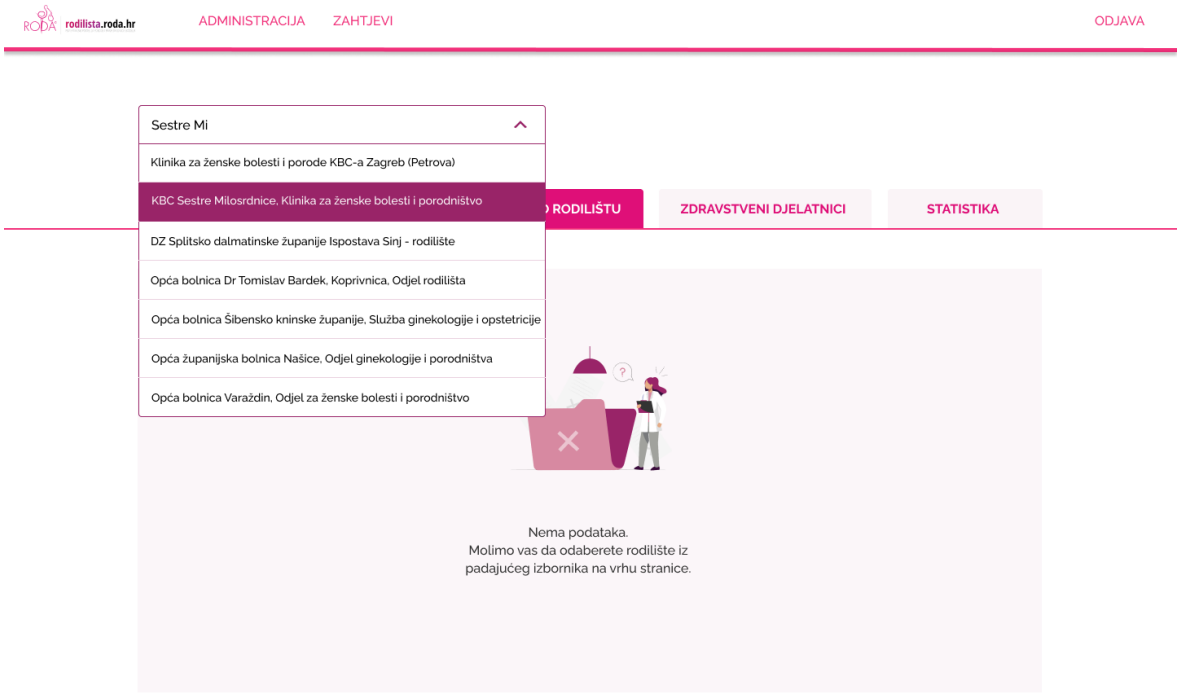
Slika 3.18 Dizajn stranice za prijavu korisnika



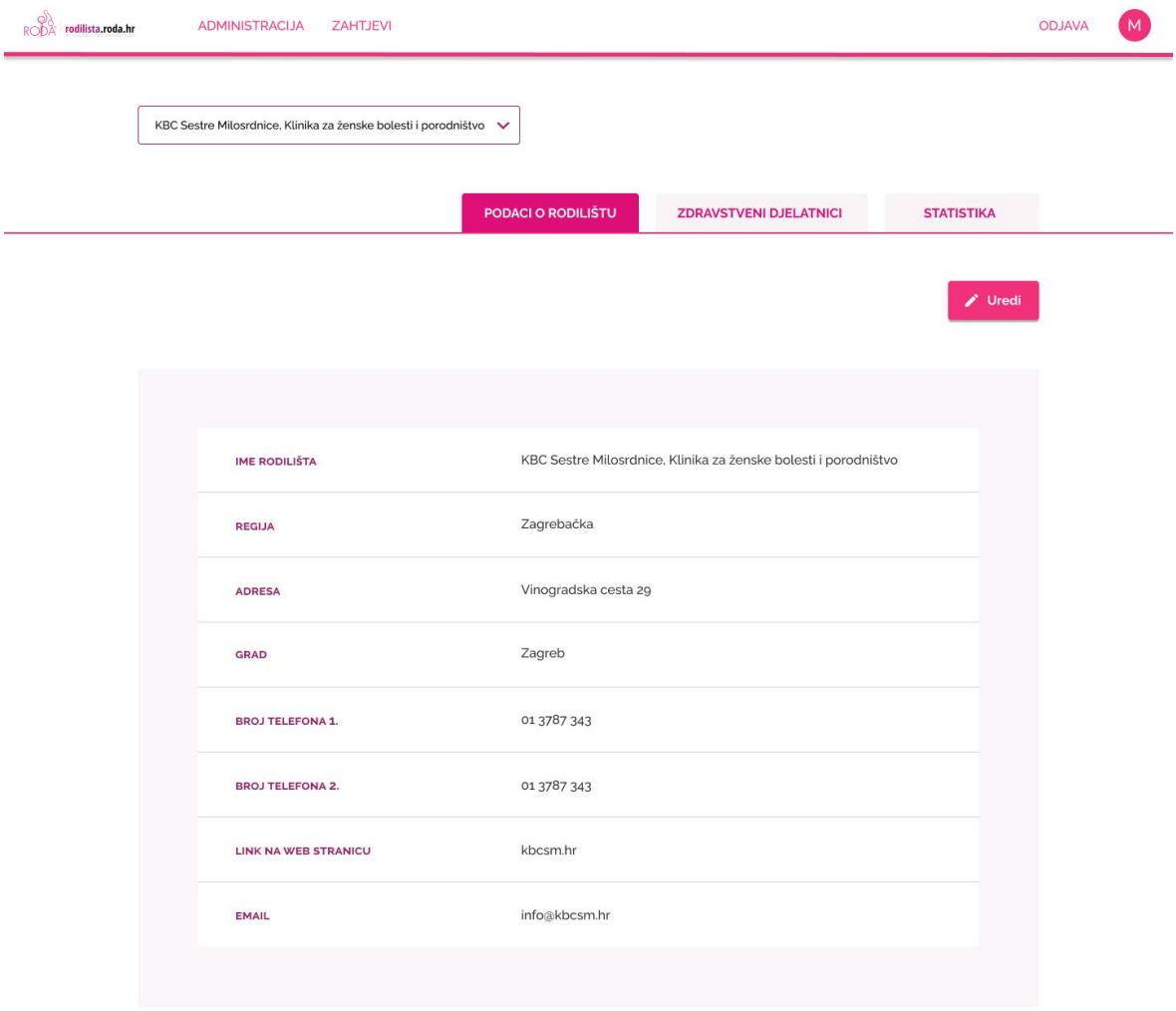
Slika 3.19 Dizajn stranice ako je istekla poveznica za promjenu lozinke



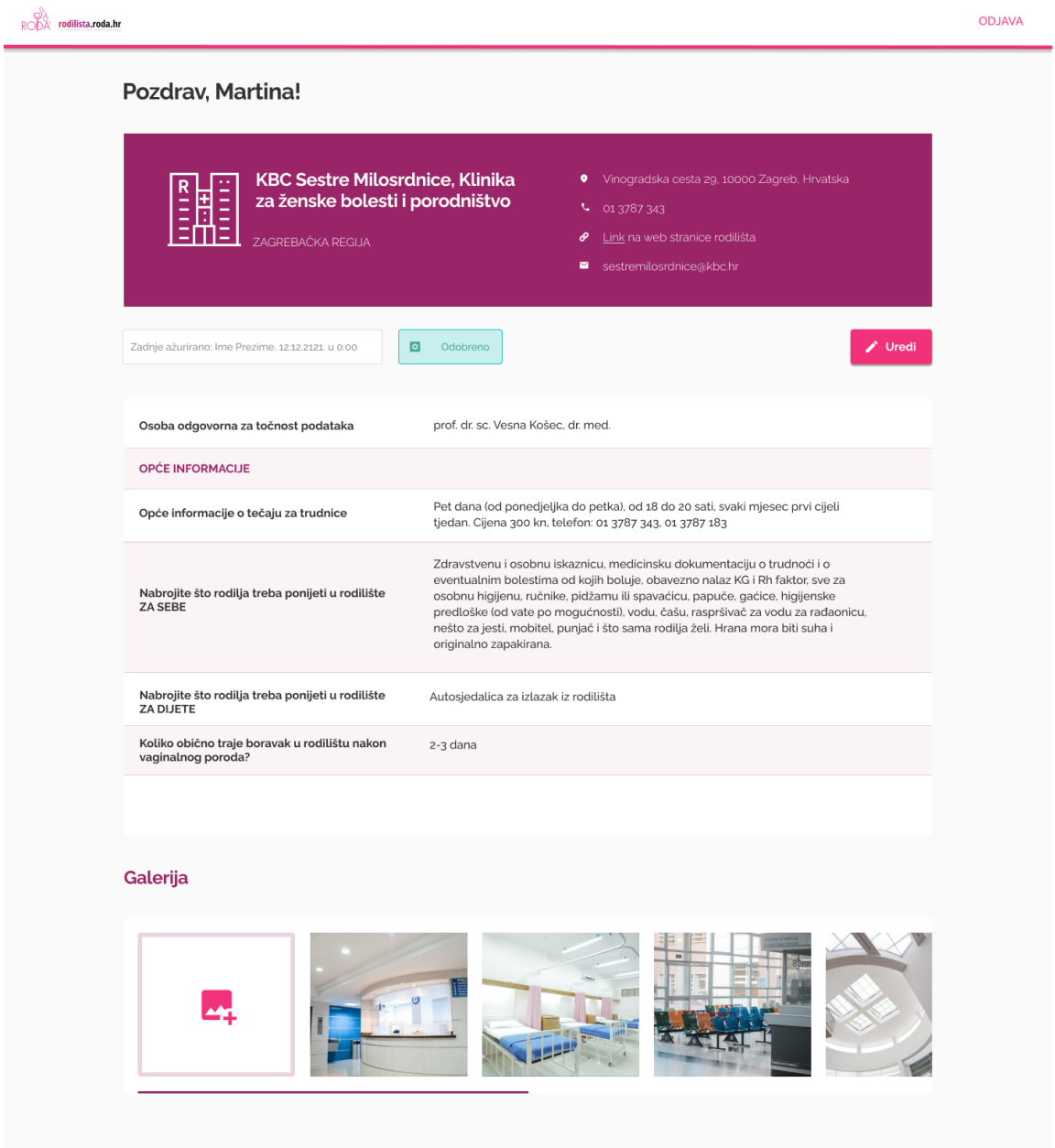
Slika 3.20 Dizajn početne stranice administratora udruge RODA



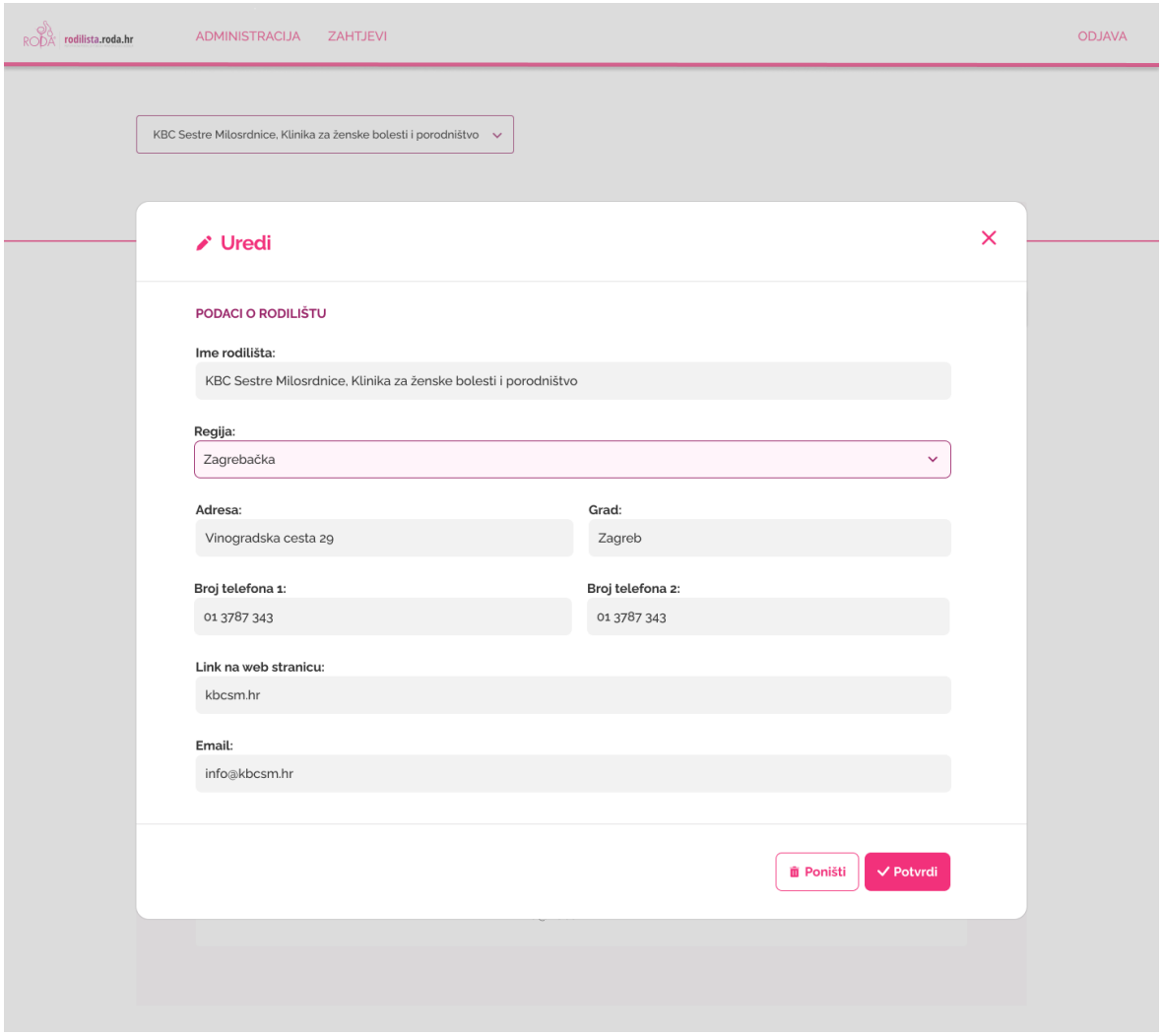
Slika 3.21 Dizajn početne stranice administratora udruge RODA



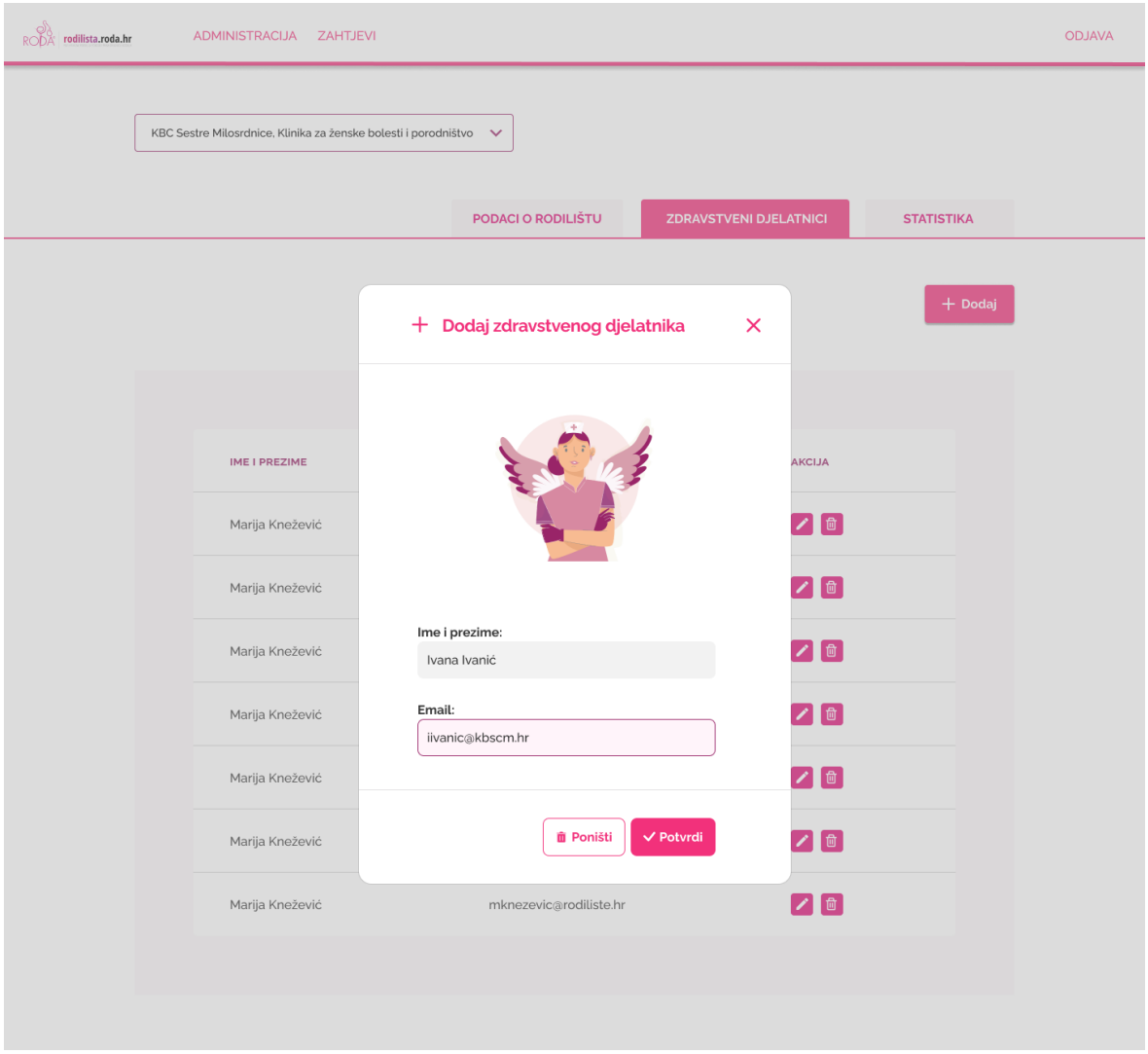
Slika 3.22 Dizajn početne stranice administratora udruge RODA s odabranim rodilištem



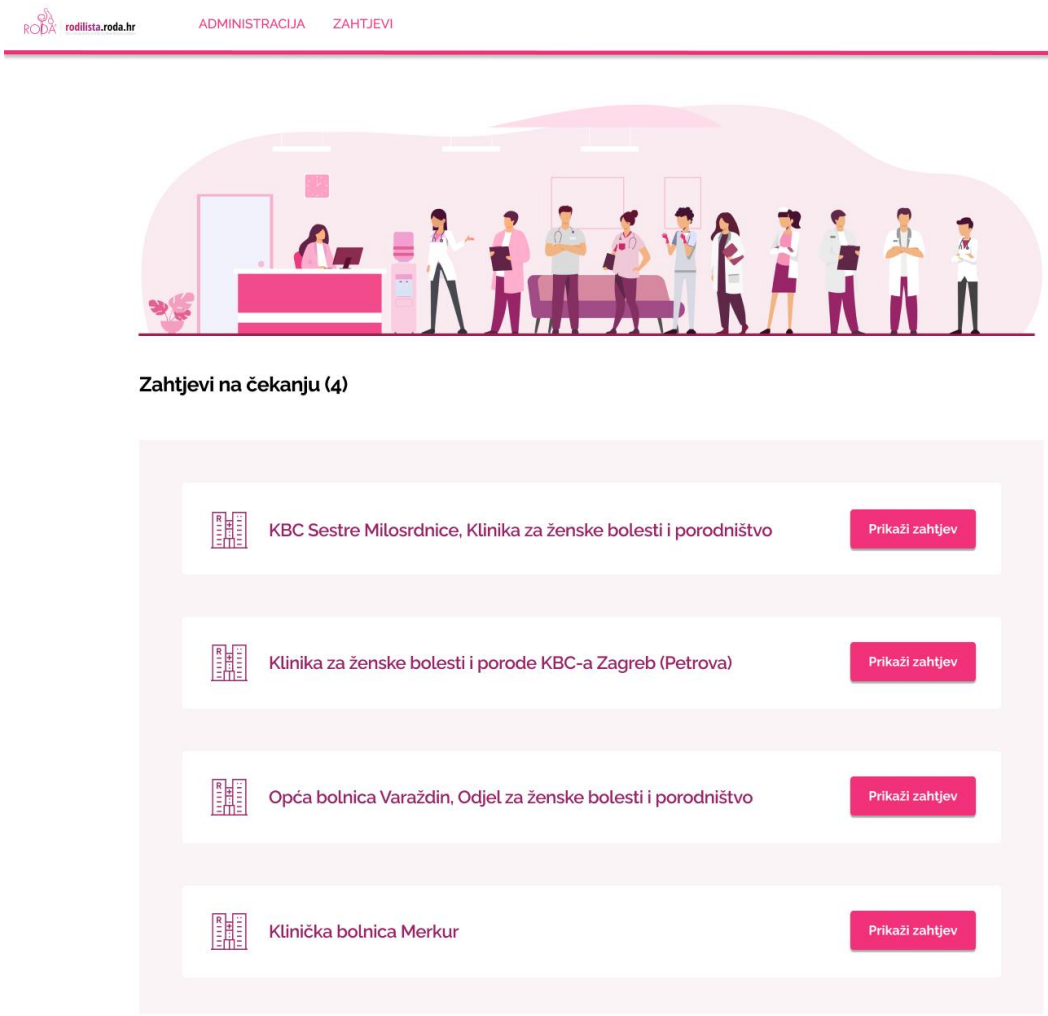
Slika 3.23 Dizajn početne stranice medicinskoga tehničara



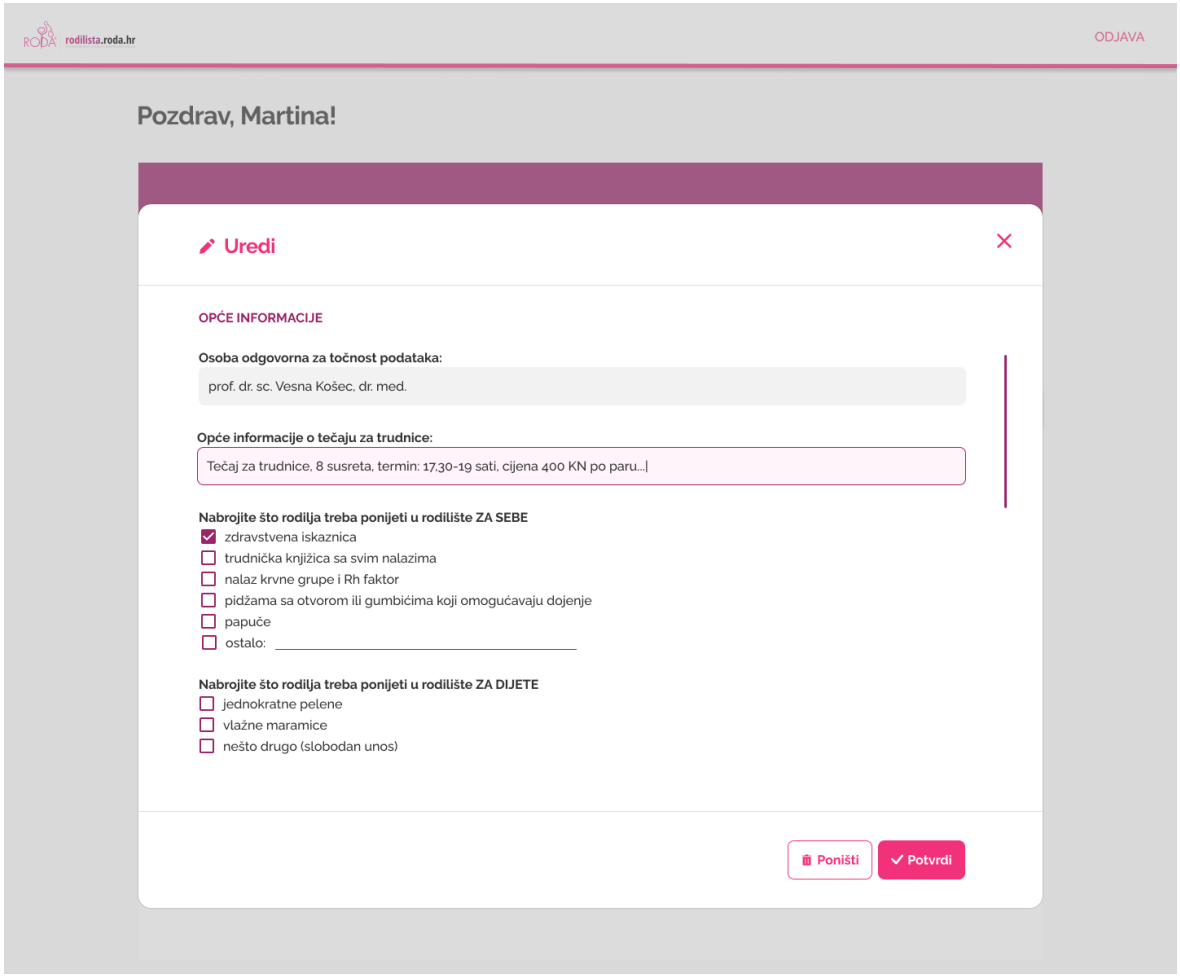
Slika 3.24 Dizajn početne stranice administratora udruge RODA s otvorenim skočnim prozorom



Slika 3.25 Dizajn početne stranice administratora udruge RODA s otvorenim skočnim prozorom

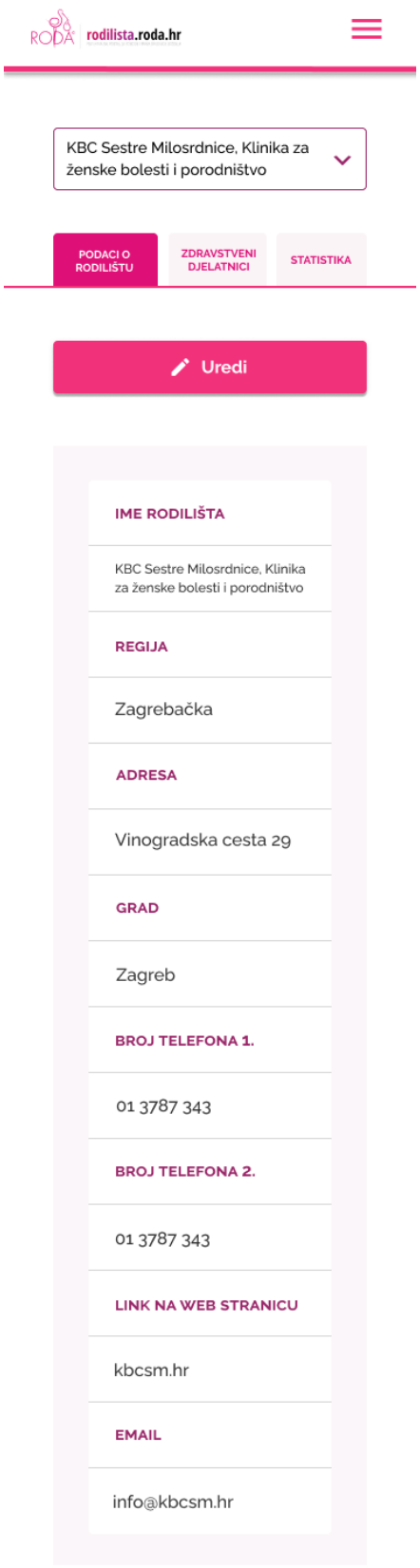


Slika 3.26 Dizajn stranice *Zahtjevi*



Slika 3.27 Dizajn početne stranice medicinskoga tehničara s otvorenom formom za popunjavanje ankete

Slika 3.28 Dizajn početne stranice medicinskoga tehničara za mobilne uređaje



Slika 3.29 Dizajn početne stranice administratora udruge s odabranim rodilištem za mobilne uređaje

4. Izrada *web* aplikacije

Web aplikacija interaktivna je aplikacija izrađena korištenjem tehnologija za *web* razvoj i kojoj korisnici mogu pristupiti iz *web* preglednika. Za razvoj *web* aplikacije koriste se različite *front-end* i *back-end* tehnologije. *Web* aplikacije usko su povezane s *web* stranicama, stoga razvoj *web* aplikacija i *web* stranica dijele mnoge karakteristike. Aplikacija opisana u ovome u radu ujedno je *single-page* aplikacija (skraćeno SPA), što znači da učitava samo jedan hipertekstualni dokument, a zatim ga ažurira pomoću JavaScript API-ja prilikom prikazivanja različitoga sadržaja. Drugim riječima, SPA je *web* aplikacija koja stvara iluziju više stranica, iako je korisnik cijelo vrijeme samo na jednoj stranici. Rezultat navedenoga povećanje je performansi i dinamičnije korisničko iskustvo, uz nedostatke kao što je optimizacija *web* stranice za tražilice (engl. *Search engine optimization*, skraćeno SEO).

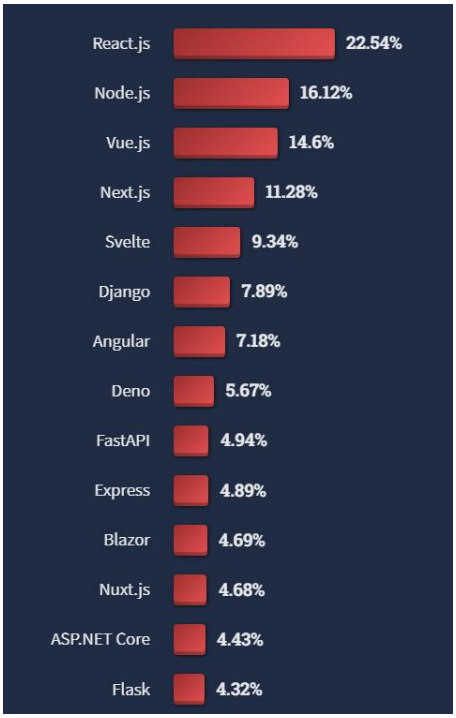
4.1. Odabrane tehnologije

Budući da ovaj rad opisuje *front-end* razvoj *web* aplikacije, u ovome dijelu bit će obrazložene korištene tehnologije, razlog njihova odabira, a implementacija je objašnjena u poglavlju 4.2.

Ovakav tip aplikacije ne može postojati bez baze podataka. Za ovaj završni rad odabrana je relacijska baza MySQL. Postavlja se pitanje: zašto MySQL? Većina *web* programera favorizira uporabu MySQL-a zbog jednostavne primjene, brzine, fleksibilnosti, skalabilnosti, pouzdanosti, lakoga postavljanja i održavanja za razliku od Mongo DB-a. Posljednji, ali ne manje važan razlog odabira MySQL-a je taj što gotovo svaki programer bez iskustva može vrlo jednostavno steći izvrsnost u MySQL-u.

Svakodnevno se objavljuju novi alati za *web* razvoj pa je time postalo teže napraviti pravi izbor. Postoji jedan popularan *front-end* razvojni okvir koji ruši rekorde u prostoru *web* razvoja, a zove se React JS. React JS u osnovi je JavaScript biblioteka, koju je izgradio i koju održava Facebook. React je visokokvalitetna, učinkovita i fleksibilna JavaScript biblioteka otvorenoga koda za izgradnju jednostavnih, brzih i skalabilnih sučelja *web* aplikacija. Najnovije istraživanje, za vrijeme pisanja ovoga rada, popularne stranice *StackOverflow* otkrilo je kako je React najomiljeniji razvojni okvir već petu godinu zaredom

(Slika 4.1 Rezultati istraživanja najtraženijih *web-frameworka* 2022. godine).[3] Divovi poput *Applea*, *Netflixa* i *Paypala* koriste se Reactom u svojim softverskim produkcijama. React nudi mnogo prednosti kao što su brzina razvojnoga procesa, lako održavanje, fleksibilnost koja štedi tvrtkama vrijeme i novac, visoke performanse, jednostavna implementacija ako programer ima osnovno znanje JavaScripta, višekratna uporaba komponenti te snažna podrška zajednice.



Slika 4.1 Rezultati istraživanja najtraženijih *web-frameworka* 2022. godine

Od najranijih dana svjetske mreže (engl. *World Wide Web*, skraćeno WWW) JavaScript se koristio za razvoj interaktivnosti *web* sjedišta poput rukovanja s različitim događajima prilikom operacija mišem ili tipkovnicom, provjere valjanosti podataka itd. Danas se JavaScript koristi na različitim platformama te ju to čini odličnim odabirom za razvoj *front-enda*, kao i *back-enda*. Za manje projekte opravdano je korištenje JavaScripta, međutim kod većih projekata javlja se problem ispravljanja (engl. *debug*) i hvatanja pogreški koda. Iz te potrebe razvijen je Typescript, nadskup JavaScripta. TypeScript pruža sigurnost tipa podatka, a prevodi se u JavaScript, koji preglednik razumije. Osim sigurnosti tipa, podržani su koncepti objektno orijentiranoga programiranja poput klasa i nasljeđivanja.

[3] https://survey.stackoverflow.co/2022/

Jedna od važnih odluka koja se donosi na početku izrade projekta je korištenje, odnosno nekorištenje bibliotekom komponenti. Potrebno je odrediti prednosti i mane korištenja bibliotekom koje ovise o opsegu i prioritetima projekta. Jedna od najpopularnijih biblioteka komponenti je Material UI (skraćeno MUI). MUI je sveobuhvatna React UI biblioteka po uzoru na *Googleov Material Design*. Dizajneri korisničkoga sučelja često koriste UI setove za bržu izradu ekrana. Biblioteke poput MUI-a dizajnerima omogućuju jednostavno povlačenje i ispuštanje (engl. *drag & drop*) komponenti kako bi efikasno dizajnirali sučelje. Dizajnerski timovi mogu potrošiti više vremena na dizajniranje sjajnih korisničkih iskustava, umjesto da troše vrijeme na kreiranje i testiranje komponenti iz nule.

Važan alat za *web* razvoj upravitelj je paketa koji pomaže upravljanju ovisnostima (engl. *dependencies*) projekta. Takvi alati pružaju metode za neprimjetno instaliranje, deinstaliranje i ažuriranje paketa. Za aplikaciju opisanu u ovome završnom radu odabran je Yarn upravitelj paketa. Yarn omogućuje korištenje i dijeljenje koda s programerima širom svijeta, što drastično olakšava izradu softvera dopuštajući korištenje rješenja drugih programera za određene probleme. Operacije s paketima rade se pomoću uputa naredbenoga retka (engl. *command-line instructions*). Prilikom dodavanja, primjerice, MUI biblioteke Yarn preuzima potrebni kod iz repozitorija, dodaje ga u projekt, kao i ostale potrebne pakete potrebne za ispravan rad MUI-a.

Izrada obrazaca u Reactu može zahtijevati značajnu količinu koda. Zbog toga su programeri u potrazi za alatima koji olakšavaju taj posao. Jedan od takvih alata je Formik, besplatna i lagana biblioteka otvorenoga koda. Prednosti koje Formik nudi su fleksibilnost, smanjena latencija u odnosu na *Redux Form*, skalabilnost, integracija s Yup bibliotekom za lakše rješavanje validacijskih procesa itd. Formik osim navedenoga značajno pomaže u upravljanju stanjem.

4.2. Implementacija pojedinih tehnologija

Zahvaljujući najnovijim dizajnerskim i tehnološkim trendovima usmjerenim na *front-end* razvoj, moguće je razviti vrlo sofisticirane dizajne i interakcije s korisničkim sučeljem. Međutim, istovremeno dolazi do veće složenosti procesa, zbog koje je *front-end* postao specijalizirano područje koje zahtijeva duboku stručnost. Budući da moderne *web* aplikacije traže fleksibilnost i skalabilnost, implementacija *web* aplikacija postaje ključna u ispunjavanju željenih i postavljenih ciljeva.

Nakon korisnikova uspješnoga odabira tehnologija i odobrenoga dizajna sljedeći je korak implementacija odabranih tehnologija opisanih u ovome potpoglavlju.

4.2.1. Baza podataka

Programski jezici stalno dolaze i odlaze te vrlo malo jezika koji se danas koriste imaju korijene koji sežu više od desetljeća unatrag. Jedan je od primjera *Cobol* programski jezik, koji se još uvijek dosta koristi kod glavnih računala (engl. *mainframe*). Osim Cobola, programski jezik C još uvijek je popularan za razvoj operativnih sustava i za ugrađene sustave (engl. *embedded systems*). U mnoštvu baza podataka postoji i strukturni upitni jezik (engl. *Structured Query Language*, skraćeno SQL), čiji korijeni sežu u prošlost do 1970-ih[4] godina.

Baza je podataka strukturirani set podataka. Riječ je o tehnologiji koja je nastala s namjerom da se uklone slabosti tradicionalne „automatske obrade podataka" iz 60-ih i 70-ih godina 20. stoljeća. Ta tehnologija osigurala je veću produktivnost, kvalitetu i pouzdanost u razvoju aplikacija koje se svode na pohranjivanje i pretraživanje podataka na računalu[5]. Razlikuju se dva tipa baza podataka: relacijske i nerelacijske baze podataka. Relacijska je baza podataka ona koja čuva podatke u tablicama. Tablice se sastoje od stupaca i redaka, a sve veze među podatkovnim elementima slijede strogu logičku strukturu (engl. *schema*). Ona za relacijske baze podataka mora biti jasno definirana. Popularne relacijske baze su: SQLite, MySQL, PostgreSQL.

Nerelacijska baza podataka svaka je ona baza koja ne koristi tabličnu shemu redaka i stupaca. Njezin model pohrane podataka prilagođen je za vrstu podataka koja se pohranjuje. Primjer je nerelacijske baze MongoDB.

Aplikacija opisana u ovome završnom radu koristi MySQL relacijsku bazu podataka. MySQL je jedna od najpoznatijih tehnologija u modernome podatkovnom ekosustavu. Često je nazivana najpopularnijom bazom podataka, a zbog njezine široke uporabe bez obzira na industriju, jasno je da bi svatko tko ima doticaj s podatcima ili bilo kakvim informacijskim tehnologijama trebao težiti osnovnomu poznavanju MySQL-a. Razvila ju je švedska tvrtka MySQL AB koju su osnovali David Axmark, Allan Larsson i Michael Widenius. Trenutačno

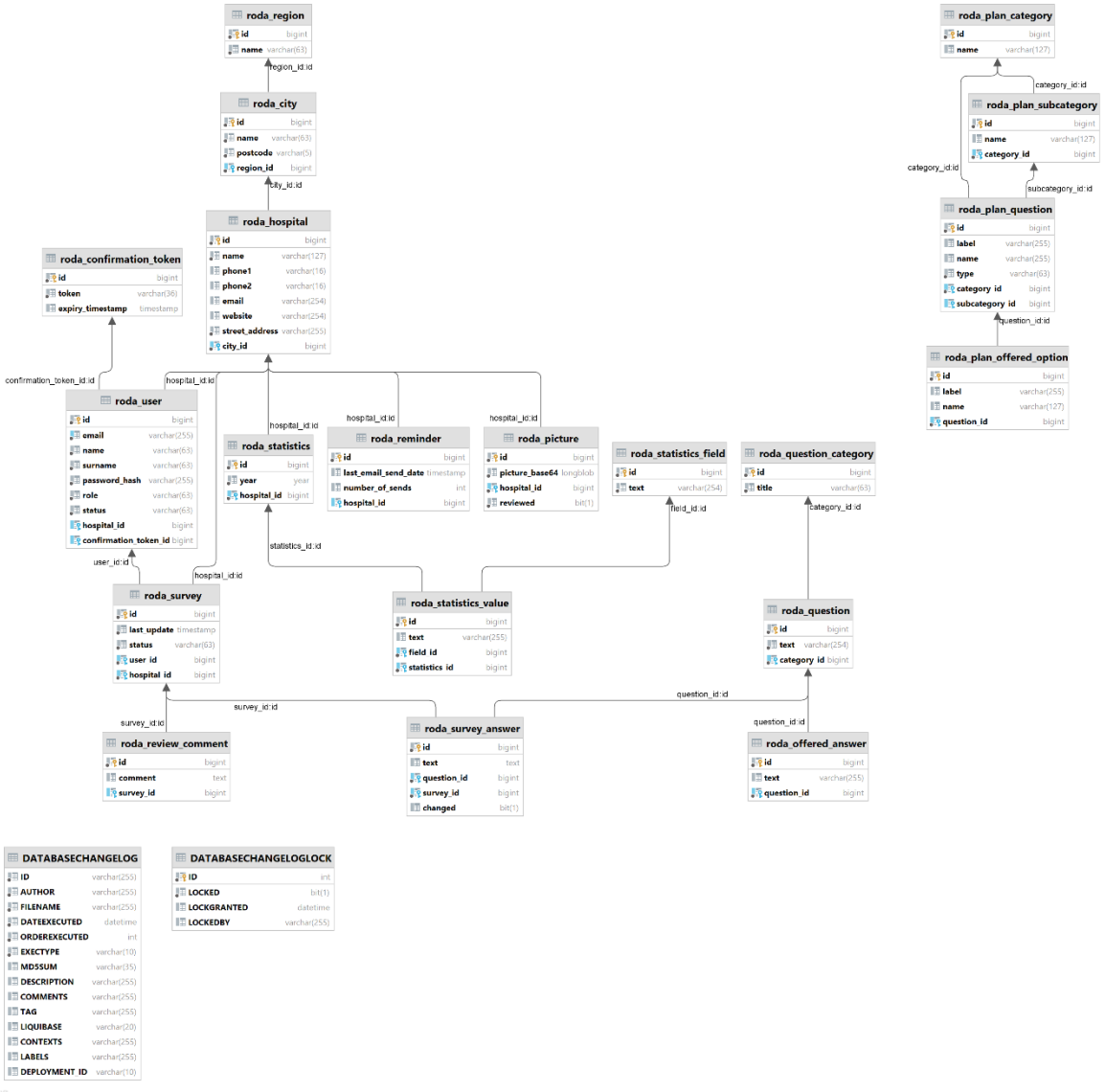
[4] Learning SQL: Generate, Manipulate, and Retrieve Data, stranica 17.
[5] Uvod u baze podataka, stranica 4.

ju razvija Oracle i temelji se na SQL-u. MySQL je sastavni dio mnogih popularnih softverskih alata za izgradnju i održavanje, počevši od *web* aplikacija do moćnih produktnih servisa. Njezina priroda otvorenoga koda, stabilnost i bogat skup značajki, upareni s podrškom Oraclea doveli su do toga da tvrtke poput Facebooka, *Twittera* i *Youtubea* koriste MySQL prilikom *back-end* razvoja. MySQL radi na svim glavnim računalnim platformama uključujući operativne sustave bazirane na *Unixu*, kao što su bezbrojne *Linux* distribucije, MAC OS ili Windows. MySQL-ov odnos klijent-poslužitelj znači da može podržati niz *back-end* tehnologija.

Primarni faktor koji razlikuje relacijske baze podataka od drugih digitalnih pohrana leži u organizaciji podataka na visokoj razini. Baze podataka poput MySQL-a sadrže zapise u višestrukim i odvojenim tablicama, za razliku od jednoga sveobuhvatnog repozitorija ili zbirke polustrukturiranih ili nestrukturiranih dokumenata. To sustavima upravljanja relacijskim bazama podataka omogućuje bolju optimizaciju radnji poput dohvaćanja i ažuriranja podataka, ali i složenijih radnji poput združivanja. Logički model definiran je nad svim sadržajima baze podataka, opisujući, naprimjer, vrijednosti dopuštene u pojedinačnim stupcima, karakteristike tablica ili kako su indeksi iz dviju tablica međusobno povezani.

MySQL čini mnoge ustupke podržavanju najšire moguće raznolikosti struktura podataka, od standardnih, ali i bogatih logičkih, numeričkih, alfanumeričkih, tipova datuma i vremena, do naprednijih JSON ili geoprostornih podataka. Osim navedenih vrsta podataka i ekspanzivnoga skupa ugrađenih značajki, MySQL ekosustav uključuje niz alata olakšavajući procese od upravljanja poslužiteljem do izvješćivanja i analize podataka.

Svaki pojedinac ili svaka tvrtka može se slobodno koristiti Oracleovom MySQL bazom otvorenoga koda, mijenjati ju, objavljivati i proširivati. Softver je objavljen pod Općom javnom licencom (eng. *General Public License*, skraćeno GPL).



Slika 4.2 Relacijski model baze podataka za aplikaciju RODA

MySQL upit je bilo koja naredba koja se koristi za upravljanje podatcima iz tablice. MySQL se može koristiti za filtriranje i sortiranje podataka, spajanje tablica, grupiranje i modificiranje podataka.

Izvršenjem upita iz primjera (Kôd 4.1) prikazuje se ispis svih rodilišta u Republici Hrvatskoj, uključujući njihove osnovne informacije poput broja telefona, *e-mail* adrese i *web* sjedišta.

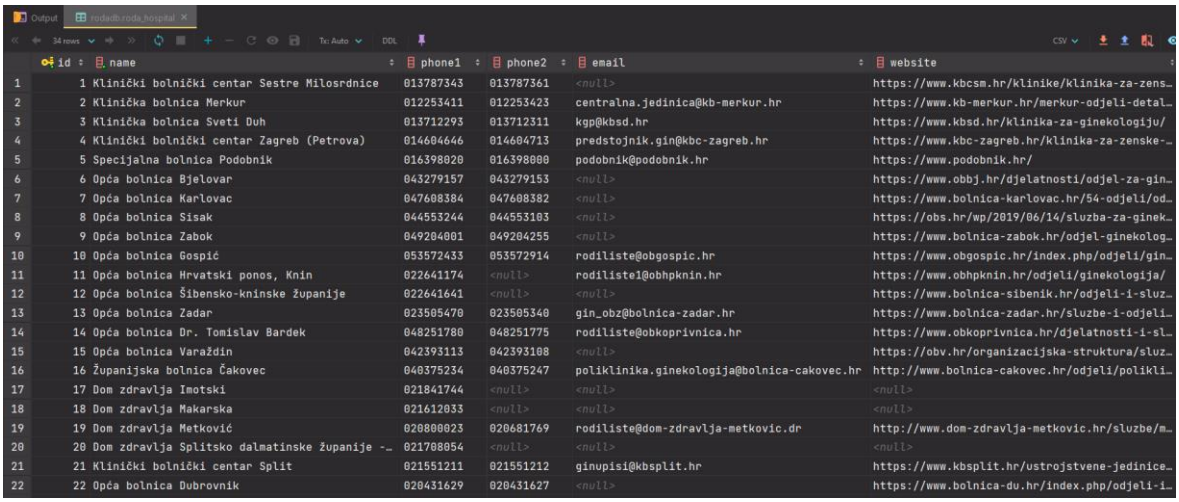
Moguće je izvršiti i složenije upite poput primjera (Kôd 4.2) u kojem se filtriraju podatci imena, prezimena, *e-maila* i uloge korisnika aplikacije te se prikazuju abecednim redom po prezimenu. Ispis trenutačno vraća dva retka s obzirom na to da su u tablici prisutna samo dva korisnika.

```
SELECT * FROM roda_hospital;
```

Kôd 4.1 Jednostavan primjer MySQL upita

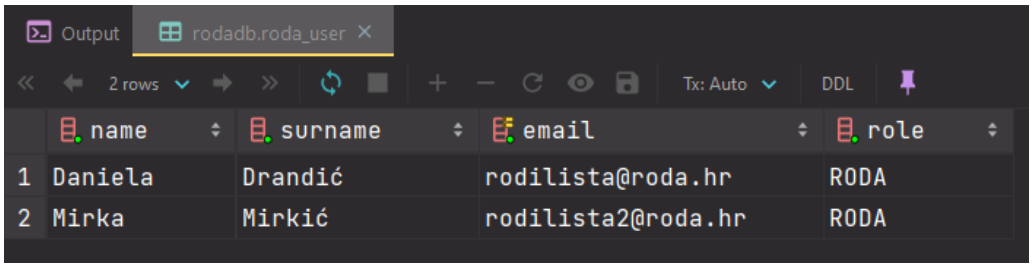
```
SELECT name, surname, email, role
FROM   roda_user
ORDER BY surname
```

Kôd 4.2 Složeniji primjer MySQL upita



	id	name	phone1	phone2	email	website
1	1	Klinički bolnički centar Sestre Milosrdnice	013787343	013787361	<null>	https://www.kbcsm.hr/klinike/klinika-za-zens…
2	2	Klinička bolnica Merkur	012253411	012253423	centralna.jedinica@kb-merkur.hr	https://www.kb-merkur.hr/merkur-odjeli-detal…
3	3	Klinička bolnica Sveti Duh	013712293	013712311	kgp@kbsd.hr	https://www.kbsd.hr/klinika-za-ginekologiju/
4	4	Klinički bolnički centar Zagreb (Petrova)	014604646	014604713	predstojnik.gin@kbc-zagreb.hr	https://www.kbc-zagreb.hr/klinika-za-zenske-…
5	5	Specijalna bolnica Podobnik	016398020	016398080	podobnik@podobnik.hr	https://www.podobnik.hr/
6	6	Opća bolnica Bjelovar	043279157	043279153	<null>	https://www.obbj.hr/djelatnosti/odjel-za-gin…
7	7	Opća bolnica Karlovac	047608384	047608382	<null>	https://www.bolnica-karlovac.hr/54-odjeli/od…
8	8	Opća bolnica Sisak	044553244	044553103	<null>	https://obs.hr/wp/2019/06/14/sluzba-za-ginek…
9	9	Opća bolnica Zabok	049204001	049204255	<null>	https://www.bolnica-zabok.hr/odjel-ginekolog…
10	10	Opća bolnica Gospić	053572433	053572914	rodiliste@obgospic.hr	https://www.obgospic.hr/index.php/odjeli/gin…
11	11	Opća bolnica Hrvatski ponos, Knin	022641174	<null>	rodiliste1@obhpknin.hr	https://www.obhpknin.hr/odjeli/ginekologija/
12	12	Opća bolnica Šibensko-kninske županije	022641641	<null>	<null>	https://www.bolnica-sibenik.hr/odjeli-i-sluz…
13	13	Opća bolnica Zadar	023505470	023505340	gin_obz@bolnica-zadar.hr	https://www.bolnica-zadar.hr/sluzbe-i-odjeli…
14	14	Opća bolnica Dr. Tomislav Bardek	048251780	048251775	rodiliste@obkoprivnica.hr	https://www.obkoprivnica.hr/djelatnosti-i-sl…
15	15	Opća bolnica Varaždin	042393113	042393108	<null>	https://obv.hr/organizacijska-struktura/sluz…
16	16	Županijska bolnica Čakovec	040375234	040375247	poliklinika.ginekologija@bolnica-cakovec.hr	http://www.bolnica-cakovec.hr/odjeli/polikli…
17	17	Dom zdravlja Imotski	021841744	<null>	<null>	<null>
18	18	Dom zdravlja Makarska	021612033	<null>	<null>	<null>
19	19	Dom zdravlja Metković	020800023	020681769	rodiliste@dom-zdravlja-metkovic.dr	http://www.dom-zdravlja-metkovic.hr/sluzbe/m…
20	20	Dom zdravlja Splitsko dalmatinske županije -…	021708054	<null>	<null>	<null>
21	21	Klinički bolnički centar Split	021551211	021551212	ginupisi@kbsplit.hr	https://www.kbsplit.hr/ustrojstvene-jedinice…
22	22	Opća bolnica Dubrovnik	020431629	020431627	<null>	https://www.bolnica-du.hr/index.php/odjeli-i…

Slika 4.3 Rezultat izvršenja SELECT upita na tablici bolnica



Output | rodadb.roda_user

2 rows | Tx: Auto | DDL

	name	surname	email	role
1	Daniela	Drandić	rodilista@roda.hr	RODA
2	Mirka	Mirkić	rodilista2@roda.hr	RODA

Slika 4.4 Rezultat izvršenja složenijega upita na tablici korisnika

4.2.2. Biblioteka React

React je JavaScript biblioteka, koja se koristi za izradu kvalitetnih korisničkih sučelja. Stvorio ju je Facebook 2011. godine za vlastitu uporabu, a 2013. učinio ga je tehnologijom otvorenoga tipa (engl. *open-source*). React funkcionira tako da traži od programera da koristi komponente koje rade odvojeno jedna od druge. React komponenta može biti bilo što u *web* aplikaciji poput gumba, teksta ili oznake polja za unos teksta. Svaka komponenta upravlja vlastitim stanjem, a zatim se zajedno sastave u složenome korisničkom sučelju. React je fleksibilan i koristi se na velikome broju platformi i za različite potrebe. Stvoren je s jednim fokusom – stvaranja komponenata za *web* aplikacije.

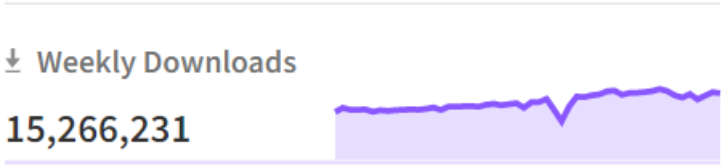
Primjer koda (Kôd 4.3 Primjer funkcijske React komponente) funkcijska je komponenta koja vraća JavaScript XML (skraćeno JSX). JSX je posebna sintaksa koja izgleda kao HTML, pretvara Reactove API pozive i konačno generira HTML.

```
import React from "react";

export default function Hello() {
  return <div>Hello World!</div>;
}
```

Kôd 4.3 Primjer funkcijske React komponente

Od 2015. godine Reactova popularnost i uporaba su iz dana u dan u usponu. Samim time okuplja veliku aktivnu zajednicu i GitHub repozitorij s preko 189 tisuća zvjezdica. Reactov NPM paket također ima milijune tjednih preuzimanja (Slika 4.5).



Slika 4.5 Broj tjednih preuzimanja u svibnju 2022.

Prilikom razvoja Reacta razvojni tim shvatio je da je JavaScript brz, ali ažuriranje *Document Object Modela* (skraćeno DOM-a) čini ga sporim. Stoga React minimizira DOM promjene na sljedeći način: React prati vrijednosti stanja svake komponente pomoću *Virtual* DOM-a. Kada se stanje komponente promijeni, React uspoređuje postojeće stanje DOM-a s onim kako bi DOM trebao izgledati, tj. novim stanjem. Nakon toga pronalazi najučinkovitiji način ažuriranja DOM-a. Ovaj proces React vrlo pametno rješava „ispod haube". Budući da se većina *web* aplikacija koristi i na mobilnim uređajima, one moraju biti brze i učinkovite, što dovodi do zaključka da su štednja baterije i snaga procesora od ključnoga značaja. Reactov jednostavan model programiranja omogućuje automatsku promjenu stanja komponenti prilikom ažuriranja podataka. Taj je proces brz jer se događa u memoriji. Veličina Reactove biblioteke je malena, točnije 6kb, što je značajno manji broj kilobajta od konkurenata.

Strukturalno, komponente su poput JavaScript funkcija. Prihvaćaju određene argumente, tzv. *Propse*, i vraćaju React elemente, koji opisuju što bi se trebalo pojaviti na zaslonu. Funkcija (Kôd 4.3) je valjana React komponenta jer prihvaća *props* s podatcima i vraća React element. Takve komponente zovu se i funkcijske komponente.

```
type TitleProps = {
label: string;
};
export default function Title({ label }: TitleProps) {
const classes = useStyles();
return (
    <Typography variant="h4" color="secondary"
className={classes.title}>
      {label}</Typography>
  );}
```

Kôd 4.4 Primjer *Title* funkcijske React komponente

4.2.3. Typescript

Typescript je programski jezik temeljen na JavaScriptu. Dodaje dodatnu sintaksu JavaScriptu pružajući čvršću integraciju s uređivačem koda (engl. *code editor*) i rano otkrivanje grešaka. TypeScript kôd pretvara se u JavaScript, koji se izvodi svugdje gdje se izvodi i JavaScript: primjerice u pregledniku ili na *Node*u. TypeScript koristi tipove kako bi pružio dodatne mogućnosti bez dodatnoga koda. Jednako tako čini kôd lakšim za razumijevanje i čitanje, daje sve prednosti *ECMAScript 6* (skraćeno ES6) standarda, veću produktivnost i strukturiranost. Iz spomenutih razloga često se koristi prilikom izrade kompleksnih projekata.

```
function addNumbers(a: number, b: number) {
    return a + b;
}
var sum: number = addNumbers(10, 15)

console.log('Sum of the two numbers is: ' +sum);
```

Kôd 4.4 Primjer generičkoga TypeScript kôda

```
export default function RodaRequestsList() {
  const classes = useStyles();
  const [requests, setRequests] =
useState<ReviewRequestItem[]>([]);
  const [selectedRequest, setSelectedRequest] =
useState<number>(0);
  const [selectedRequestName, setSelectedRequestName] =
useState<string>("");
```

```
  useEffect(() => {
    async function fetchData() {
      const response = await getRequests();
      setRequests(response.data.requests);
    }
    fetchData();
  }, [selectedRequest]);

  function handleOpenRequest(hospitalId: number,
hospitalName: string) {
    setSelectedRequest(hospitalId);
    setSelectedRequestName(hospitalName);
  }
```

Kôd 4.5 Primjer TypeScript kôda korištenoga za dohvat zahtjeva rodilišta

Osnovni tipovi varijabla mogu biti broj (engl. *number*), tekst (engl. *string*), *boolean*, *any* itd. Kod na primjeru prikazuje korištenje osnovnih tipova. Tip *SurveyAnswer* koristi se u komponenti ankete za rodilišta za tipizaciju odgovora dobivenih anketom, a tip *User* u formi za kreiranje novoga korisnika služi tipizaciji podataka o novome korisniku.

```
    export type SurveyAnswer = {
  questionId: number;
  questionText: string;
  answerText: string;
  changed: boolean;
};

export type User = {
  id?: number;
  email: string;
  name: string;
  surname: string;
  hospitalId?: number;
  isActive?: boolean;
};
```

Kôd 4.6 Primjer TypeScript kôda korištenoga za dohvat zahtjeva rodilišta

4.2.4. Material UI

Material UI (skraćeno, MUI) knjižnica je otvorenoga koda koja sadrži React komponente. Baziran je na *Googleovu* materijalnome dizajnu (engl. *Material Design*) za pružanje visokokvalitetnoga digitalnog iskustva prilikom razvoja korisničkoga sučelja. *Material Design* usredotočen je na pružanje odvažnih i oštrih dizajna te gradi teksturu fokusirajući se na to kako komponente bacaju sjene i reflektiraju svjetlost.

Postoji nekoliko ključnih prednosti u razvoju s ovom knjižnicom:

- material UI ima detaljnu dokumentaciju koja olakšava snalaženje i korištenje biblioteke
- postoje redovita ažuriranja
- komponente su dosljedne dizajnom i tonovima boja, što pridonosi estetici *web* aplikacije.

Knjižnicu je moguće instalirati pomoću Yarn upravitelja paketa jednom linijom koda (Kôd 4.6). Verzija knjižnice kojom se koristilo u projektu je 4.12.2.

```
yarn add @material-ui/core
```

Kôd 4.6 Instalacija MUI knjižnice pomoću *Yarna*

```
<Grid item lg={3} md={6} sm={6} xs={12}>
    <Card className={card.id == 3 ? classes.pink : ""}>
            <img src={card.image} alt="plan poroda" style={{
width: "4rem" }} />
<CardContent>
<Typography gutterBottom variant="body2"
component="h2">{card.title}</Typography>
<Typography variant="body1">{card.description}</Typography>
</CardContent>
<CardActions>
  <a className={classes.link} href={card.link}
target="_blank">
<Button className={card.id == 3 ? classes.whiteButton : ""}>
Više</Button></a>
</CardActions>
</Card>
</Grid>
```

Kôd 4.7 Primjer jednostavnoga korištenja gotovih Material UI komponenti

Material UI jednostavno se prilagođava unaprijed definiranoj paleti boja i odabiru pisma. Tema određuje boju komponenti, stilove pisma za određene vrste naslova, razinu sjene, prozirnost elemenata itd. Tema omogućuje prilagođavanje svih aspekata dizajna projekta kako bi zadovoljili određene potrebe korisnika. Ako je potrebno prilagođavanje teme, nužno je korištenje komponente `ThemeProvider`. Komponenta ThemeProvider oslanja se na Reactov kontekst kako bi proslijedila temu do svih komponenti te ju je zbog toga svojstva potrebno definirati na vrhu stabla, odnosno mora se ponašati kao roditelj ostatku komponenti.

```
function App() {
  return ( <div>
      <ThemeProvider theme={theme}>
        <Router basename="rode">
          <Routes />
        </Router>
      </ThemeProvider>
    </div> ); }
```

Kôd 4.8 Korištenje ThemeProvider komponente

Nakon izrade vlastita objekta teme (detaljnije objašnjeno u sljedećem odlomku) ona se prosljeđuje kao *prop ThemeProvider* komponenti.

Promjena konfiguracijskih varijabli teme najučinkovitiji je način za usklađivanje Material UI-a vlastitim potrebama. Najvažnije konfiguracijske varijable su: paleta boja, tipografija, razmak, prijelomne točke, z-indeks i globali. Paleta boja omogućuje promjenu boje komponenti kako bi odgovarale bojama *branda*. Zasebno se mogu definirati primarna i sekundarna boja (engl. *primary*, *secondary*) te statusne boje za greške, upozorenje, informacije i uspješne akcije (engl. *error*, *warning*, *info*, *success*). Za stvaranje prilagođene teme MUI izlaže *createTheme* metodu, koja prihvaća prilagođeni objekt teme kao argument.

```
const theme = createTheme({
palette: {
primary: {
      main: "#F2137B",
      contrastText: "#fff",
    },
secondary: {
main: "#992468",
      contrastText: "#fff",
```

```
    light: "#FAF4F7",
    dark: "#EBD3E1",
  },
  info: {
    main: "#F5D9E9",
  },
  error: {
    main: "#DA3030",
    contrastText: "#fff",
  },
  success: {
    main: "#307D2B",
    contrastText: "#fff",
  },
},
```

Kôd 4.9 Definiranje *brand* boja u temi

Z-index je CSS svojstvo koje pomaže kontrolirati izgled pružajući treću os za raspored sadržaja. Globali se odnose na poništavanje svih unaprijed definiranih stilova i koriste se u slučaju da preostale konfiguracijske varijable nisu dovoljno snažne. Na primjeru (Kôd 4.10) vidljiva je globalna konfiguracija komponente kartice (engl. *Card*), što znači da će svaka *Card* komponenta u projektu biti oblikovana prema ovdje definiranim stilovima.

```
 MuiCard: {
  root: {
    display: "flex",
    textAlign: "center",
    flexDirection: "column",
    alignItems: "center",
    justifyContent: "center",
    height: "22vw",
    padding: "0px 8px",
    boxShadow: "rgba(99, 99, 99, 0.2) 0px 2px 8px 0px",
},
```

Kôd 4.10 Stilska konfiguracija Card komponente

4.2.5. Yarn

Yarn je upravitelj paketa koji omogućuje korištenje kodom i njegovo dijeljenje među programerima cijeloga svijeta. Primarna je funkcija upravitelja paketa instalacija određena paketa iz globalnoga registra u lokalno okružje softver inženjera. Na taj način svaka aplikacija može koristiti već postojeće programsko rješenje za određeni interni problem kako bi se olakšao razvojni proces. Kôd se dijeli paketom. Paket sadrži sav kôd rješenja kao i *package.json* datoteku, koja opisuje paket. Jedan paket može, ali i ne mora, ovisiti o drugim paketima te ih jedan projekt može imati desetke, stotine ili čak tisuće.

Proces instalacije paketa odvija se na sljedeći način – Yarn počinje tražiti zavisnosti (engl. *dependencies*) upućivanjem zahtjeva prema registru i rekurzivnim prolaženjem kroz zavisnosti. Zatim u predmemoriji (engl. *cache*) traži informaciju globalnoga direktorija je li paket već preuzet. Ako nije, dohvaća paket i stavlja ga u globalni *cache* kako bi mogao raditi izvan mreže. Konačno, Yarn sve zajedno povezuje kopiranjem svih potrebnih datoteka iz globalnog *cachea* u lokalni *node_modules* direktorij.

```
$ yarn add formik
yarn add v1.22.18
warning package-lock.json found. Your project contains lock files generated by t
ools other than Yarn. It is advised not to mix package managers in order to avoi
d resolution inconsistencies caused by unsynchronized lock files. To clear this
warning, remove package-lock.json.
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
```

Slika 4.6 Dodavanje Formik paketa pomoću Yarna

Prilikom pokretanja *yarn* naredbe mogu se dogoditi dvije stvari:

1. ako već ne postoji, generira se *yarn.lock* datoteka prema sadržaju iz datoteke *package.json*
2. instalacija paketa prema sadržaju postojeće *yarn.lock* datoteke.

Kada je datoteka *yarn.lock* prisutna, ona je glavni izvor informacija o trenutačnim verzijama paketa korištenim u projektu. Prilikom nadogradnje, dodavanja ili uklanjanja paketa pomoću *yarn* naredbe datoteka se automatski ažurira te se ne preporuča njezino izravno uređivanje. Informacije iz *lock* datoteke korisne su ostatku programera kako bi kreirali isto okružje za daljnji razvoj aplikacije.

4.2.6. Create React App

Create React App najbolji je način za početak izrade nove *single-page* aplikacije u Reactu. Postavlja razvojno okružje za korištenje najnovijih mogućnosti JavaScripta, pruža odlično razvojno iskustvo i optimizira aplikaciju za produkciju. Preduvjet je korištenja ovoga alata instaliran Node s verzijom 14 ili višom i *npm* s verzijom 5.6 ili višom. Create React App ne upravlja *back-endom* ni bazama podataka, samo na neki način stvara prolaz (engl. *pipeline*) kako bi se koristio s različitim tehnologijama na *back-endu*. Iza kulisa ovoga alata rade *Babel* i *Webpack*.

Npm upravitelj je paketa koji se dobiva zajedno s instalacijom Nodea. Omogućava programerima instalaciju globalnih i lokalnih paketa. S druge strane, *npx* pokretač je paketa i unaprijed je povezan s *npmom*. *Npx* komandno je linijski alat (engl. *command-line interface*, skraćeno CLI), koji olakšava instalaciju i upravljanje paketima smještenim u *npm* registru. Create React App nudi instalaciju pomoću *npma* i *yarna*.

Naredba korištena za pokretanje React projekta je: `yarn create react-app rode-frontend --template typescript`. Dodatni parametar –template typescript omogućava jednostavno kreiranje *typescript* aplikacija pomoću *typescript* predloška.

4.2.7. useState i *useEffect*

React kuke (engl. *hooks*) funkcije su koje omogućavaju upravljanje stanjima (engl. *stateovima*) i značajkama životnoga ciklusa funkcijskih komponenti. Predstavljene su u Reactovoj verziji 16.8. Prije *hookova* nije bila moguća promjena stanja u funkcijskim komponentama te su se one morale pretvoriti u *class* komponente. Zahvaljujući *hookovima* taj se proces više ne mora raditi, a ni *hookovi* ne rade unutar *class* komponenti.

useState hook je *hook* koji je potrebno pozvati unutar funkcijske komponente kada se želi dodati lokalno stanje. *useState* vraća dva parametra, od kojih je prvi parametar vrijednost početnoga stanja, tj. početna vrijednost, a drugi predstavlja funkciju koja omogućuje ažuriranje prvoga parametra. *useState* može se koristiti više puta unutar jedne komponente.

```
  const [surveyData, setSurveyData] =
useState<HospitalSurvey>(blankSurvey);
  const [modalOpen, setModalOpen] = useState<boolean>(false);
  const [denyComment, setDenyComment] = useState<string>("");
```

Kôd 4.11 Primjer korištenja *useState hooka*

Upućivanje API poziva za dohvaćanje podataka, kao i promjena *propova*, može utjecati na promjenu stanja komponenti. U tim slučajevima najčešće se koristi *useEffect hook*. Istu svrhu u *class* komponentama imaju *componentDidMount*, *componentDidUpdate* i *componentWillUnmount* metode, ali su one u ovome *hooku* ujedinjene u jedan API. *useEffect hook* pokreće se pri svakome renderiranju i ponovnome generiranju, što znači da će se prilikom svake promjene stanja u komponenti ili prilikom primanja novih *propova*, komponenta ponovno generirati i uzrokovati pokretanje *useEffect hooka*. Kuke se, dakle, pokreću na svaki *redner* komponente, no postoji način osiguravanja samo jednoga pokretanja *useEffect hooka*. Naprimjer, ako komponenta dohvaća podatke s API-ja, pogrešno ponašanje bi bilo kada bi se oni dohvaćali na svaki ponovni *render* komponente. Stoga *useEffect* prima i drugi parametar, niz (engl. *array*) koji sadrži popis parametara koji će uzrokovati ponovno pokretanje *useEffecta*. Prilikom promjene spomenutih parametara pokreće se *useEffect*. Kako bi se omogućilo samo jednokratno pokretanje *useEffecta*, niz mora biti prazan, odnosno prosljeđujemo prazan niz kao drugi parametar *hooka*.

```
useEffect(() => {
  async function fetchData() {
    try {
      const response = await getHospitals();
      setHospitals(response.data.hospitalNameList);
    } catch (reason: any) {
      if (
        reason.response &&
        reason.response.status === errorCodes.NOT_FOUND
      ) {
        setError(true);
      }
    }
  }

  fetchData();
}, []);
```

Kôd 4.12 Primjer korištenja *useEffect hooka*

4.2.8. useContext

Kako bi se objasnio *useContext hook*, najprije je potrebno objasniti React Context API. Kontekst pruža način prosljeđivanja podataka kroz stablo komponenti bez ručnoga prosljeđivanja *propova* na svakoj pojedinoj razini. U tipičnoj React aplikaciji podatci se prosljeđuju odozgo prema dolje, odnosno *propsima* s roditelja na djecu. Taj način prosljeđivanja može biti zahtjevan za određene *propse*, npr. za temu ili lokalizaciju, koje zahtijevaju mnoštvo komponenata unutar aplikacije. Kontekst omogućuje dijeljenje vrijednosti poput tih između komponenata bez eksplicitnoga definiranja *propova* u svakoj razini stabla. Kontekst je dizajniran za dijeljenje podataka koji se smatraju globalnim za stablo React komponenti. Ti podatci mogu biti trenutačni autentificirani korisnik, tema ili preferirani jezik. Kontekst se prvenstveno koristi kada podatci moraju biti dostupni mnogim komponentama na različitim razinama.

`React.createContext` stvara objekt konteksta. Kada React renderira komponentu koja je pretplaćena na kontekst, ona će pročitati trenutačnu vrijednost konteksta iz najbližeg *context Providera* u stablu. Svaki kontekst dolazi s komponentom *Provider*, koja ostalim komponentama omogućuje praćenje promjena konteksta.

useContext hook radi na sličan način. Koristi se za stvaranje zajedničkih podataka kojima se može pristupiti kroz cijelu hijerarhiju komponenti bez ručnoga prosljeđivanja *propova*. Definirani kontekst bit će dostupan svim *child* komponentama bez uključivanja *propova*.

Primjer (Kôd 4.13 Primjer korištenja prikazuje kreiranje *HospitalsContexta* pomoću *createContext* React funkcije, koja vraća *context* kao objekt. S obzirom na to da je u prikazanoj komponenti *context* izvezen (engl. *export*), druge komponente mogu ga koristiti pozivom Reactove funkcije *useContext* i prosljeđivanjem parametra *HospitalsContext* (Kôd 4.14 Primjer korištenja *Contexta* u komponenti izvan mjesta definicije). Prilikom kreiranja *contexta*, postavljaju se zadane (engl. *default*) vrijednosti koje će se nalaziti u *contextu*. Sve komponente koje se nalaze unutar *HospitalsContext* Providera mogu pristupiti proslijeđenim vrijednostima *contexta*. Ako se proslijeđene vrijednosti promijene, React će ponovno učitati komponentu čitajući *context*.

```
export const HospitalsContext =
createContext<HospitalsContextType>({
  currentHospital: 0,
  setCurrentHospital: () => {},
```

```
  error: false,
  setError: () => {},
});

export default function RodaHomepage() {
  const classes = useStyles();
  const [currentHospital, setCurrentHospital] =
useState<number>(0);
  const [error, setError] = useState<boolean>(false);

  return (
    <>
      <MetaDecorator />
      <Navigation />
      <Box pb={8} pt={7}>
        <Container maxWidth="lg">
          <HospitalsContext.Provider
            value={{
              currentHospital,
              setCurrentHospital,
              error,
              setError,
            }}
          >
            <HospitalDropdownMenu />
            <RodaTabs />
          </HospitalsContext.Provider>
        </Container>
        <Typography
className={classes.bottom}>{appVersion}</Typography>
      </Box>
    </>
  );
}
```

Kôd 4.13 Primjer korištenja React Contexta

```
  const { setCurrentHospital, setError } =
useContext(HospitalsContext);
```

Kôd 4.14 Primjer korištenja *Contexta* u komponenti izvan mjesta definicije

4.2.9. React Router

React Router biblioteka je za usmjeravanje (engl. *routing*) u Reactu. Omogućuje navigaciju među prikazima različitih komponenti u *web* aplikaciji, omogućuje promjenu URL preglednika i sinkronizira UI s URL-om. Glavne komponente React Routera su:

- *BrowserRouter* – usmjerivač koji koristi API kako bi sinkronizirao korisničko sučelje s URL-om. Ujedno je i roditeljska komponenta u koju se pohranjuju ostale komponente.
- *Switch* – komponenta koja pregledava sve svoje podređene putanje i prikazuje prvu, čiji put odgovara trenutačnom URL-u
- Putanja – uvjetno prikazana komponenta koja generira korisničko sučelje kada se njezina putanja podudara s trenutačnim URL-om
- Link – komponenta koja se koristi za stvaranje poveznica na različite putanje i implementaciju navigacije. Radi kao HTML oznaka sidra.

Na početku *typescript* datoteke potrebno je napraviti uvoz (engl. *import*) komponenti iz paketa na način koji je prikazan u kodu (Kôd 4.15).

```
import {
  Redirect,
  Route,
  Switch,
  useHistory,
  useLocation,
} from "react-router-dom";
```

Kôd 4.15 Primjer uključivanja React Router komponenti

React Router radi djelomično podudaranje, primjerice „/rodiliste" djelomično odgovara putanji „/rodiliste/pocetna", što bi vratilo pogrešnu putanju. U takvim slučajevima koristi se *exact prop*, koji onemogućava djelomično podudaranje i vraća samo onu putanju koja se eksplicitno poklapa s trenutačnim URL-om (Kôd 4.16 Primjena React Router komponenti). Komponenta *Redirect* preusmjerit će na novu putanju. Nova putanja nadjačat će trenutačnu u povijesti, kao što to čine preusmjeravanja na strani poslužitelja (engl. *server-side redirects*). Funkcija `redirectToHomepage` prima jedan parametar, *user authority*,

odnosno pripadajuću ulogu korisnika koja se dohvaća iz *UserContexta*. U slučaju da sustav utvrdi prijavu korisnika udruge, preusmjerava ga na URL „roda-pocetna”, a ako se radi o medicinskome tehničaru, odnosno korisniku rodilišta, sustav ga preusmjerava na URL „rodiliste-pocetna”. Zaštićene su putanje (engl. *protected route*) one koje dopuštaju pristup samo ovlaštenim korisnicima, što doprinosi sigurnosti *web* aplikacije. To znači da korisnik mora ispuniti određene uvjete prije nego što pristupi određenoj putanji. *GuardedRoute* je komponenta napravljena u svrhu zaštite pojedinih putanja tako što je ugrađena dodatna logika koja provjerava ulogu korisnika i ako korisnik ima prava pristupiti sustavu, preusmjerava ga na pripadajući URL. To znači da korisnik koji nije član udruge, a može se prijaviti kao medicinski tehničar, ne može vidjeti dio aplikacije namijenjen isključivo članovima udruge.

useHistory hook daje pristup instanci povijesti s nekoliko funkcija, a sve u svrhu navigacije kroz aplikaciju.

```
          <Switch>
            <Route path={routes.LOGIN_URL}>
              <Redirect to={redirectToHomepage(user.authority)}
/>
            </Route>
            <Route exact path={"/"}>
              <Redirect to={redirectToHomepage(user.authority)}
/>
            </Route>
            <UserContext.Provider value={{ user, setUser }}>
              <IdleTimerLogout handleLogout={onLogout} />
              <GuardedRoute
                userRole={user.authority}
                requiredRole="NURSE"
                path={routes.NURSE_HOMEPAGE_URL}
                component={NurseHomepage}
              />
              <GuardedRoute
                userRole={user.authority}
                requiredRole="RODA"
                path={routes.RODA_HOMEPAGE_URL}
                component={RodaHomepage}
              />
              <GuardedRoute
```

```
                userRole={user.authority}
                requiredRole="RODA"
                path={routes.RODA_USER_ADMINISTRATION_URL}
                component={RodaUserAdministration}
              />
              <GuardedRoute
                userRole={user.authority}
                requiredRole="RODA"
                path={routes.RODA_REQUESTS_URL}
                component={RodaRequestsList}
              />
              <Route path={routes.PUBLIC_STATISTICS_URL}
exact={true}>
                <PublicStatistics />
              </Route>
              <Route path={routes.BIRTH_PLAN_URL} exact={true}>
                <BirthPlanHomepage />
              </Route>
              <Route path={routes.BIRTH_PLAN_FORM} exact={true}>
                <BirthPlanFormPage />
              </Route>
            </UserContext.Provider>
          </Switch>
```

Kôd 4.16 Primjena React Router komponenti

4.2.10. Formik

Formik je najpopularnija svjetska knjižnica otvorenoga kôda za React i React Native. Formik se brine za repetitivne i naporne stvari poput praćenja vrijednosti, pogrešaka, posjećenih polja, orkestriranje validacije i rukovanje slanjem forme. Samim time troši se manje vremena na povezivanje *stateova* s rukovateljima promjena (engl. *change handlers*) usredotočujući se na poslovnu logiku. Formik se ne koristi vanjskim bibliotekama kao što su *Redux* ili *MobX*, što ga čini jednostavnijim za korištenje i pridonosi smanjenoj veličini paketa.

Može se instalirati s *npmom* ili *yarnom* koristeći naredbu `yarn add formik`. Formik prati stanje obrasca forme, a zatim ga *propovima* izlaže formi zajedno s nekoliko metoda i *event handlera* (`handleChange, handleBlur i handleSubmit`). Kako bi *developerima* uštedio vrijeme, Formik dolazi s nekoliko dodatnih komponenti: `<Form />`,

<Field /> i <ErrorMessage />. One koriste React Context za povezivanje s Formikovim *stateovima* i metodama.

```
import React from 'react';
 import { Formik, Form, Field, ErrorMessage } from 'formik';

 const Basic = () => (
   <div>
     <h1>Any place in your app!</h1>
     <Formik
       initialValues={{ email: '', password: '' }}
       validate={values => {
         const errors = {};
         if (!values.email) {
           errors.email = 'Required';
         } else if (
           !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-
Z]{2,}$/i.test(values.email)
         ) {
           errors.email = 'Invalid email address';
         }
         return errors;
       }}
       onSubmit={(values, { setSubmitting }) => {
         setTimeout(() => {
           alert(JSON.stringify(values, null, 2));
           setSubmitting(false);
         }, 400);
       }}
     >
       {({ isSubmitting }) => (
         <Form>
           <Field type="email" name="email" />
           <ErrorMessage name="email" component="div" />
           <Field type="password" name="password" />
           <ErrorMessage name="password" component="div" />
           <button type="submit" disabled={isSubmitting}>
             Submit
           </button>
         </Form>
       )}
```

```
      </Formik>
    </div>
  );

  export default Basic;
```

Kôd 4.17 Primjena Formika na jednostavnoj formi za prijavu

4.2.11. Yup

U primjeru koda (Kôd 4.17Kôd 4.17 Primjena Formika na jednostavnoj formi za prijavu) koristi se posebna funkcija za validaciju i time se programeru prepušta pisanje vlastitih validatora. Međutim, uz Formik se najčešće veže biblioteka treće strane (engl. *third-party library*) imena Yup. Yup ima API koji je dovoljno malen za preglednike i brz tijekom izvođenja. Budući da se Formik i Yup odlično nadopunjuju, Formik sadrži posebnu konfiguracijsku opciju, tj. *prop* za sheme Yup objekata `validationSchema`. Yup se može instalirati s *npma* ili *yarna*.

```
const loginSchema = yup.object({
  email: yup
    .string()
    .required("Upišite e-mail.")
    .email("Unesite ispravan e-mail."),
  password: yup.string().required("Upišite lozinku."),
});
```

Kôd 4.18 Primjena Yup validacijske sheme

Prvi parametar predstavlja pravilo .string, .required ili .email. Drugi parametar prikazuje poruku koja se prikazuje u slučaju nevažećega polja. Validacijska shema u primjeru (Kôd 4.18 Primjena Yup validacijske sheme) provjerava je li tekst unesen u polje za unos *e-maila* tipa *string*, je li polje uopće ispunjeno i odgovara li standardnomu obliku *e-maila*. Jednako tako provjerava se postoji li upisana lozinka u polje za unos lozinke.

4.2.12. React Testing biblioteka

React Testing biblioteka predstavlja skup pomoćnih metoda koje omogućuju testiranje React komponenti bez oslanjanja na detalje njihove implementacije. Projekti kreirani s naredbom *create-react-app* već imaju podršku za React Testing knjižnicu te ne zahtijevaju nikakvu

konfiguraciju za korištenje. Ako to nije slučaj, jednostavno se dodaje pomoću naredbe (Kôd 4.19 Dodavanje React Testing Library paketa u projekt).

```
yarn add --dev @testing-library/react
```

Kôd 4.19 Dodavanje React Testing Library paketa u projekt

Ova knjižnica potiče aplikacije da budu pristupačnije i omogućuje približavanje testova korištenju komponenti na način na koji će to učiniti korisnik. Testovi samim time ulijevaju povjerenje razvojnom timu da će aplikacija raditi kada je koristi pravi korisnik.

Na primjeru (Kôd 4.20 Testiranje komponente gumba pomoću React Testing knjižnice) demonstrirana je primjena React Testing knjižnice na komponenti gumba. Prve dvije linije koda dodaju potrebne funkcije iz knjižnice za korištenje u dokument. Funkcija *render* koristi se za virtualno prikazivanje (engl. *rendering*) komponente gumba koji se dodaje u dokument u trećoj liniji koda i ubacuje ju u *document.body* čvor (engl. *node*). *Document.body* čvor predstavlja dio koji se nalazi između <body> tagova u HTML dokumentu. Pomoću *screen* objekta možemo pristupiti renderiranom HTML-u.

Primjer sadrži tri testa. Prvi test testira prikaz komponente na korisničkome sučelju pomoću *render* metode. Test radi na način da se pomoću *screen* objekta i *button* oznake u HTML-u uhvati gumb. *getbyRole* funkcija prihvaća dva argumenta: prvi je uloga komponente, u ovome slučaju gumb, a drugi parametar predstavlja ime, odnosno u ovome određenom slučaju tekstualni sadržaj gumba. Kosa crta i slovo „i" (/i) označavaju zanemarivanje velikih i malih slova. Nakon toga se funkcijom *expect* radi provjera podudarnosti prikaza na ekranu u odnosu na ono što je definirano u samome testu. Iz primjera je vidljivo da se `.toBeInTheDocument` funkcijom provjerava postoji li komponenta gumba na ekranu, a funkcijom `.toHaveTextContent` sadrži li tekstualni sadržaj „Click". Sljedeća dva testa na sličan način testiraju okida li se gumb na ponašanje klika i je li klik onemogućen ako je gumb u onemogućenome stanju (engl. *disabled*).

```
import { render, screen } from "@testing-library/react";
import userEvent from "@testing-library/user-event";

import Button from "../components/Button";
```

```
test("Button renders", () => {
render(<Button>Click</Button>);
    const button = screen.getByRole("button", { name:
/Click/i });
    expect(button).toBeInTheDocument();
    expect(button).toHaveTextContent("Click");
});
test("Button onClick triggers", () => { const testOnClick =
jest.fn();
    render(<Button onClick={testOnClick}>Click</Button>);
    const button = screen.getByRole("button", { name:
/Click/i });
    userEvent.click(button);
    expect(testOnClick).toHaveBeenCalledTimes(1);
});
test("Button is disabled", () => { const testOnClick =
jest.fn();
    render( <Button onClick={testOnClick}
disabled>Click</Button> );
    const button = screen.getByRole("button", { name:
/Click/i });
    userEvent.click(button);
    expect(button).toBeInTheDocument();
    expect(button).toBeDisabled();
    expect(testOnClick).toHaveBeenCalledTimes(0);
});
```

Kôd 4.20 Testiranje komponente gumba pomoću React Testing knjižnice

4.3. Korisničko testiranje

Korisničko testiranje dugo je uspostavljana istraživačka tehnika kojom se daje odgovor na pitanje: „Kako bi krajnji korisnik reagirao na naš softver u realnim uvjetima?" Sastoji se od promatranja reprezentativnoga krajnjeg korisnika u interakciji s proizvodom, kojemu je dan cilj koji treba postići, ali bez danih uputa za korištenje proizvodom. Članovi tima promatraju radnje korisnika bez uplitanja i bilježe ono što se događa. Analizom nakon testiranja utvrđuju se sve poteškoće s kojima se korisnik susreo ilustrirajući tako razlike između pretpostavki

tima i stvarnoga ponašanja. Korisničko testiranje nije izričito agilna praksa, ali je 2008. godine privuklo veliku pozornost integracije s agilnim tehnikama razvoja.

Mnoge organizacije prelaze s tradicionalnih praksi razvoja softvera na prakse agilnoga razvoja. Ova *web* aplikacija razvijala se iterativnim pristupom s dvotjednim sprintom kao osnovnom jedinicom razvoja. Uz tako kratke rokove programeri često zaobilaze uporabljivost jer pretpostavljaju da nema vremena za testiranje i korisnička istraživanja. Jednostavno korisničko testiranje s nekoliko sudionika, izrada prototipova i heuristička procjena nude jeftin, brz i rani fokus na uporabljivost.

Prilikom korisničkoga testiranja aplikacije RODA uključeni su razvojni tim, vlasnik proizvoda (engl. *product owner*), dvije korisnice iz udruge RODA i *scrum master*, odnosno osoba u timu koja je odgovorna za to što će se iz dogovorenih zahtjeva korisnika razvijati i po kojem redoslijedu. Testiranje se provodilo na platformi Microsoft Teams te se razgovor snimao kako bi bio pogodan za kasniju analizu. Uključivanje prikaza sudionika *web* kamerom dodaje potpuno novu razinu dubine i detalja kvalitativnim uvidima u testiranje korisničkoga doživljaja. Time je ostvareno bolje razumijevanje ispitanikova kontekstualnoga okružja dok komunicira s korisničkim sučeljem i odgovara na pitanja. Uz spomenuto, prisutna je i vizualna analiza ispitanika kojom se prate njegove emocionalne i/ili fiziološke reakcije na proizvod u stvarnome vremenu.

Korištena je metoda razmišljanja naglas (engl. *thinking-aloud*) te je ona već devetnaest godina dobar pokazatelj dugovječnosti metoda korisničkoga testiranja. U testu se od korisnika traži razmišljanje naglas, tj. jednostavno verbaliziranje svojih misli prilikom kretanja kroz korisničko sučelje. Voditelj testa u tome slučaju obično mora poticati korisnika kako on ne bi prestao govoriti. Prednosti ove metode su otkrivanje što korisnik zaista misli o dizajnu i prepoznavanje njihovih zabluda, što često dovodi do djelotvornih preporuka za redizajn.

Prije sastanka s korisnicima napravljen je popis općih pitanja i nekoliko testnih scenarija. Prvi popis sastojao se od pitanja poput:

1. Što vidite na zaslonu?
2. Gdje se nalazite na zaslonu?
3. Što mislite za što služi zaslon na kojem smo trenutačno?
4. Koliko je lako ili teško bilo pronaći informacije na zaslonu?
5. Što mislite o rasporedu elemenata na stranici?

6. Koji su potencijalni problemi i prilike?

Unos podataka o rodilištu jedan je od testnih scenarija. Ovaj test provjerava unos i uređivanje podataka o rodilištu. Preduvjeti koji moraju biti zadovoljeni za testni scenarij unosa podataka su:

- korisnik je prijavljen u sustav
- korisnik se nalazi na početnoj stranici.

Pitanja koja se nalaze u formi za unos podataka prikazana su u tablici (Tablica 4.1 Prikaz podataka o rodilištu s numeriranim pitanjima).

Tablica 4.1 Prikaz podataka o rodilištu s numeriranim pitanjima

Naziv pitanja	Broj pitanja
Osoba odgovorna za točnost podataka.	1
Opće informacije o tečaju za trudnice.	2
Nabrojite što rodilja treba ponijeti u rodilište ZA SEBE.	3
Nabrojite što rodilja treba ponijeti u rodilište ZA DIJETE.	4
Koliko dana obično traje boravak u rodilištu nakon vaginalnoga poroda?	5
Koliko dana obično traje boravak u rodilištu nakon carskoga reza?	6
Kada je zakazan termin za posjete?	7
Kada starija braća i sestre mogu vidjeti dijete i majku?	8
Može li se s osobljem dogovoriti drugi termin za posjet ako obitelj rodilje nije u mogućnosti posjetiti ju za vrijeme redovnih posjeta (zbog poslovnih obveza, udaljenosti itd.)?	9
Postoji li apartman za boravak prije/poslije poroda?	10
Cijena apartmana u kunama po danu.	11

Postoji li vađenje matičnih stanica iz pupkovine?	12
Cijena vađenja matičnih stanica.	13
Ograničavate li u određenim slučajevima pratnju na porodu? (osim što osoba mora biti trijezna i ne smije biti nasilna)	14
Može li osoba u pratnji biti primalja?	15
Može li osoba u pratnji biti osobna asistentica ako je rodilja žena s invaliditetom?	16
Mora li osoba u pratnji platiti naknadu za prisustvovanje porodu?	17
Ako je odgovor DA, kolika je cijena u kunama?	18
Postoji li u rodilištu mogućnost epiduralne analgezije?	19
Kada je anesteziolog na raspolaganju za epiduralnu analgeziju?	20

Sudionici koji sudjeluju u ovome testnom scenariju medicinski su tehničari, no prilikom izvođenja testa, test je izvela korisnica udruge. Koraci za polaganje testa nalaze se u tablici (Tablica 4.2). Zelena boja u stupcu *Prolaz* označava prolaz, a crvena neuspjeh testa.

Tablica 4.2 Koraci za polaganje testa unosa podataka o rodilištu

Korak	Opis koraka	Očekivani rezultat	Prolaz	Napomena
1	S početne stranice pokrenuti akciju *Unesi.*	Sustav otvara formu za unos podataka.		
2	Unijeti podatke.	Podatci su uneseni.		
3	Kliknuti na *Potvrdi.*	Sustav sprema podatke i šalje ih na odobrenje.		

Preduvjeti za testni scenarij uređivanja podataka su:

- korisnik je prijavljen u sustav
- korisnik se nalazi na početnoj stranici
- korisnik je kliknuo na gumb *Uredi*.

Tablica 4.3 Koraci za polaganje testa uređivanja podataka o rodilištu

Korak	Opis koraka	Očekivani rezultat	Prolaz	Napomena
1	S početne stranice pokrenuti akciju *Uredi.*	Sustav otvara formu za unos podataka.		
2	Promijeniti neke podatke.	Podatci su promijenjeni.		
3	Kliknuti na gumb *Potvrdi.*	Sustav sprema podatke i šalje ih na odobrenje.		

Sljedeći test odnosi se na slučaj postavljanja lozinke. Testom se provjerava registracija novih članova (članova udruge i medicinskih tehničara), odnosno provjerava se inicijalno postavljanje lozinke. Preduvjeti koji moraju biti zadovoljeni za izvođenje testa su:

- korisnik je registriran u sustav
- korisnik je dobio *e-mail* za postavljanje lozinke.

Sudionici prikladni za ovaj test članovi su udruge i medicinski tehničari. Korisnik je kao ulazni podatak dobio lozinku (Tablica 4.4 Ulazni podatak za pristup testu postavljanja lozinke). Korake provedbe testa prikazuje tablica (Tablica 4.5 Koraci za polaganje testa postavljanja lozinke).

Tablica 4.4 Ulazni podatak za pristup testu postavljanja lozinke

	Lozinka
Ulazni podatak	lozinka2

Tablica 4.5 Koraci za polaganje testa postavljanja lozinke

Korak	Opis koraka	Očekivani rezultat	Prolaz	Napomena
1	Kliknuti na poveznicu dobivenu u *e-mailu* za postavljanje lozinke.	Prikazuje se stranica za postavljanje lozinke.		
2	Upisati ulazni podatak u polje za postavljanje lozinke.	Podatci su upisani.		
3	Upisati ponovno ulazni podatak u drugo polje za postavljanje lozinke.	Podatci su upisani.		
4	Kliknuti *Potvrdi*.	Sustav prikazuje skočni prozor s porukom: „Uspješno ste promijenili lozinku.”		

Tijekom ovakvih testova nerijetko se događa da neki dio sustava ne radi kako je korisnik zamislio. U tim slučajevima vodi se tablica s prikazom povijesti testiranja (Tablica 4.6 Prikaz povijesti testiranja za test postavljanja lozinke).

Tablica 4.6 Prikaz povijesti testiranja za test postavljanja lozinke

Broj testa	Test izvršio	Datum	Uspješnost testiranja	Komentar
1	Daniela Drandić	1. 9. 2021.	Test pao.	Poveznica za postavljanje lozinke vodi na stranicu za prijavu.
2	Daniela Drandić	15. 9. 2021.	Test prošao.	

Testiranje s korisnikom rađeno je u dvotjednim intervalima te su nakon svakoga implementiranja potrebna poboljšanja. Poboljšanja su najčešće vezana uz modificiranje određenih riječi i rečenica ili popularno zvano kreativno pisanje (engl. *copywriting*), kako bi korisniku bilo jasnije što se od njega traži. Također, korisnik je odobrio dizajn aplikacije.

Tester u timu provodio je češća testiranja te bi prijavljivao greške. Poboljšanja koja su napravljena najčešće su vezana uz greške u kodu koje bi na ekranu izazvale nepredviđena ponašanja, primjerice:

- nakon klika na gumb *Ispiši u PDF* i vraćanja na prethodne korake, na gumbu piše *DaljePreuzmi PDF*. Kod učitavanja koraka ili mijenjanja nekih polja, gumb je pod nazivom *DaljeUčitavanje...*.
- klikom na „Prikaži zahtjev" kod određenoga zahtjeva, par se sekundi prikazuje kako nema više zahtjeva za tu bolnicu (iako ima jer ne bi postojao gumb *Prikaži zahtjev*) te se naknadno prikaže cijeli zahtjev
- nedostaje validacija forme za dodavanje zdravstvenih djelatnika
- nakon što se klikne na poveznicu za postavljanje lozinke, preusmjeri se na stranicu za postavljanje lozinke, a nakon što se lozinka postavi, ne prikaže se *pop-up* s obavijesti da je lozinka postavljena, nego se preusmjeri na stranicu koja „ne postoji"
- potrebno je promijeniti „Broj telefona, adresu i radno vrijeme ambulante ili dnevne bolnice" u „Broj telefona, adresa i radno vrijeme ambulante ili dnevne bolnice".

Sve prijavljene greške za određeni sprint bile bi ispravljene te se po završetku sprinta aplikacija ponovno testirala s korisnikom.

Konačni je cilj razvoja ovakvoga tipa proizvoda lansiranje savršenoga proizvoda, no samo testiranjem i povratnim informacijama korisnika moguće je doraditi i poboljšati korisničko iskustvo.

Zaključak

Osnovni motivi izrade ove *web* aplikacije bili su: bolja informiranost budućih roditelja te optimizacija procesa prikupljanja podataka o rodilištima.

Omogućavanje rodilištima unos i uređivanje informacija o tome što nude, kao i dodavanje fotografija svojih zdravstvenih ustanova rodilišta, neke su od funkcionalnosti ove *web* aplikacije. Članovi udruge RODA više neće morati unositi i uređivati te informacije sami već će dodavati nove korisnike i odobravati njihove zahtjeve. Budući će roditelji, osim svih aktualnih informacija, imati mogućnost izrade osobnoga plana poroda u nekoliko klikova.

Proces izrade bio je vrlo dinamičan, imao je svoje uspone i padove te je svakim danom tražio visoku razinu koncentracije. Po završetku finalizirana aplikacija isporučena je krajnjemu korisniku, što opravdava korisnost i važnost ovoga završnog rada.

Dio aplikacije trenutačno je pohranjen na privatnome repozitoriju dok je ostatak javno objavljen i može mu se pristupiti na sljedećoj poveznici: https://rodilista-app.roda.hr/rode/plan-poroda.

Popis kratica

UI	*User Interface*	korisničko sučelje
UX	*User Experience*	korisničko iskustvo
HTML	*Hypertext Markup Language*	jezik za označavanje hiperteksta
CTA	*Call To Action*	poziv na akciju
SEO	*Search Engine Optimization*	optimizacija stranica za tražilice
SPA	*Single Page Application*	jednostrana aplikacija
DOM	*Document Object Model*	objektni model dokumenta
URL	*Uniform Resource Locators*	jedinstveni lokatori resursa
JSX	*JavaScript XML*	javascript XML
XML	*Extensible Markup Language*	proširivi jezik za označivanje
NPM	*Node Package Manager*	Node upravitelj paketa
NPX	*Node Package Execute*	Node izvršitelj paketa
SQL	*Structured Query Language*	strukturni jezik upita

Popis slika

Slika 2.1 Ploča vizije proizvoda 2

Slika 2.2 Persona – predsjednica Udruge RODA 4

Slika 2.3 Persona – trudnica / buduća majka 5

Slika 2.4 Persona – voditelj rodilišta 6

Slika 3.1 Dijagram *web* aplikacije za korisnika Udruge 12

Slika 3.2 Dijagram portala Udruge RODA 12

Slika 3.3 *Wireframe* prijave u desktop verziji 14

Slika 3.4 *Wireframe* početne stranice medicinskoga tehničara u desktop verziji 16

Slika 3.5 *Wireframe* početne stranice administratora Udruge RODA u desktop verziji 17

Slika 3.6 *Wireframe* početne stranice s odabranim rodilištem u desktop verziji 18

Slika 3.7 *Wireframe* početne stranice s označenom karticom *Zdravstveni djelatnici* u desktop verziji 19

Slika 3.8 *Wireframe* početne stranice s označenom karticom *Statistika* u desktop verziji . 20

Slika 3.9 *Wireframe* početne stranice s označenom karticom *Podatci o rodilištu* i pripadajućim skočnim prozorom u desktop verziji 21

Slika 3.10 *Wireframe* sa skočnim prozorom za dodavanje zdravstvenoga djelatnika..... 22

Slika 3.11 *Wireframe* sa skočnim prozorom za uređivanje zdravstvenoga djelatnika 23

Slika 3.12 Dizajn-sistem – paleta boja 25

Slika 3.13 Dizajn-sistem – tipografija 26

Slika 3.14 Dizajn-sistem – *Website grid* za stolna računala 27

Slika 3.15 Dizajn-sistem – *Website grid* za mobilne uređaje 28

Slika 3.16 Dizajn-sistem – ikone iz knjižnice Material UI..... 29

Slika 3.17 Dizajn-sistem – komponenta gumba iz knjižnice Material UI..... 30

Slika 3.18 Dizajn stranice za prijavu korisnika 31

Slika 3.19 Dizajn stranice ako je istekla poveznica za promjenu lozinke 31

Slika 3.20 Dizajn početne stranice administratora udruge RODA 32

Slika 3.21 Dizajn početne stranice administratora udruge RODA 32

Slika 3.22 Dizajn početne stranice administratora udruge RODA s odabranim rodilištem 33

Slika 3.23 Dizajn početne stranice medicinskoga tehničara 34

Slika 3.24 Dizajn početne stranice administratora udruge RODA s otvorenim skočnim prozorom 35

Slika 3.25 Dizajn početne stranice administratora udruge RODA s otvorenim skočnim prozorom 36

Slika 3.26 Dizajn stranice *Zahtjevi* 37

Slika 3.27 Dizajn početne stranice medicinskoga tehničara s otvorenom formom za popunjavanje ankete 38

Slika 3.28 Dizajn početne stranice medicinskoga tehničara za mobilne uređaje 39

Slika 3.29 Dizajn početne stranice administratora udruge s odabranim rodilištem za mobilne uređaje 40

Slika 4.1 Rezultati istraživanja najtraženijih *web-frameworka* 2022. godine 42

Slika 4.2 Relacijski model baze podataka za aplikaciju RODA 46

Slika 4.3 Rezultat izvršenja SELECT upita na tablici bolnica 47

Slika 4.4 Rezultat izvršenja složenijega upita na tablici korisnika 47

Slika 4.5 Broj tjednih preuzimanja u svibnju 2022. 48

Slika 4.6 Dodavanje Formik paketa pomoću Yarna 54

Popis tablica

Tablica 4.1 Prikaz podataka o rodilištu s numeriranim pitanjima 67

Tablica 4.2 Koraci za polaganje testa unosa podataka o rodilištu 68

Tablica 4.3 Koraci za polaganje testa uređivanja podataka o rodilištu 69

Tablica 4.4 Ulazni podatak za pristup testu postavljanja lozinke 69

Tablica 4.5 Koraci za polaganje testa postavljanja lozinke 70

Tablica 4.6 Prikaz povijesti testiranja za test postavljanja lozinke 70

Popis kôdova

Kôd 4.1 Jednostavan primjer MySQL upita 47

Kôd 4.2 Složeniji primjer MySQL upita 47

Kôd 4.3 Primjer funkcijske React komponente 48

Kôd 4.4 Primjer generičkoga TypeScript kôda 49

Kôd 4.5 Primjer TypeScript kôda korištenoga za dohvat zahtjeva rodilišta 50

Kôd 4.6 Instalacija MUI knjižnice pomoću Yarna 51

Kôd 4.7 Primjer jednostavnoga korištenja gotovih Material UI komponenti 51

Kôd 4.8 Korištenje ThemeProvider komponente 52

Kôd 4.9 Definiranje *brand* boja u temi 53

Kôd 4.10 Stilska konfiguracija Card komponente 53

Kôd 4.11 Primjer korištenja *useState hooka* 56

Kôd 4.12 Primjer korištenja *useEffect hooka* 56

Kôd 4.13 Primjer korištenja React Contexta 58

Kôd 4.14 Primjer korištenja Contexta u komponenti izvan mjesta definicije 58

Kôd 4.16 Primjer uključivanja React Router komponenti 59

Kôd 4.17 Primjena React Router komponenti 61

Kôd 4.18 Primjena Formika na jednostavnoj formi za prijavu 63

Kôd 4.19 Primjena Yup validacijske sheme 63

Kôd 4.20 Dodavanje React Testing Library paketa u projekt 64

Kôd 4.21 Testiranje komponente gumba pomoću React Testing knjižnice 65

Literatura

[1] S. BRANSON, *UX / UI Design: Introduction Guide To Intuitive Design And User-Friendly Experience*, nezavisno objavljeno, 2020.

[2] J. GOTHELF, J. SEIDEN, *Lean UX: Designing Great Products with Agile Teams,* 2rd Edition, O'Reilly Media, 2016.

[3] A. BEAULIEU, *Learning SQL: Generate, Manipulate, and Retrieve Data*, United States of America, O'Reilly Media, 2020.

[4] V. ZDEŠIĆ, *Priručnik - Standardi u primjeni internetske tehnologije*, Zagreb, Visoka škola za primjenjeno računarstvo, 2009.

[5] L. MARAS, *Razvoj bogatih internet aplikacija*, Zagreb, Visoka škola za primjenjeno računarstvo, 2014.

[6] https://reactjs.org/, travanj 2022.

[7] https://mui.com/material-ui/icons/, travanj 2022.

[8] https://www.typescriptlang.org/, svibanj 2022.

[9] https://reactrouter.com/, svibanj 2022.

[10] https://formik.org/, svibanj 2022.

[11] https://yarnpkg.com/, svibanj 2022.

[12] https://reactjs.org/docs/hooks-overview.html, svibanj 2022.

[13] https://www.mysql.com/, kolovoz 2022.

[14] https://testing-library.com/, kolovoz 2022.

[15] https://classic.yarnpkg.com/lang/en/docs/yarn-lock/, listopad 2022.

[16] https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/, studeni 2022.

Prilog

Dio aplikacije namijenjen za administraciju (engl. *back office*) dostupan je na poveznici https://rodilista-app.roda.hr/rode/prijava te su za prijavu potrebni pristupni podatci. Navedeni dio posjetiteljima portala nije dostupan, već samo zaposlenicima udruge i rodilišta, kojima pomaže u upravljanju informacijama kako bi olakšao razmjenu informacija između udruge RODA i rodilišta.

Web stranica koja sadrži informacije i personaliziranu izradu plana poroda javno je dostupna i može joj se pristupiti na poveznici: https://rodilista-app.roda.hr/rode/plan-poroda. Izrada plana poroda, odnosno obrazac za izradu plana poroda, nalazi se na poveznici: https://rodilista-app.roda.hr/rode/moj-plan-poroda.