Razvoj web aplikacije namijenjene online naručivanju termina kod stomatologa

Lončarić, Laura

Sveučilište Jurja Dobrile

Fakultet informatike u Puli

Laura Lončarić

Razvoj web aplikacije namijenjene online naručivanju termina kod stomatologa

Završni rad

Sveučilište Jurja Dobrile

Fakultet informatike u Puli

Laura Lončarić

Razvoj web aplikacije namijenjene online naručivanju termina kod stomatologa

Završni rad

Ime Prezime studenta, JMBAG: Laura Lončarić, 0303092470

Studijski smjer: preddiplomski sveučilišni studij Informatika

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Kolegij: Programiranje

Mentor: izv. prof. dr. sc. Tihomir Orehovački

U Puli, rujan, 2022.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Laura Lončarić, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Laura Lončarić

U Puli, 20.09.2022.

IZJAVA O KORIŠTENJU
AUTORSKOG DJELA

Ja, Laura Lončarić dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj Završni rad pod nazivom „Razvoj web aplikacije namijenjene online naručivanju kod stomatologa“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 20.09.2022.

Potpis

Laura Lončarić

Sažetak

U današnje vrijeme web aplikacije sve su više tražene i razvijaju se velikom brzinom. Web aplikacije su programi koji omogućuju bolju komunikaciju između tvrtki i njihovih korisnika. Korisnici više preferiraju online naručivanje. Web aplikacija namijenjena online naručivanju kod stomatologa napravljena je koristeći Vue.js, Tailwind CSS i Firebase platformu. Pruža listu privatnih ordinacija na jednom mjestu. Omogućava korisnicima online zakazivanje termina u nekoliko koraka, odabirom ordinacije, usluge i datuma termina što olakšava i ubrzava proces naručivanja kod stomatologa. S druge strane, pruža sučelje ordinacija za praćenje rasporeda zakazanih termina, dodavanje, brisanje i mijenjanje detalja termina pacijenata. Digitalizacija ordinacija razvija poslovanje i održava klijente zadovoljnima što pomaže ordinacijama poboljšati poveznice s pacijentima.

Ključne riječi

Online naručivanje, stomatološka ordinacija, web aplikacija, Vue.js, Firebase, Tailwind CSS

Abstract

Nowadays, web applications are increasingly in demand and are developing at a high speed. Web applications are programs that enable better communication between companies and their users. Users prefer online ordering. The web application intended for online ordering at dentists was created using Vue.js, Tailwind CSS and the Firebase platform. It offers a list of private surgeries in one place. It allows users to schedule a terminal online in a few steps by selecting the office, service and date of the terminal, which facilitates and speeds up the ordering process at the dentist. On the other hand, it provides an interface for practices to monitor the schedule of booked appointments, add, delete and change patient appointment details. Digitizing practices increases business and makes customers happy, which helps practices improve patient relationships.

Keywords

Online ordering, dental office, web application, Vue.js, Firebase, Tailwind CSS

Sadržaj

1. Uvod

Web aplikacije postale su bitna značajka poslovanja u današnjem svijetu. Korištenjem web aplikacija tvrtke se mogu puno brže razvijati i postizati svoje ciljeve. Koriste se za povećanje učinkovitosti, dijeljenje i pružanje informacija. Web aplikacija je interaktivni program koji radi na web poslužitelju kojoj korisnici pristupaj putem softvera, poznatog kao web preglednik [1]. Može im se pristupiti s bilo kojeg uređaja s omogućenim internetom, poput stolnih i prijenosnih računala, telefona i sl. Obično su kodirane u jeziku koji podržava preglednik kao što su JavaScript i HTML, budući da se ti jezici koriste za prikaz izvršnog programa na pregledniku.

Razvoj web aplikacija odnosi se na proces utvrđivanja zahtjeva, dizajniranja, izgradnje, testiranja i implementacije web aplikacije. Svaka aplikacija počinje s idejom koja je potencijalno rješenje nekog problema. Zahtjev je svojstvo koje treba udovoljiti kako bi se riješio problem. Stoga, utvrđivanjem zahtjeva identificiraju se funkcionalni i nefunkcionalni zahtjevi web aplikacije te postavljaju ciljevi koje treba ispuniti. Procesom dizajniranja određuje se arhitektura sustava, komponente, sučelje i ostale karakteristike sustava. Zatim se implementira rješenje korištenjem programskih jezika i alata.

Sustavi za online zakazivanja termina su budućnost. To su programi koji omogućuju korisnicima zakazivanje termina na web stranici, nakon čega se termin automatski sprema u kalendar. Pružaju rješenje koje pružateljima usluga olakšava upravljanje terminima. Dokumentacija objašnjava razvoj web aplikacije namijenjene online naručivanju termina kod stomatologa. Aplikacija predstavlja skup privatnih stomatoloških ordinacija omogućavajući korisnicima odabir željene ordinacije te odabir termina za istu. S druge strane prikazuje sučelje pojedine ordinacije s rasporedom zakazanih termina i dodatnim funkcionalnostima. Omogućila bi bolje korisničko iskustvo, više fleksibilnosti te smanjila vrijeme čekanja. Isto tako omogućila bi pružateljima usluge bolju kontrolu nad pacijentima i poboljšala učinkovitost osoblja. Za razvoj aplikacije korišten je Vue.js 3 JavaScript Framework, Tailwind CSS Framework za izgradnju korisničkog sučelja te Firebase platforma koja je NoSQL baza podataka. Dokumentacija opisuje postojeća i konkurentska rješenja te prednosti i poboljšanja ove web aplikacije. U četvrtom poglavlju detaljno su opisane korištene tehnologije i razvoj okruženja. Funkcionalnosti aplikacije opisane su u četvrtom poglavlju uz priložene snimke ekrana i programskog koda. Na samom kraju dokumentacije nalazi se zaključak rada te korištena literatura.

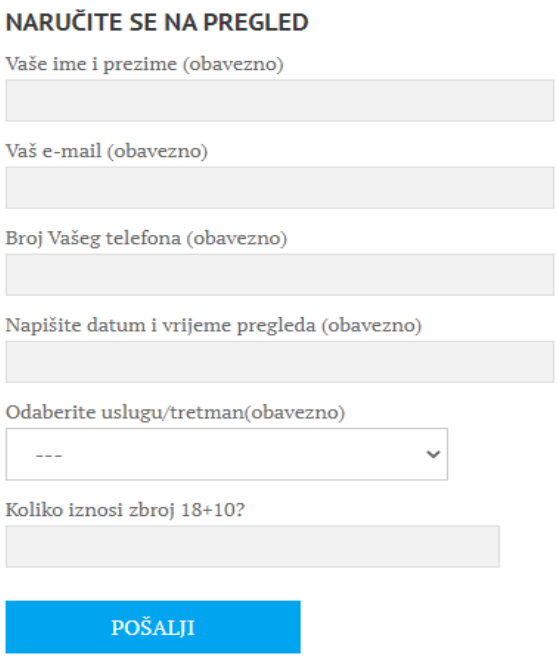
Projekt se nalazi na GitHub repozitoriju: https://github.com/LoncaricLaura/SmileWithUs .

2. Kritička analiza

Posjet stomatologu normalna je rutina svih ljudi. Redovni stomatološki pregled odvija se svakih šest mjeseci. Proces zakazivanja termina većinom se odvija telefonski. Pacijent unaprijed nazove stomatološku ordinaciju, kaže razlog svog posjeta i zatim stomatološka ordinacija nudi slobodne termine. Stomatološke ordinacije, kao i ostale zdravstvene ustanove primaju telefonske pozive i zahtjeve za termine u određenim satima. Kao posljedica toga, zauzete telefonske linije česti su problem pacijenata. Za pacijente koji rade i imaju svoje obaveze tijekom dana, ovaj nedostatak pristupa predstavlja problem.

U zdravstvu tehnologija se nije razvila u skladu s očekivanjima korisnika. Kao rezultat toga, oko 85% korisnika zakazuje termina u zdravstvenim ustanovama telefonski [2]. Isto tako oko 80% pacijenata preferira liječnika koji nudi online zakazivanje, no to se odnosi uglavnom na mlađu populaciju [2].

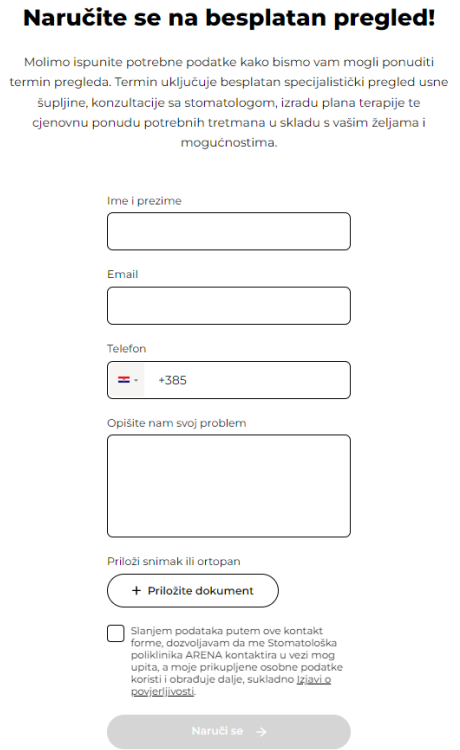
Jedan od primjera web aplikacije za online naručivanje termina kod stomatologa u Hrvatskoj je privatna stomatološka ordinacija dentalne medicine Čeović [12]. Omogućava online naručivanje prvog stomatološkog pregleda. Proces naručivanja vrši se ispunjavanjem obrasca osobnim podacima: ime i prezime, email, broj telefona, datum i vrijeme te odabir usluge. Nakon ispunjenog obrasca ordinacija telefonski kontaktira korisnika kako bi potvrdila zakazani termin. Ukoliko je upisani termin u obrascu zauzet, ordinacija nudi korisniku drugi slobodan termin. Primjer obrasca za online naručivanje stomatološke ordinacije Čeović prikazan je na slici 1.



Slika 1. Obrazac za online naručivanje stomatološkog pregleda u ordinaciji Čeović

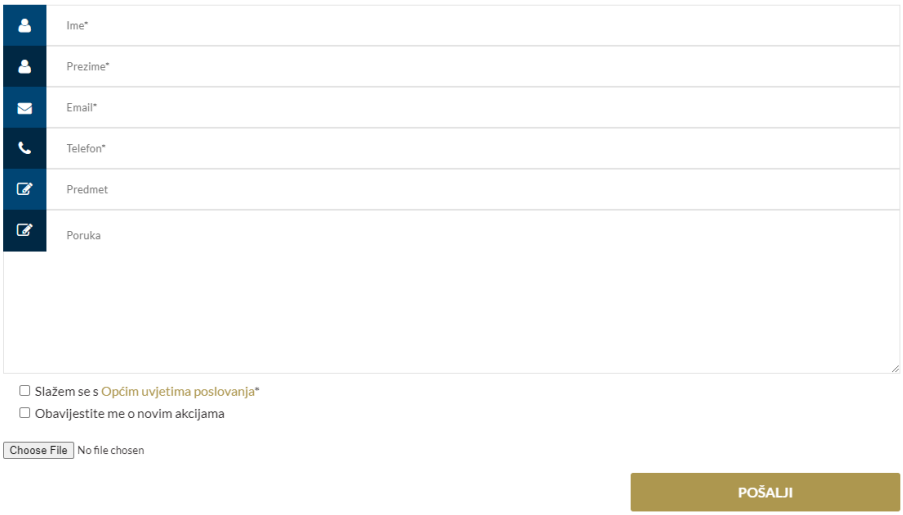
Stomatološka ordinacija Čeović pruža online naručivanje samo za prvi stomatološki pregled. Ne pruža ponovnu mogućnost online naručivanja. Također tijekom ispunjavanja obrasca podaci se unose ručno, što može dovesti do krivog unosa podataka, te može uzrokovati nesporazum oko naručivanja pregleda.

Nadalje, stomatološka ordinacija ARENA omogućuje online naručivanje ispunjavanjem obrasca osobnim podacima: ime i prezime, email, broj telefona i opis problema [13]. Također omogućuje korisnicima dodavanje dokumenta (snimke) ukoliko je potrebno. Nakon uspješno poslanog obrasca ordinacija kontaktira korisnika putem telefona ili email-a kako bi dogovorili termin pregleda. Način online naručivanja prikazan je na slici 2. Ordinacija ne pruža popis usluga koje pruža kako bi korisniku olakšali odabir. Isto tako ne pruža mogućnost odabira željenog termina. Proces naručivanja i dalje zahtjeva telefonske pozive.



Slika 2. Obrazac za online naručivanje stomatološkog pregleda u ordinaciji ARENA

Još jedan primjer stomatološke ordinacije u Hrvatskoj koja nudi online rezerviranje termina je ŠkodaDENT [14]. Ispunjavanjem obrasca unosom osobnih podataka šalje se upit, jednako kao i na prijašnjim primjerima. Ordinacija zatim šalje potvrdu o terminu s dodijeljenim datumom i vremenom termina. Obrazac za online naručivanje prikazan je na slici 2.



Slika 3. Obrazac za online naručivanje stomatoloških pregleda u ordinaciji ŠkodaDENT

S obzirom da gotovo nijedna stomatološka ordinacija u Hrvatskoj ne pruža potpuni proces online naručivanja, aplikacija pod nazivom SmileWithUs nudila bi popis privatnih stomatoloških ordinacija s detaljno opisanim uslugama, cijenama usluga i drugim bitnim informacijama. Svakom pacijentu pružila bi se mogućnost odabira željene stomatološke ordinacije, odabir usluge prema potrebi te odabir datuma termina. S druge strane, aplikacija bi sadržavala takozvano „admin“ sučelje. Namijenjeno je stomatološkim ordinacijama pružajući im sučelje koje prikazuje kalendar sa zakazanim terminima i dodatnim funkcionalnostima. Kada pacijent zakaže termin, odabrana stomatološka ordinacija dobiva obavijest o novom terminu na sučelju. Zaposlenik u stomatološkoj ordinaciji pregledava podatke zakazanog termina te mu dodjeljuje točno vrijeme. Ukoliko su svi termini odabranog datuma pacijenta zauzeti dodijelit će mu prvi idući slobodan termin. Nakon toga zakazani termin automatski se dodaje u kalendar ordinacije. Pacijent dobiva povratnu informaciju putem email-a sa svim detaljima zakazanog termina.

3. SWOT analiza

Kako bi se identificirale jake i slabe točke u donošenju odluka te ostvario željeni cilj koristi se SWOT analiza. SWOT analiza je moćna tehnika kritičkog razmišljanja za analizu svih vanjskih i unutarnjih čimbenika. Pomaže u analizi onoga što organizacija dobro radi i razvijanju buduće

strategije. Usredotočuje se na četiri elementa: snaga, slabosti, prilike i prijetnje. Elementi SWOT analize prikazani su u tablici 1.

Snaga	Slabosti	Prilike	Prijetnje
Jednostavan i brz proces zakazivanja termina Fleksibilnost Smanjeno vrijeme čekanja Poboljšana operativna učinkovitost i učinkovitost osoblja Ideja koja još nije dovoljno razvijena u zdravstvenim ustanovama	Pacijenti nemaju prikaz zauzetih termina Pacijenti nemaju mogućnost odabira točnog vremena Nije omogućena odjava termina	Bolje korisničko iskustvo Više pregledanih pacijenata Digitalizacija stomatoloških ordinacija Unapređivanje procesa	Brz razvoj novih tehnologija Povećanje troškova stomatoloških ordinacija Ne pridržavanje zakazanih termina

Tablica 1. SWOT analiza

Kao što je prije spomenuto, proces zakazivanja termina većinom se odvija telefonski. Zdravstvena tehnologija nije još razvijena u potpunosti, stoga su online zakazivanja termina u tom sektoru još uvijek rijetka. Telefonsko zakazivanje termina u prosjeku traje nekoliko minuta, dok online zakazivanje minutu. Aplikacija SmileWithUs nudi jednostavan i brz proces rezervacije termina, u samo tri koraka. Naravno, aplikacija nudi telefonski kontakt s obzirom da stariji pacijenti nisu vješti u tehnologijama kao mladi. Osim što ubrzava proces, aplikacija je potpuno fleksibilna. Prilagođena je različitim veličinama zaslona. Stoga, koristeći svoj pametni telefon pacijent se može naručiti u bilo kojem trenutku, s bilo kojeg mjesta. Online rezervacijama termina osigurava da pacijenti ne moraju provoditi vrijeme čekajući u prostorijama ordinacija što sprječava gužve. Time ordinacije imaju veću kontrolu nad pacijentima. Cijeli proces aplikacije pomaže ordinacijama popuniti praznine u terminima što je često uzrokovano nedolaskom pacijenta ili kasnim otkazivanjem termina. Ordinacije primaju puno telefonskih poziva i upita tijekom dana, a uz to potroše puno vremena kako bi ručno unijeli zakazane termine. U ovoj aplikaciji pacijenti unose sve potrebne podatke u sustav prilikom

zakazivanja termina, što znači da više nema primanja i zapisivanja informacija preko telefona. Ovi pozitivni unutarnji i vanjski čimbenici čine cijelo poslovanje učinkovitijim. S druge strane, aplikacija ima slabosti koje mogu utjecati na postizanje cilja. Prilikom zakazivanja termina pacijent nema prikaz zauzetih termina što može dovesti do neslaganja pružatelja usluga i korisnika. Opcija nije dodana zbog nemogućnosti prilagođavanja korištenog library-a u određene svrhe što će biti pojašnjeno u sljedećem odjeljku. Ukoliko bi se aplikacija realizirala, kalendar za odabir termina napravio bi se bez korištenja library-a kako bi se pacijentima pružio prikaz slobodnih termina te odabir željenog slobodnog datuma i točnog vremena. Time bi još više ubrzali proces online naručivanja. Također, aplikacija nema mogućnost otkazivanja termina što bi moglo ponovno uzrokovati ordinacijama puno telefonskih poziva. S obzirom da se web aplikacije i nove tehnologije razvijaju velikom brzinom, potrebno je predvidjeti moguće promijene i prijetnje kako bi aplikacija ostvarila cilj i značajan pomak u odnosu na konkurenciju te privukla što više pacijenata.

4. Opis korištenih tehnologija i razvojnog okruženja

Kao što je spomenuto u uvodu aplikacija SmileWithUs napravljena je koristeći Vue.js 3 JavaScript Framework, Firebase platforme i Tailwind CSS Framework-a.

Aplikacija je izrađena u Visual Studio Code editoru koda. Uključuje podršku za otklanjanje pogrešaka, isticanje sintakse, inteligentno dovršavanje koda, isječke te ugrađeni Git [11]. Pruža alate koji omogućavaju brže i lakše pisanje koda.

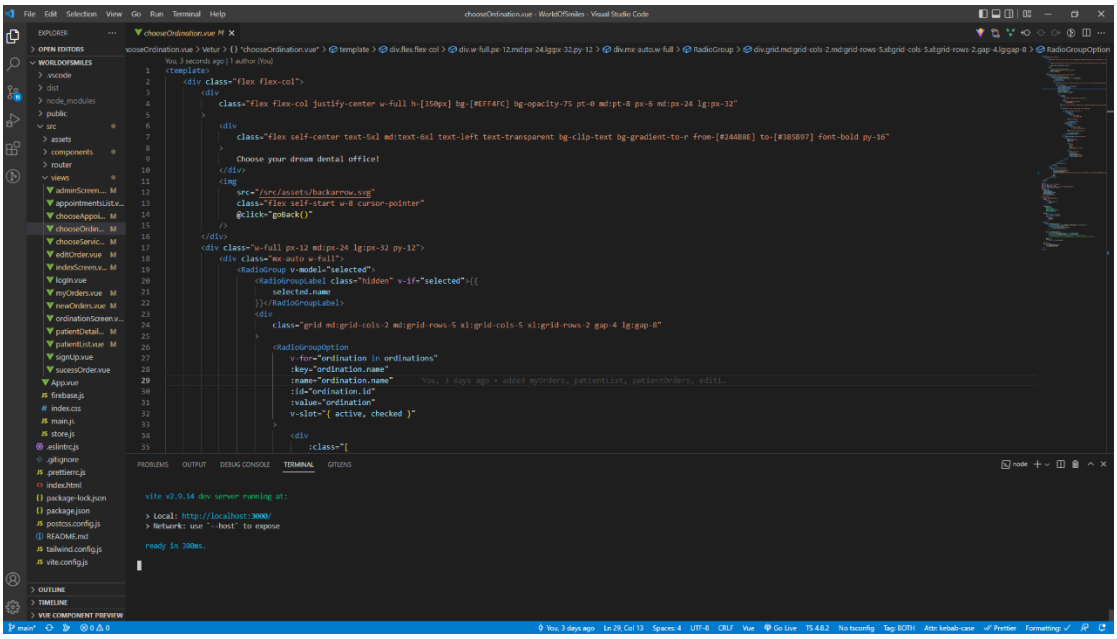
Vue je popularan JavaScript okvir za izgradnju korisničkih sučelja [3]. Jedan je od najjednostavnijih okvira prilagođen korisniku koji dolazi sa sjajnim bibliotekama. Otvorenog je koda koji je lako usvojiti. Odnosi se na HTML, CSS i JavaScript koji su korišteni u implementaciji aplikacije te pruža deklarativni model programiranja temeljen na komponentama koje omogućuju učinkovito razvijanje korisničkog sučelja [3]. Bavi se objektnim modelima dokumenata (DOM) koji su temeljeni na HTML-u. Osigurava da se aplikacija kreće u skladu s vremenom i novim značajkama sa svakom novom verzijom.

JavaScript je jedan od najpopularnijih programskih jezika, programski jezik weba. Omogućuje implementaciju složenih značajki na web stranicama i koristi se za mnoga okruženja poput Node.js, Apache CouchDb i Adobe Acrobat [4]. To je skriptni jezik koji se dinamički ažurira, kontrolira multimedije, animiranje slika i sl. [5].

HTML je jezik koji se koristi za strukturiranje i davanje značenja web sadržaju kao što je definiranje odlomaka, naslova, tablica, ugrađivanje slika i sl. [5]. Korišten je CSS koji predstavlja jezik stilskih pravila. Koristi se za primjenu stila na HTML sadržaj (postavljanje boja, fontova i sl.) [5].

U projektu korišten je Tailwind CSS, okvir koji je dizajniran da korisnicima omogući bržu i lakšu izradu aplikacija [6]. Temelji se na CSS-u te pruža CSS definirane klase koje čine proces oblikovanja jednostavnijim i praktičnijim. Time sprječava korisnika da piše dugačak kod. Tailwind CSS skenira sve HTML datoteke, JavaScript komponente i ostale predloške za nazive klasa te generira odgovarajuće stilove i zapisuje u statičnu CSS datoteku [7].

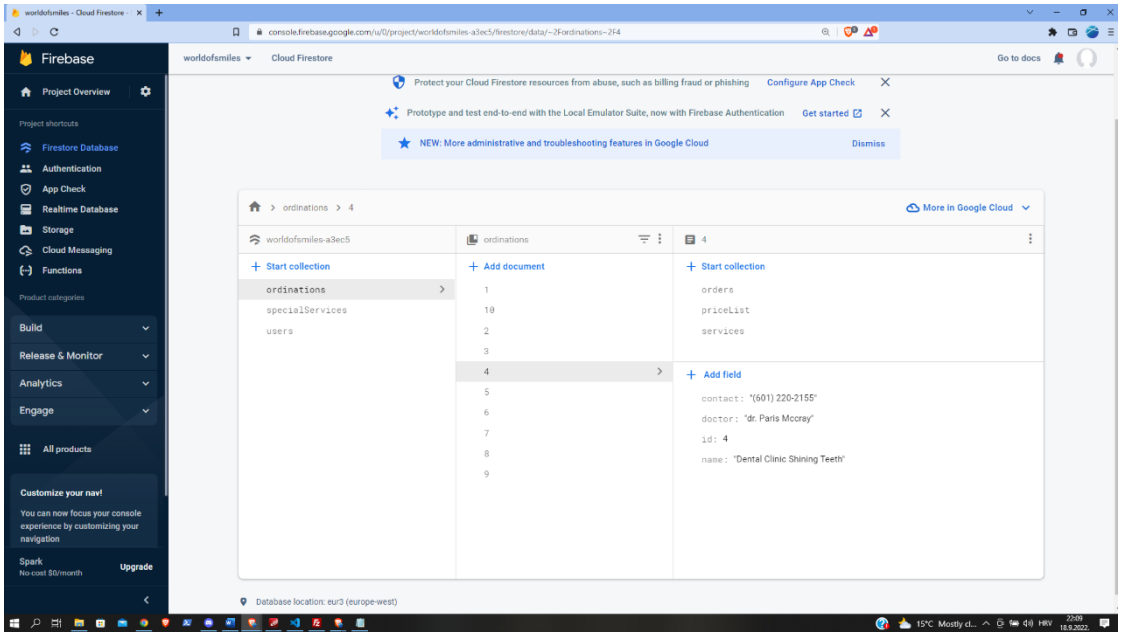
Na slici 4 prikazan je editor Visual Studio Code u kojem je postavljen Vue JavaScript okvir. Prikazuje način pisanja koda, koristeći Tailwind CSS.



Slika 4. Isječak iz Visual Studio Code editora

Baza podataka je temelj razvoja web aplikacije. Predstavlja organiziranu zbirku podataka ili strukturiranih informacija pohranjenih u računalnom sustavu. U projektu korištena je Firebase platforma koja služi za razvoj aplikacija. Firebase Realtime Database je NoSQL baza podataka smještena u oblaku koji omogućuje pohranu i sinkronizaciju podataka. Ne temelji se na relacijskog algebri kao većina baza, što znači da ne sadržava tablice i retke nego dokumente i kolekcije te nije ograničena unaprijede poznatom shemom. Ažurira podatke u stvarnom vremenu kako se mijenjaju u bazi podataka i ostaju dostupni čak i kada je aplikacija izvan

mreže. Također podržava autentifikaciju pomoću lozinki, telefonskih brojeva, Googlea i sl. Isječak Firebase platforme projekta prikazana je na slici 5.

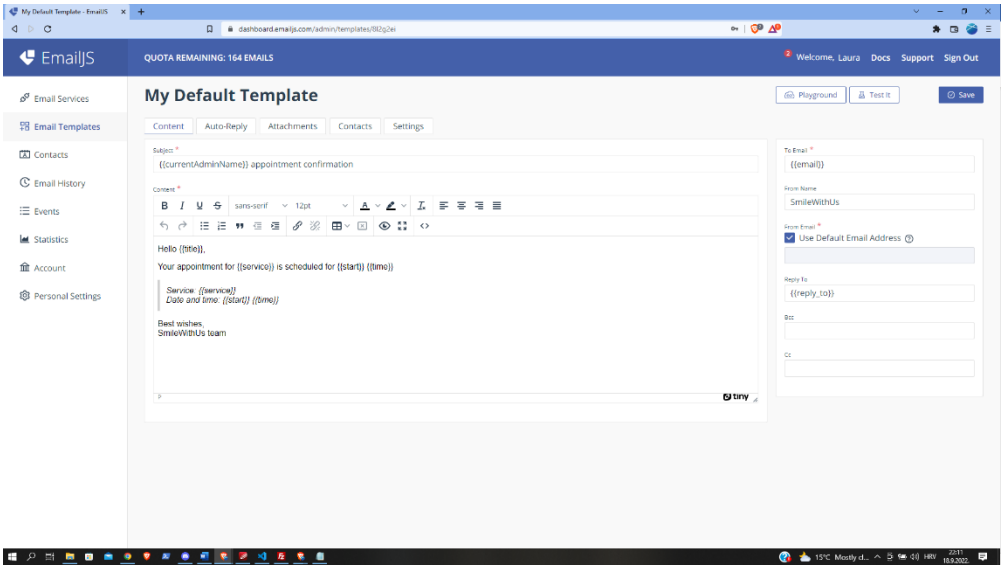


Slika 5. Isječak Firebase platforme

Osim postavljenog Vue.js okvira, JavaScript Framework-a za bazu podataka i Tailwind CSS okvira za dizajn korištene su razne biblioteke poput Vuex-a, Datepicker, FullCalendar i EmailJS. Vuex je obrazac upravljanja stanjem i biblioteka za Vue.js aplikacije [8]. Služi kao središnje spremište za komponente u aplikaciji [8]. Sadržava state kao osnovni spremnik koji sadržava state aplikacije. Vuex state je reaktivan što omogućuje da kada se komponente dohvate iz njega, on će se reaktivno i učinkovito ažurirati ako se stanje state-a promijeni.

S obzirom da aplikacija sadržava nekoliko komponenti kalendara korištene su komponente Datepicker i FullCalendar, Datepicker je jednostavna Vue.js komponenta za odabir datuma [9]. Korišten je na korisničkom i admin sučelju. Komponenta nudi niz props-a koji omogućuju prilagođavanje kalendara prema potrebi. Onemogućen je odabir datuma nedjeljom te omogućen odabir datuma dva mjeseca unaprijed. Također ograničen je odabir sati prema radnom vremenu ordinacija. Datepicker pruža onemogućavanje odabira određenih datuma, no ova komponenta nije omogućila potpuno prilagođavanje potrebama projekta. S obzirom da se u kalendaru pacijentima trebaju prikazati slobodni termini prikazujući slobodne datume i slobodne sate, nije moguće izvesti navedeno. S druge strane korišten je FullCalendar za prikaz rasporeda zakazanih termina pojedine ordinacije. To je JavaScript biblioteka koja se integrira s JavaScript okvirima, kao što su Vue, React, Angular [10]. Omogućuje dodavanje događaja u JavaScriptu te podržava različite poglede uključujući dnevne, tjedne i mjesečne.

Na samom kraju projekta korišten je EmailJS, usluga koja omogućuje slanje e-pošte izravno s JavaScript koda na strani klijenta. EmailJS povezuje se s podržanom uslugom e-pošte, zatim se radi predložak e-pošte. Predlošci sadržavaju dinamičke varijable koje se popunjavaju iz JavaScript poziva kojima su definirani elementi predloška (predmet, sadržaj, TO adresa, FROM naziv i sl.). Predložak e-pošte EmailJS-a prikazan je na slici 6.



Slika 6. EmailJS, predložak e-pošte

5. Opis funkcionalnosti

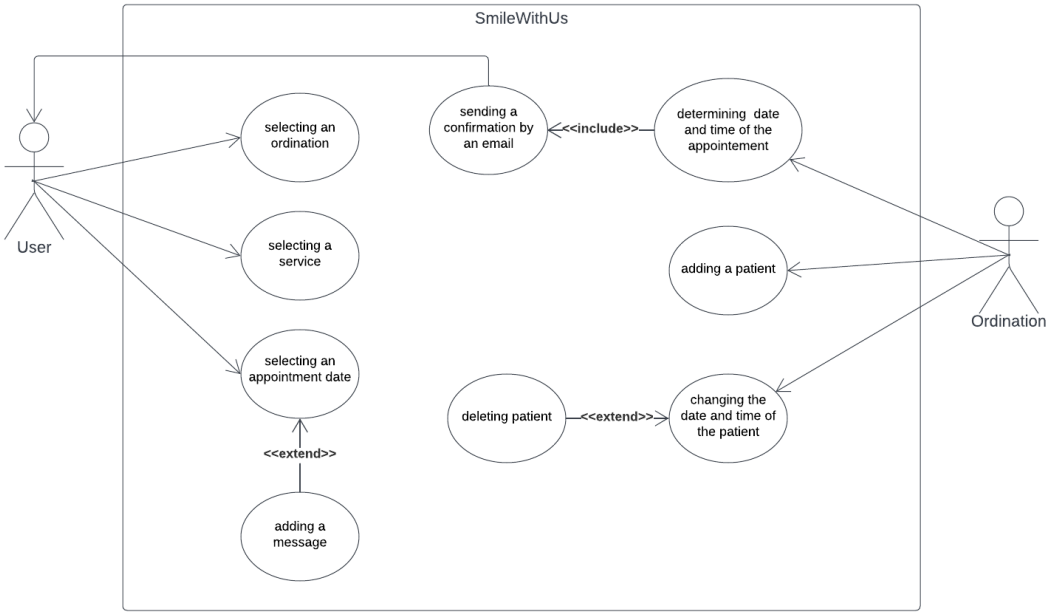
5.1 Use Case dijagram

Učinkovit model je ključan za pravilnu izvedbu aplikacije. Kako bi se postigli željeni ciljevi projekta izrađen je Use Case dijagram. To je grafički prikaz mogućih interakcija korisnika sa sustavom. Predstavlja niz operacija koje izvodi sustav čiji je rezultat akteru od neke koristi. Prikazuje različite interakcije i tipove korisnika koje aplikacija nudi što je prikazano na slici 7.

Aplikacija omogućuje krajnjim korisnicima pregled ponuđenih privatnih ordinacija, njihovih usluga i cijena. Svaki korisnik ima mogućnost odabira željene ordinacije, nakon čega odabire potrebnu uslugu. Na kraju odabire željeni datum termina te dodaje dodatnu poruku kao napomenu ukoliko je potrebno. Također korisnici mogu pregledavati svoje zakazane narudžbe u svim ordinacijama.

S druge strane, aplikacija sadržava admin sučelje, kojem imaju pristup samo ordinacije. Prijavom u sustav svakoj ordinaciji nudi se kalendar s rasporedom zakazanih termina po

mjesecu, tjednu i danu. Ukoliko se pacijent naruči telefonski, ordinacije imaju mogućnost dodavanja pacijenta. Zakazivanjem termina pacijenta, ordinacija dobiva obavijest o novom terminu sa svim njegovim detaljima, nakon čega postavlja točno vrijeme termina. Ukoliko je odabrani datum termina pacijenta zauzet, ordinacija osim postavljanja točnog vremena, mijenja i datum termina. Nakon uspješnog određivanja termina korisnik dobiva potvrdu termina sa svim njegovim detaljima putem email-a. Ordinacije imaju pregled svih pacijenata i njihovih zakazanih termina. U slučaju da se korisnik ipak nije u mogućnosti doći na zakazani termin ordinacija ima opciju mijenjanja detalja termina. Isto tako ukoliko pacijent odustane od svog termina ordinaciji se pruža mogućnost brisanja pacijenta.



Slika 7. Use Case dijagram

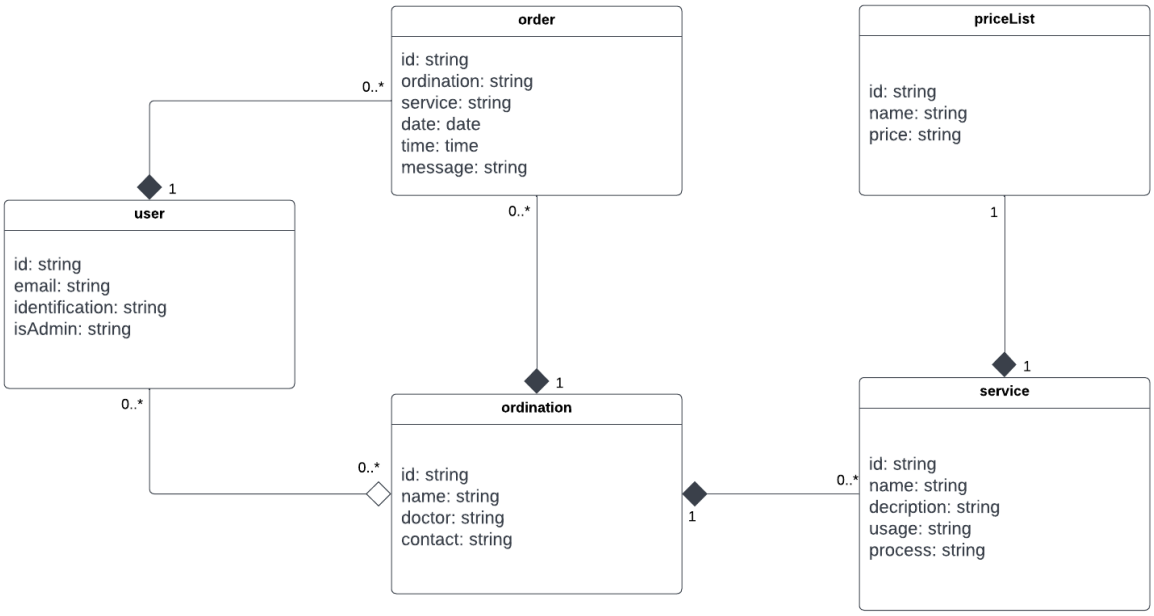
5.2 Klasni dijagram

Kao što je već spomenuto, aplikacija nudi prikaz svih privatnih ordinacija prijavljenim i ne prijavljenim korisnicima. Svaki korisnik ima mogućnost registracije ili prijave u aplikaciju kako bi pristupio svim značajkama aplikacije. Aplikacija komunicira s bazom podataka te provjerava podatke, u ovom slučaju Firebase, ne-relacijska baza podataka. Registracijom u aplikaciju korisnik unosi svoje ime i prezime, identifikacijski broj, email i lozinku nakon čega se podaci spremaju u bazu podataka u kolekciju users. Također registracijom korisnika, u bazu podataka dodaje se podatak isAdmin koji je zadan kao user. Prijavom u sustav uneseni podaci (korisničko ime i lozinka) provjeravaju se u bazi. Ukoliko su točni korisnik dobiva pristup svim

značajkama aplikacije. Dobiva mogućnost zakazivanja termina kod stomatologa. Odabirom stomatološke ordinacije, usluge i datuma termina, odabranu podaci se spremaju u podkolekciju orders.

Nakon uspješno zakazanog termina ordinacija prima informacije o terminu te mu dodjeljuje točno vrijeme i datum po potrebi, nakon čega se podaci spremaju u podkolekciju orders. Nadalje, klikom na gumb dodaj, ordinacija upisom potrebnih podataka dodaje novog pacijenta te se podaci uspješno spremaju u bazu. Klikom na gumb lista pacijenata prikazuje se lista svih zakazanih termina s detaljima. Svakom terminu ordinacija može izmijeniti potrebne informacije nakon čega se podaci u bazi mijenjaju. Isto tako klikom na gumb za brisanje, zakazani termin se uspješno briše iz baze podataka sa svim njegovim podacima.

Objektno orijentirano programiranje omogućuje pisanje programa uz pomoć određenih klasa i objekata u stvarnom vremenu. Klasnim dijagramom prikazuje se konstrukcija i vizualizacija orijentiranih sustava. Opisuje strukturu sustava s klasama, njihovim atributima i odnosima među objektima. Klasni dijagram prikazan na slici 8 temelj je za stvaranje i realiziranje sustava.



Slika 8. Klasni dijagram

Dijagram prikazuje način na koji bi se kod trebao implementirati. Prikazuje tipove objekata koji se nalaze u sustavu i odnose među njima. Glavna svrha klasnog dijagrama je izgradnja statičnog

prikaza aplikacije. Klasa je nacrt objekta koji se sastoji od atributa koji opisuju objekt koji se modelira.

Klasa users sastoji se od atributa id, email, identification i isAdmin. Klasa ordination sastoji se od id-a, name, doctor i contact. Nadalje, klasa order sadržava atribute id, ordination, service, date, time i message. Atributi id, name, description, usage i process čine klasu service, te atributi id, name i price čine klasu priceList.

Svaka klasa može biti uključena u jedan ili više odnosa s drugim klasama. Odnosi između dvije klase označavaju da promjena u jednoj može izazvati promjene u drugoj klasi. Odnosi među klasama mogu biti: generalizacija, asocijacija, agregacija ili kompozicija. Generalizacija je odnos između dvije ili više klasa gdje promjena u jedno klasi uzrokuje promjene u drugoj. Asocijacija opisuje vezu između dva ili više objekata te sadrži informaciju i kardinalnosti. Agregacija je specijalna vrsta asocijacije koja označava sadržavanje, dok kompozicija još dodatno označava da komponirani objekt sam nema smisla bez roditelja.

Klasa user i klasa order povezane su kompozicijom što znači da korisnik sadržava narudžbe te da narudžba nema smisla bez korisnika. Ukoliko se korisnik obriše, njegove narudžbe se brišu zajedno s njim. Označene su kardinalnosti koje prikazuju da svaki korisnik može imati nula ili više narudžbi, dok narudžba može imati samo jednog korisnika.

Klasa user povezana je agregacijom s klasom ordination čime je označeno da svaka ordinacija sadržava korisnike. Kardinalnosti pokazuju da svaka ordinacija može imati jednog ili više korisnika, te isto tako korisnik može imati jednu ili više ordinacija s obzirom da se radi o privatnim ordinacijama.

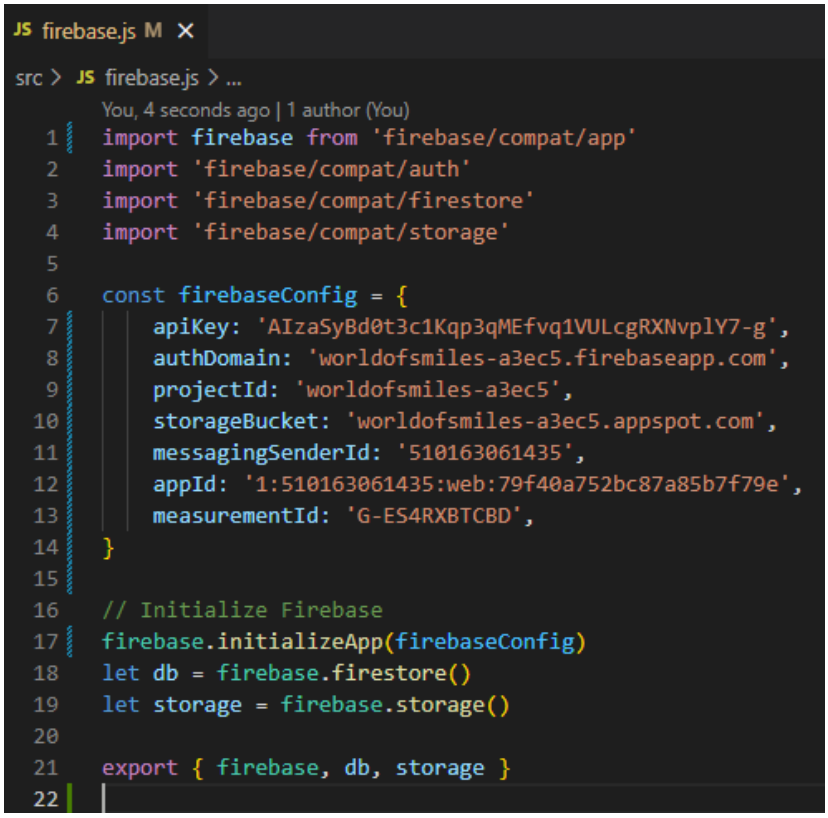
Svaka ordinacija sadržava narudžbe od strane korisnika. Klasa ordination i order povezane su kompozicijom što govori da ukoliko se ordinacija obriše, brišu se i sve njezine narudžbe. Nemaju smisla jedno bez drugog. Postavljene kardinalnosti prikazuju da svaka ordinacija može imati nula ili više narudžbi, dok svaka narudžba ima jednu ordinaciju.

Isto tako, svaka ordinacija sadržava usluge koje nudi. Klase ordination i service povezane su kompozicijom jer svaka ukoliko se ordinacija briše, brišu se i njezine usluge zajedno s njom. Postavljena kompozicija prikazuje da svaka ordinacija može sadržavati nula ili više usluga, dok svaka usluga namijenjena za jednu, određenu ordinaciju.

5.3 Implementacija

Nakon definiranog i objašnjenog klasnog dijagrama omogućen je detaljan uvid u strukturu i funkcionalnosti aplikacije.

Prvo, postavljen je osnovni kostur projekta. Instaliran je Vue.js, dodan Tailwind CSS okvir za izradu dizajna te instaliran i dodan Firebase za bazu podataka. Napravljen je Firebase projekt i registrirana aplikacija u Firebase konzoli, nakon čega je Vue projekt spojen na Firebase projekt, na način prikazanim na slici 9.



Slika 9. Povezivanje Firebase projekta s Vue projektom

Ulaskom u aplikaciju korisnici imaju mogućnost registracije i prijave u aplikaciju ili korištenje iste s ograničenim sadržajem. Napravljena je komponenta Signup.vue u kojoj je definirana metoda koja učitava unesene podatke te poziva Firebase metodu za registraciju createUserWithEmailAndPassword. Definirane su Vue varijable za ime i prezime, identifikacijski broj, email i dvije lozinke koje su povezane sa v-model na odgovarajuća input polja. Zatim je definirana funkcija signup() u sekciji methods, prikazana na slici 10. Ukoliko je kreiran novi račun korisnik se automatski prijavljuje te se uneseni podaci spremaju u bazu u kolekciju users.

Na sličan način napravljena je Login.vue komponenta. Definirane su varijable za email i lozinku koje su povezane sa v-model na input polja. Unosom podataka korisnička adresa i lozinka prosljeđuju se na signInWithEmailAndPassword. Funkcija login() dohvaća podatke o korisniku te provjera da li je korisnik admin ili običan user kako bi ga preusmjerila na određenu rutu (indexScreen ili adminScreen). Funkcija login() prikazana je na slici 11.

Kako bi pratili stanje autentifikacije korisnika, razlikovali kada je korisnik ulogiran i prema tome prikazivali ili sakrivali određene dijelove aplikacije potrebno je postaviti takozvani promatrač Auth objekt.

Kada se korisnik prvi put prijavi, kreira se novi korisnički račun i povezuje se s vjerodajnicama (eng. credentials), to jest s podacima koje je korisnik upisao tijekom autorizacije. Pohranjeni podaci koriste se za identifikaciju korisnika. Objektom Auth dobivaju se osnovne informacije o trenutnom korisniku. Način korištenja objekta Auth prikazan je na slici 12.

Kao što je prije spomenuto, korišten je Vuex za reaktivno spremanje varijabli. Napravljen je file store.js u kojem dodajemo varijable kako bi se učinkovito ažurirale ako se stanje state-a promijeni. Dodane su varijable currentUid, currentName i currentUserEmail inicijalizirane na null, pomoću kojih razlikujemo postoji li trenutni korisnik.

```
methods: {
    signup() {
        const auth = getAuth()
        createUserWithEmailAndPassword(auth, this.username, this.password)
            .then((userCredential) => {
                const user = userCredential.user
                const uid = user.uid
                store.state.currentName = this.fullName
                console.log(user)

                // add new user in document with uid from auth
                setDoc(doc(db, 'users', uid), {
                    fullName: this.fullName,
                    identification: this.identification,
                    email: user.email,
                    isAdmin: 'user',
                })
                this.isLoading = false
                console.log('Reg Success! Email: ' + user.email)
            })
            .then(() => {
                // for updating profile on signup
                updateProfile(auth.currentUser, {
                    displayName: this.fullName,
                })
                this.$router.replace({ path: '/' })
            })
            .catch((e) => {
                console.error(e.message)
                alert(e.message)
                store.currentName = null
            })
        this.isLoading = true
    },
},
```

Slika 10. Signup() funkcija za registraciju korisnika

Slika 11. login() funkcija za prijavu korisnika

```
const auth = getAuth()
//monitoring the user's login status
onAuthStateChanged(auth, (user) => {
  if (user) {
        const isAdmin = localStorage.getItem('isAdmin')
        const ordinationId = localStorage.getItem('ordinationId')
        localStorage.setItem('checkLogedUser', user.email)
        localStorage.setItem('currentUid', user.uid)
        store.state.currentUserEmail = user.email
        store.state.currentName = user.displayName
        store.state.currentUid = user.uid
        store.state.adminOrdiantionId = ordinationId
        console.log(store.state.adminOrdiantionId)
        if (isAdmin === 'admin') {
            router.push({ path: '/adminscreen' })
            }
    } else {
        store.state.currentUserEmail = null
        store.state.currentName = null
        store.state.userRole = null
        localStorage.clear()
    }
})
```

Slika 12. Objekt Auth za dobivanje informacija o trenutnom korisniku

Ukoliko korisnik aplikacije zaboravi lozinku, klikom na gumb Forgot password otvara se ruta koja prikazuje obrazac za zaboravljenu lozinku. Metodom sendPasswordResetEmail provjerava se postoji li adresa e-pošte u bazi te šalje korisniku e-poštu za ponovno postavljanje lozinke. Funkcija sendEmail() prikazana je na slici 13.

```
65      methods: {
66          sendEmail() {
67              const auth = getAuth()
68              sendPasswordResetEmail(auth, this.email)
69                  .then(() => {
70                      alert('Email sent!')
71                      this.$router.push({ path: '/login' })
72                  })
73                  .catch((error) => {
74                      this.error = error.message
75                      alert('Please type in a valid email address.')
76                  })
77          },
78      },
```

Slika 13. Funkcija sendEmail() za ponovno postavljanje lozinke

Nakon omogućene registracije i prijave korisnika, napravljena je funkcija za odjavu. Kako bi se korisnik odjavio iz aplikacije koristi se Firebase metoda signOut prikazana na slici 14.

```
339          signout() {
340              signOut(auth)
341                  .then(() => {
342                      store.state.userRole = null
343                      this.$router.replace({ path: '/' })
344                      localStorage.clear()
345                      this.isLoading = false
346                  })
347                  .catch((error) => {
348                      console.log(error)
349                  })
350              this.isLoading = true
351          },
```

Slika 14. signout() funkcija za odjavu korisnika

Početna stranica aplikacije pruža informacije o samoj aplikaciji. Prikazuje listu ordinacija koje pruža. Ordinacije su prikazane u slider-u napravljenim koristeći splide.js, fleksibilne biblioteke napisane u TypeScriptu. Na početnoj stranici, indexScreen.vue napravljena je funkcija koja dohvaća sve ordinacije iz kolekcije ordinations. Za dohvat više dokumenata iz kolekcije koristi se getDocs() upit. Izvršava upit i vraća rezultate kao QuerySnapshot. QuerySnapshot sadrži nula ili više DocumentSnapshot objekata koji predstavljaju rezultate upita. Funkcija getOrdinations() prikazana je na slici 15.

```
141     methods: {
142         async getOrdiantions() {
143             const querySnapshot = await getDocs(collection(db, `ordinations`))
144             querySnapshot.forEach((doc) => {
145                 this.ordinations.push(doc.data())
146                 // console.log(doc.data())
147             })
148         },
```

Slika 15. Funckija getOrdinations() za dohvat ordinacija iz baze podataka

Slider prikazan na indexScreen-u, definiran je u komponenti OrdinationSlider.vue. Proslijeđeni su joj dohvaćeni podaci o ordinacijama putem props-a, ključne riječi koja označava svojstva za prijenos podataka. Klikom na gumb slider-a pojedine ordinacije prikazuje se ordinationScreen koji prikazuje informacije i usluge pojedine ordinacije. Dinamičkom rutom proslijeđeni su parametri na ordinationScreen.vue Kada su uspješno proslijeđeni i prikazani podaci za svaku ordinaciju napravljena je funkcija za dohvat usluga putem id-a odabrane ordinacije, prikazana na slici 16. Id ordinacije spremljen je u store.js kako bi se ažurirao promjenom stanja varijable. Funkcija koristi onSnapshot() metodu koja stvara snimku dokumenata s trenutnim sadržajem jednog dokumenta te svaki put kada se sadržaj promijeni, drugi poziv ažurira dokumente.

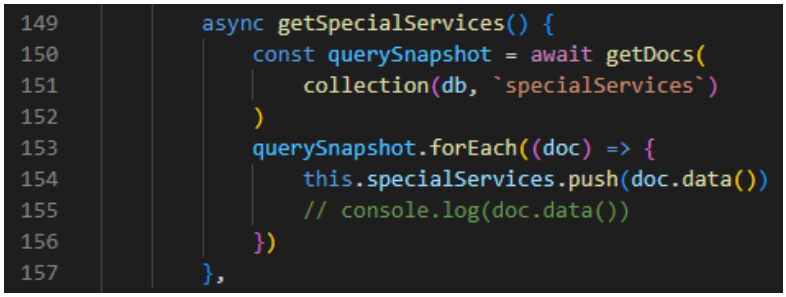
```
58  methods: {
59      getServices() {
60          store.state.ordinationIdSlider = this.id
61          const q = query(collection(db, `ordinations/${store.state.ordinationIdSlider}/services`))
62          onSnapshot(q, (snapshot) => {
63              snapshot.docChanges().forEach((change) => {
64                  this.services.push(change.doc.data())
65                  console.log(this.services)
66                  this.isLoading = false
67              })
68          })
69          this.isLoading = true
70      },
71      goBack() {
72        return this.$router.go(-1)
73      },
74  },
```

Slika 16. Funkcija getServices() za dohvat usluga ordinacije

Klikom na gumb Price list prikazuje se komponenta s cjenikom navedenih usluga odabrane ordinacije. U komponenti PriceList.vue napravljena je funkcija za dohvat cjenika odabrane ordinacije preko id-a koji je spremljen u store.js. Funkcija je prikazana na slici 17.

Slika 17. Funkcija getPriceList() za dohvat cjenika ordinacije

Početna stranica također prikazuje specijalne usluge koje sve ordinacije pružaju. Dohvaćane su funkcijom getSpecialServices() prikazanom na slici 18.



Slika 18. Funkcija getSpecialServices() za dohvat specijalnih usluga ordinacija

Svaki korisnik stvara svoju narudžbu. Na početnoj stranici nalazi se gumb za zakazivanje termina. Ukoliko korisnik nije prijavljen u aplikaciju nema mogućnost rezervacije termina. Klikom na gumb Book an appointemnt ruta ga vodi na login screen. Varijablom currentName iz store.js provjeravamo da li je korisnik prijavljen. Ako je korisnik prijavljen kolikom na gumb ruta ga vodi na screen chooseOrdination. Chooseordination.vue prikazuje popis ponuđenih privatnih ordinacija. Select za biranje ordinacije napravljen je pomoću Headless UI biblioteke koja pruža komponente korisničkog sučelja, dizajnirane za integraciju s Tailwind CSS-om. Ordinacije su dohvaćene iz kolekcije ordinations isto kao i na početnoj stranici. Funkcija getOrdinations() prikazana je na slici 19.

Slika 19. Funkcija getOrdinations() za dohvat ordinacija

Definirana je varijabla selected povezana je s v-model sa RadioGroup komponentom koja predstavlja odabranu ordinaciju. Podaci odabrane ordinacije spremaju se u store.js funkcijom selectOrdination() prikazanom na slici 20.

```
152     async selectOrdination() {
153         store.state.selectedOrdination = this.selected.name
154         store.state.selectedOrdinationId = this.selected.id
155         // console.log(store.state.selectedOrdination, store.state.selectedOrdinationId)
156     },
```

Slika 20. Funkcija selectOrdination() za spremanje odabrane ordinacije u store.js

Nadalje, korisnik odabire željenu uslugu. U podkolekciji services nalaze se usluge s detaljnim informacijama koje ordinacija nudi. Podatke iz baze dohvaćamo funkcijom getServices() preko id-a odabrane ordinacije iz store.js-a koji je spremljen u prijašnjem koraku. Funkcija koristi onSnapshot() metodu koja ažurira dohvaćene podatke u stvarnom vremenu. Odabir usluga napravljen je jednako kao i odabir ordinacija. Odabirom usluge podaci se spremaju u store.js u varijablu selectedService funkcijom selectService(). Funkcije getService() i selectService() prikazane su na slici 21.

```
148     methods: {
149         async getServices() {
150             const q = query(
151                 collection(
152                     db,
153                     `ordinations/${store.state.selectedOrdinationId}/services`
154                 )
155             )
156             onSnapshot(q, (snapshot) => {
157                 snapshot.docChanges().forEach((change) => {
158                     this.services.push(change.doc.data())
159                     // console.log(this.services)
160                 })
161             })
162         },
163         async selectService() {
164             store.state.selectedService = this.selected.name
165             // console.log(store.selectedService)      You, 3 days ago • added
166         },
```

Slika 21. Funkcija getService() i funkcija selectService()

Posljednji korak zakazivanja termina je odabir datuma. Kalendar je napravljen uz pomoć Datepicker komponente, objašnjene u odjeljku prije. Definirana je varijabla date i povezana s v-model na komponentu Datepicker. U komponenti su određeni props-i kojima je definiran format datuma, onemogućen odabir datuma nedjeljom te ograničeno rezerviranje termina najviše dva mjeseca unaprijed. Klikom na željeni datum sprema se u varijablu date. Funkcijom setOrder() spremamo podatke zakazanog termina u podkolekciju orders preko id-a odabrane ordinacije koja je spremljena u store.js na prvom koraku procesa zakazivanja termina. Potrebne

podatke o terminu i korisniku dohvaćamo iz store.js-a te ih spremamo u bazu (ime korisnika, identifikacijski broj, email., odabrana ordinacija, usluga, datum, poruka). Također korisnik imam mogućnost pisanja dodane poruke kao napomene ukoliko je potrebno. Funkcija za spremanje podataka zakazanog termina prikazana je na slici 22.

```
async setOrder() {
    store.state.selectedDate = this.getSelectedDate(this.date)
    // console.log(store.state.selectedDate)
    await setDoc(
        doc(
            db,
            `ordinations/${store.state.selectedOrdinationId}/orders/${store.state.currentName}`
        ),
        {
            title: store.state.currentName,
            service: store.state.selectedService,
            start: this.getSelectedDate(this.date),
            time: '',
            email: store.state.currentUserEmail,
            identification: store.state.identification,
            message: this.message,
            userid: store.state.currentUid,
            ordination: store.state.selectedOrdination,
        }
    )
```

Slika 22. Funkcija setOrder()

Nakon što korisnik uspješno rezervira željeni termin, ordinacije dobije obavijest o novom terminu. Dakle, ordinacije imaju status admin-a. Prijavom u sustav prikazuju se informacije o ordinaciji. Ordinacija ima mogućnost pregleda svih zakazanih termina, dodavanje novih, brisanje ili mijenjanje postojećih. Funkcijom getOrdinationInfo() prikazanoj na slici 23, dohvaćaju se podaci o prijavljenoj ordinaciji.

```
76          async getOrdinationInfo() {
77              const q = query(
78                  collection(db, `ordinations`),
79                  where('id', '==', store.state.adminOrdiantionId)
80              )
81              onSnapshot(q, (querySnapshot) => {
82                  querySnapshot.forEach((doc) => {
83                      this.data.push(doc.data())
84                      console.log(this.data)
85                  })
86              })
87          },
```

Slika 23. Funkcija getOrdinationInfo()

Klikom na gumb new orders prikazuju se nove rezervacije pacijenata s detaljima kojima je potrebno odrediti točno vrijeme termina. Funkcijom getOrders() prikazanom na slici 24, dohvaćamo termine zakazane od strane korisnika kojima još nije određeno točno vrijeme. Korištena je onSnapshot() metoda kako bi se podaci ažurirali promijenim podataka u bazi.

```
80      methods: {
81          async getOrders() {
82              const q = query(
83                  collection(
84                      db,
85                      `ordinations/${store.state.adminOrdiantionId}/orders`
86                  ),
87                  where('time', '==', '')
88              )
89              onSnapshot(q, (querySnapshot) => {
90                  querySnapshot.forEach((doc) => {
91                      this.orders.push(doc.data())
92                      //store.state.currentAdminName = this.ordinations
93                  })
94                  if (this.orders.length === 0) {
95                      this.error = 'No new orders'
96                  }
97              })
98          },
99      },
```

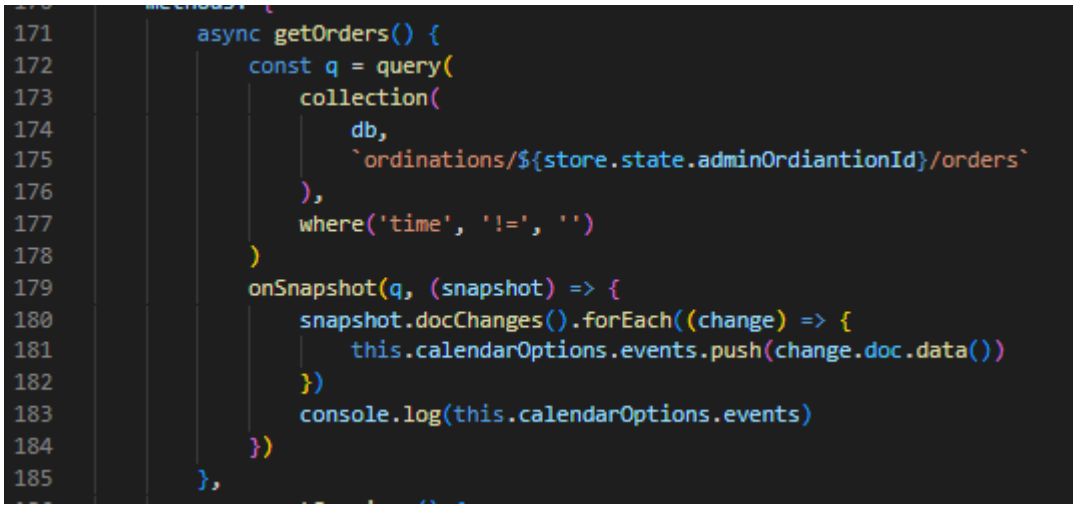
Slika 24. Funkcija getOrders() za dohvat novih termina

Klikom na gumb edit ordinacija određuje točno vrijeme termina. Ukoliko je odabrani datum od strane korisnika potpuno zauzet ordinacija mu dodjeljuje prvi slobodan termin. Funkcijom editOrder() sve podatke spremamo u podkolekciju orders preko id-a trenutne ordinacije. Nakon što su podaci spremljeni u bazu podataka korisnik dobiva potvrdu zakazanog termina sa svim detaljima na email. Slanje email-a napravljen je pomoću EmailJS-a. Napravljen je predložak e-pošte, dodani su parametri serviceID preko kojeg treba poslati e-poštu, templateID (id predloška pošte) i templateParams (parametri predloška). Metodom emailjs.send vraćamo Promise gdje je odgovor objekt koji sadržava status i tekst. Funkcija za spremanje podataka u bazu i slanje email-a prikazana je na slici 25.

```
async editOrder() {
    await setDoc(
        doc(
            db,
            `ordinations/${store.state.adminOrdiantionId}/orders/${this.title}`
        ),
        {
            title: this.title,
            service: this.service,
            start: this.getSelectedDate(this.date) + ' ' + this.time,
            time: this.time,
            email: this.email,
            message: this.message,
            identification: this.identification,
            ordination: this.ordination,
            userid: this.id,
        }
    )
    const templateParams = {
        currentAdminName: store.state.currentAdminName,
        title: this.title,
        service: this.service,
        start: this.start,
        time: this.time,
        email: this.email,
    }
    await emailjs
        .send(
            'service_wqgif9z',
            'template_myfiyd7',
            templateParams,
            'bHpqVlWLvmbQh6PA1'
        )
        .then(
            (res) => {
                console.log(
                    'The appointment is successfully scheduled!',
                    res.status,
                    res.text
                )
            },
            (e) => {
                alert('Failed! E-mail confirm not sent!')
            }
        )
},
},
```

Slika 25. Funkcija editOrder()

Svaka ordinacija ima kalendar s rasporedom zakazanih termina prikazanih na appointemntslist ruti. Kalendar je napravljen koristeći FullCalendar. Funkcijom getOrders(), prikazanoj na slici 26, dohvaćaju se zakazani termini iz podkolekcije orders kojima je određeno točno vrijeme termina. Funkcija također koristi metodu onSnapshot() kako bi se podaci ažurirali ukoliko se detalji termina promijene ili obrišu. Dohvaćeni podaci spremaju se u varijablu events kako bi se prikazivali kao događaji na kalendaru s pregledom po mjesecu, tjednu ili danu.



Slika 26. Funkcija getOrders() za dohvat zakazanih termina s točno određenim vremenom

Također, ordinacija ima mogućnost dodavanja termina ručno, ukoliko pacijent nazove telefonski. Klikom na gumb add otvara se komponenta za unos podataka. Definirane su varijable (ime i prezime, identifikacijski broj, usluga, datum i vrijeme) koje su povezane v-modelom na određena input polja. Podaci se spremaju u bazu podataka funkcijom addNewOrder() prikazanom na slici 27.

Slika 27. Funkcija addNewOrder()

Klikom na gumb Patient list, ordinacije imaju pregled liste pacijenata s detaljima njihovih zakazanih termina. Pacijenti kojima je točno određen termin, dohvaćeni su iz kolekcije orders funkcijom getPatients() prikazanoj na slici 28. S obzirom da pacijent može biti u nemogućnosti doći na zakazani termin omogućeno je mijenjanje i brisanje termina klikom na gumb details. Putem parametara prosljeđuje se id i ime pacijenta na rutu patientdetails kako bi dohvatili sve

podatke odabranog pacijenta. Funkcija za dohvat pacijenta i njegovih detalja prikazana je na slici 29.

```
97      methods: {
98          async getPatients() {
99              const q = query(
100                 collection(
101                     db,
102                     `ordinations/${store.state.adminOrdiantionId}/orders`
103                 ),
104                 where('time', '!=', '')
105             )
106             onSnapshot(q, (snapshot) => {
107                 snapshot.docChanges().forEach((change) => {
108                     this.patients.push(change.doc.data())
109                 })
110             })
111             console.log(this.patients)
112         },
```

Slika 28. Funkcija getPatients()

```
201     async getPatient() {
202         const docRef = doc(
203             db,
204             `/ordinations/${store.state.adminOrdiantionId}/orders/${this.title}`
205         )
206         const docSnap = await getDoc(docRef)
207         this.patient.push(docSnap.data())
208
209         console.log(this.patient)
210     },
```

Slika 29. Funkcija getPatient()

Promjenom zakazanog termina, podaci se spremaju u bazu podataka, tj. mijenjaju već postojeće podatke. Zato je u funkciji editOrder() korištena metoda setDoc() koja upisuje podatke u podkolekciju orders. Funkcija za promjenu detalja zakazanog termina prikazana je na slici 30.

Također, klikom na gumb delete, zakazani termin uspješno se briše iz baze podataka iz podkolekcije orders, Funkcija za brisanje termina prikazana je na slici 31.

```
247         async editOrder() {
248             await setDoc(
249                 doc(
250                     db,
251                     `ordinations/${store.state.adminOrdiantionId}/orders/${this.title}`
252                 ),
253                 {
254                     title: this.title,
255                     service: this.service,
256                     start: this.getSelectedDate(this.date) + ' ' + this.time,
257                     time: this.time,
258                     email: this.email,
259                     identification: this.identification,
260                     message: this.message,
261                     ordination: this.ordination,
262                     userid: this.userid,
263                 }
264             )
265             alert('Order edit successful!')
266             this.$router.push({ name: 'patientList' })
267         },
```

Slika 30. Funkcija editOrder()

```
211         async deletePatient() {
212             await deleteDoc(
213                 doc(
214                     db,
215                     `ordinations/${store.state.adminOrdiantionId}/orders/${this.title}`
216                 )
217             )
218             alert('Order deleted successful!')
219             this.$router.push({ name: 'patientList' })
220         },
```

Slika 31. Funkcija deletePatient() za brisanje zakazanog termina pacijenta

Kada je termin pacijenta uspješno zakazan s određenim točnim vremenom, korisnik klikom na My orders ima pregled svojih zakazanih termina. Termini su dohvaćeni funkcijom getOrders() prikazanom na slici 32.

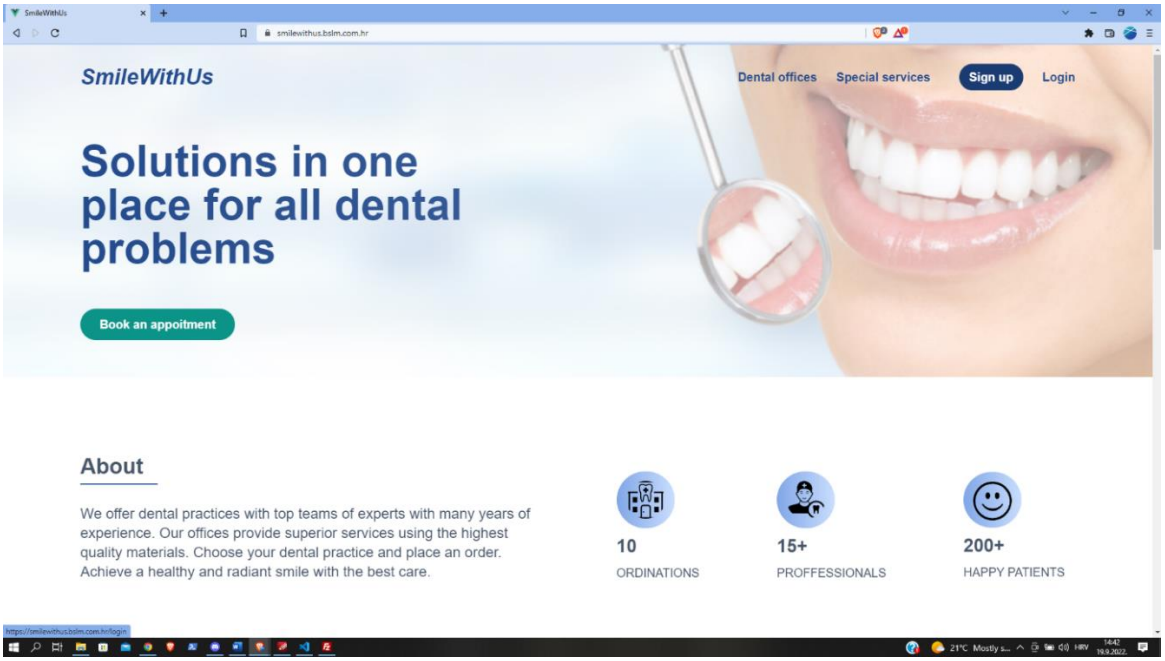
```
88          async getOrders() {
89              const q = query(
90                  collection(db, `users/${store.state.currentUid}/orders`),
91                  where('time', '!=', '')
92              )
93              onSnapshot(q, (querySnapshot) => {
94                  querySnapshot.forEach((doc) => {
95                      this.orders.push(doc.data())
96                      //store.state.currentAdminName = this.ordinations
97                  })
98                  if (this.orders.length === 0) {
99                      this.error = 'No new orders'
100                 }
101             })
102         },
```

Slika 32. Funkcija getOrders() za dohvat zakazanih termina korisnika

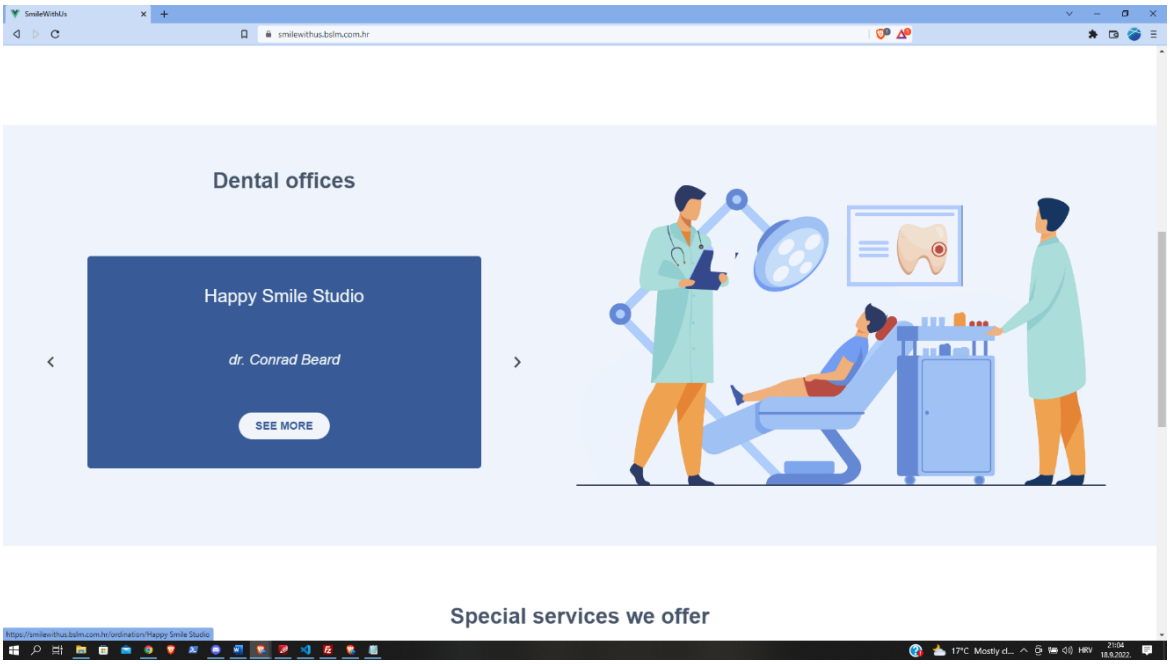
5.1 Snimke zaslona

5.1.1 Početna stranica

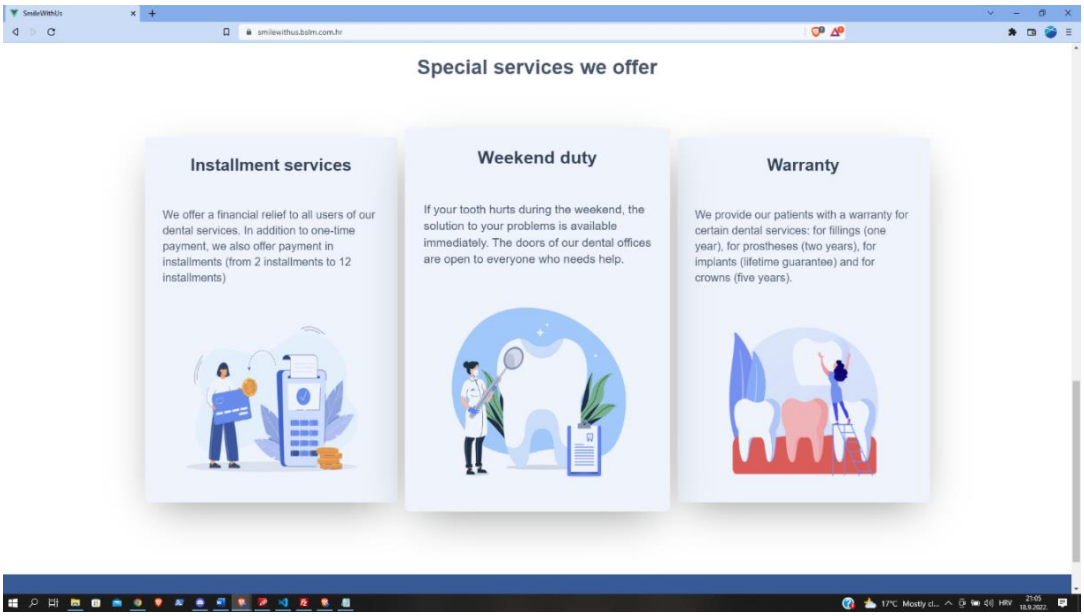
Ulaskom u aplikaciju prikazuje se početna stranica koja nudi informacije o aplikaciji prikazana na Slika *33*. Prikazuje popis stomatoloških ordinacija koje nudi prikazanih u slider-u što je prikazano na slici 34. Također nudi opis specijalnih usluga koje pruža poput plaćanja na rate, dežurstva i garancije. Specijalne usluge prikazane su na slici 35.



Slika 33. Početna stranica

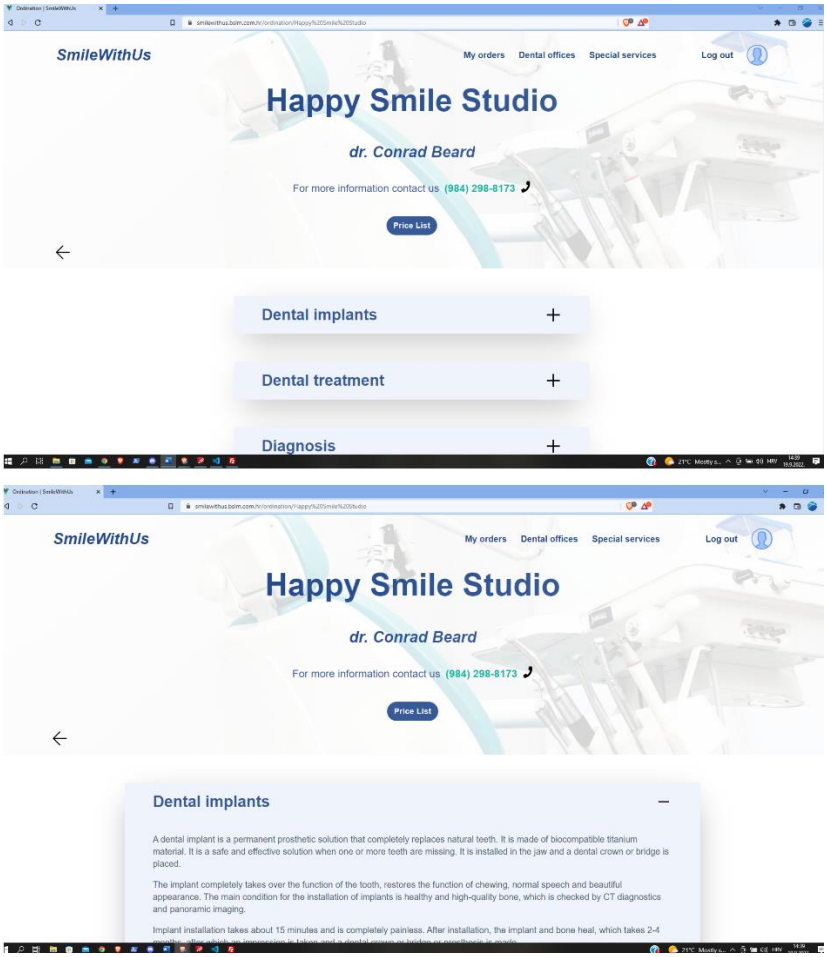


Slika 34. Početna stranica – stomatološke ordinacije



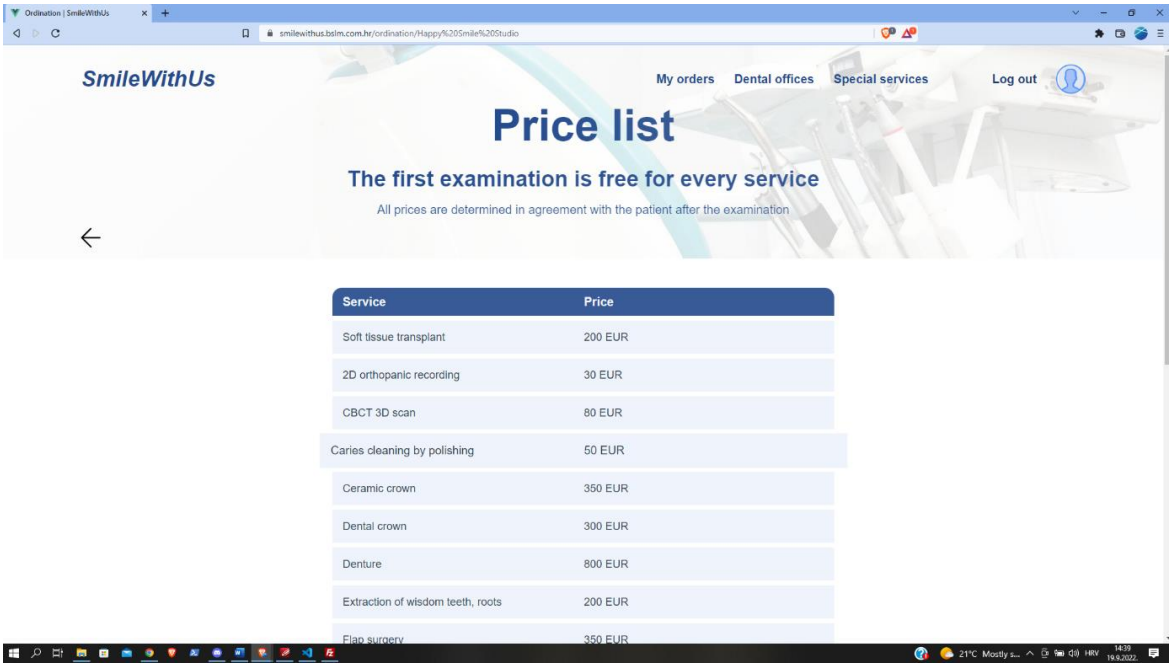
Slika 35. Početna stanica – specijalne usluge

Klikom na gumb slider-a See more, prikazuju se detaljne informacije odabrane ordinacije. Pruža pregled usluga koje nudi prikazanih na slici 36.



Slika 36. Usluge ordinacije

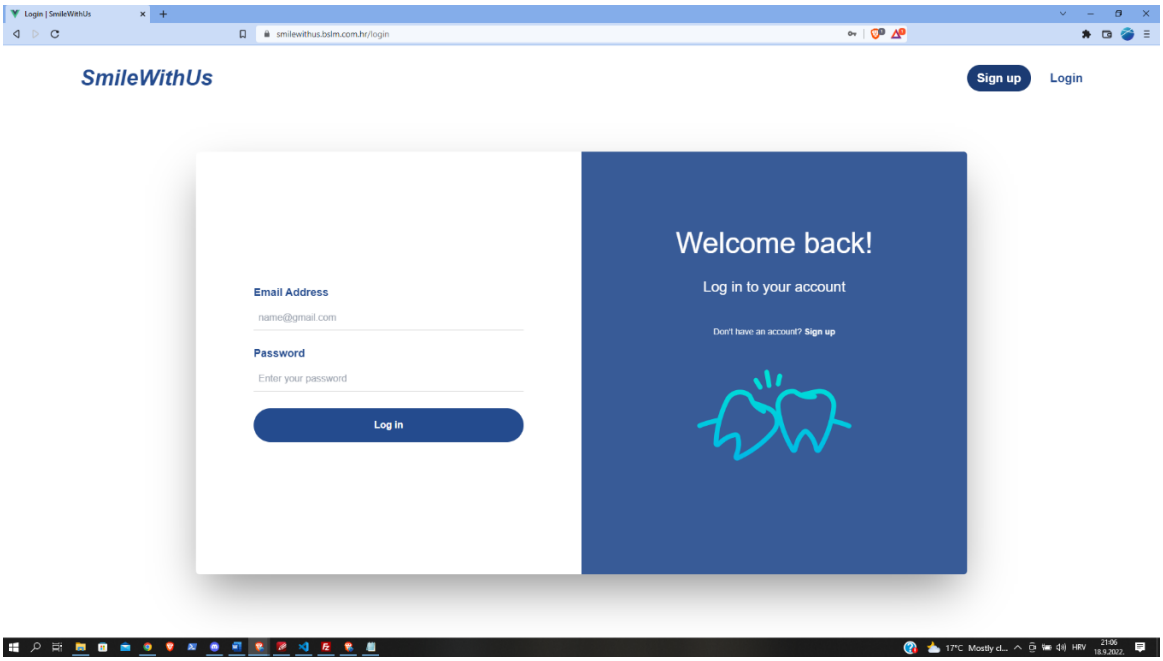
Isto tako, klikom na gumb Price list prikazuje se cjenik svake usluge koje odabrana ordinacija nudi. Cjenik je prikazan na slici 37.



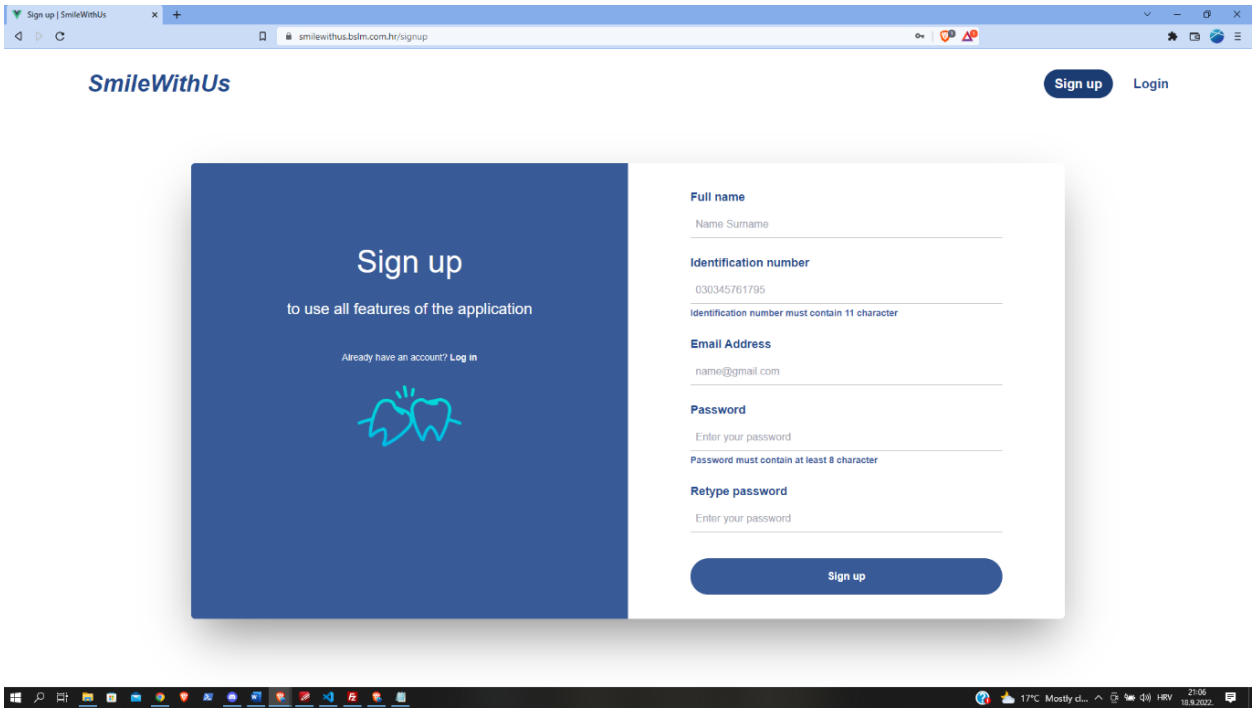
Slika 37. Cjenik ordinacije

5.1.2 Prijava i registracija korisnika

Kako bi korisnik imao pristup svim značajkama aplikacije mora se registrirati ili prijaviti u aplikaciju. Registracija korisnika prikazana je na slici 38. Registrirani korisnici prijavljuju se u aplikaciju kako bi rezervirali termin stomatološkog pregleda, što je prikazano na slici 39.

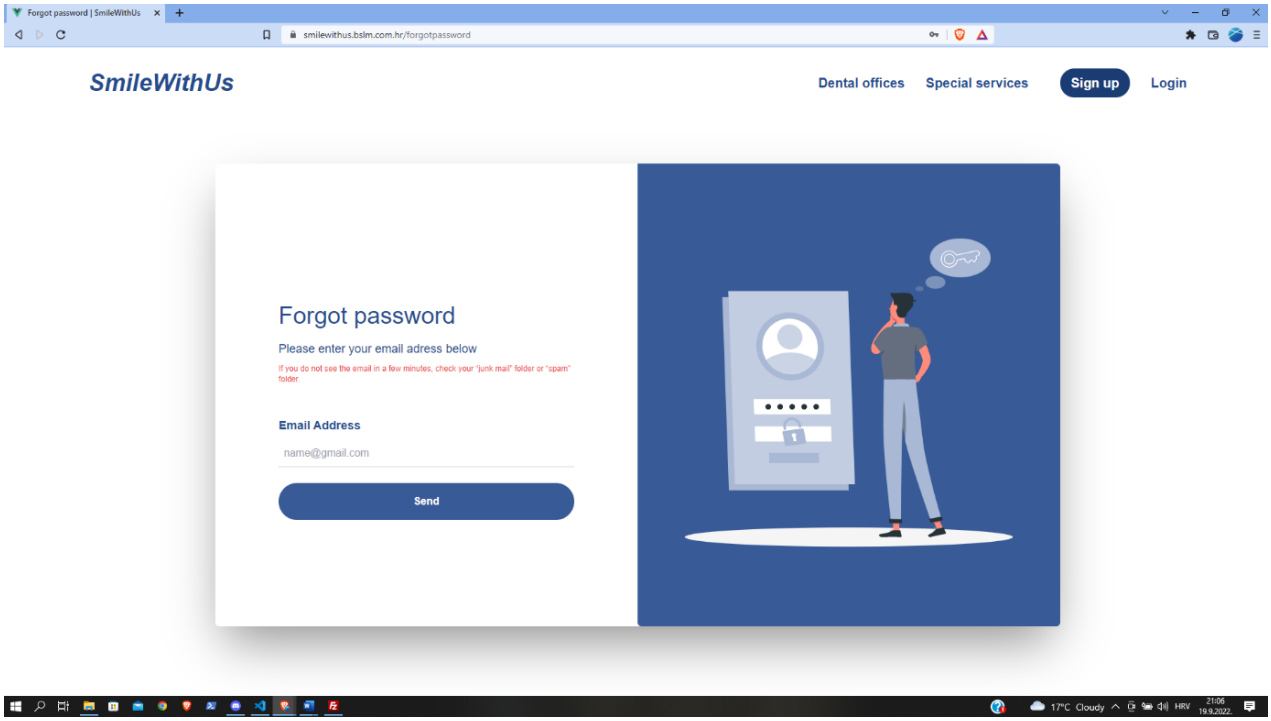


Slika 38. Prijava korisnika



Slika 39. Registracija korisnika

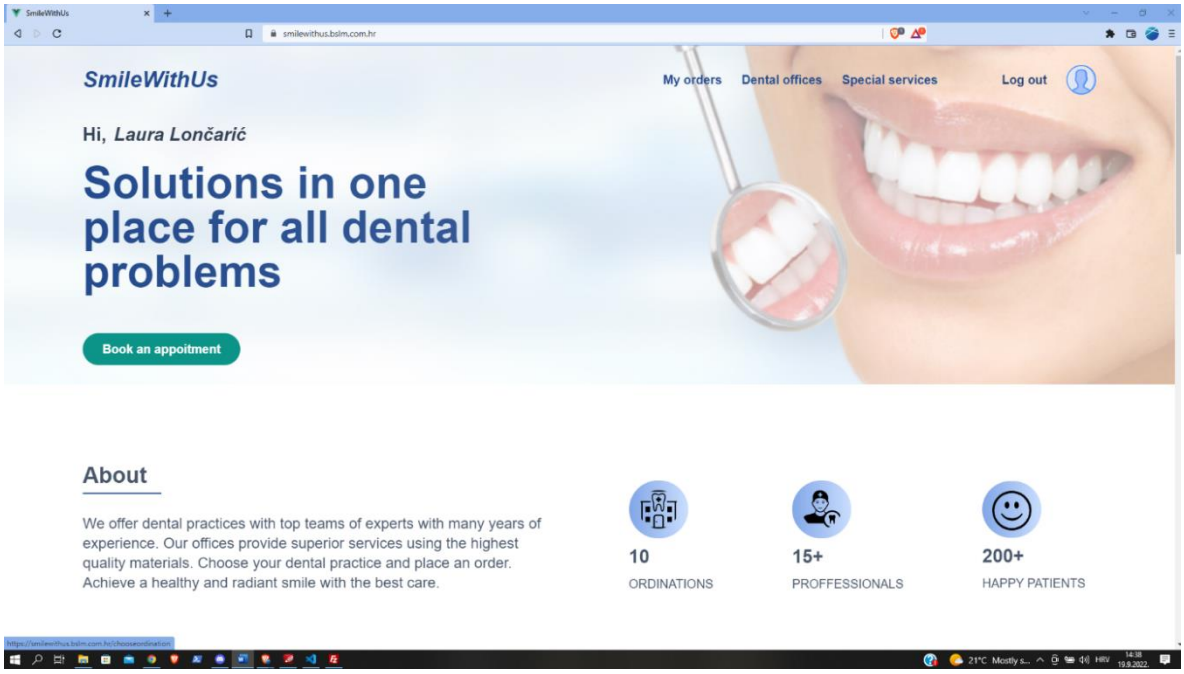
Ukoliko korisnik zaboravi svoju lozinku klikom na gumb Forgot password upisuje svoj email kako bi dobio povratni email za ponovno postavljanje lozinke što je prikazano na slici 40.



Slika 40. Zaboravljena lozinka

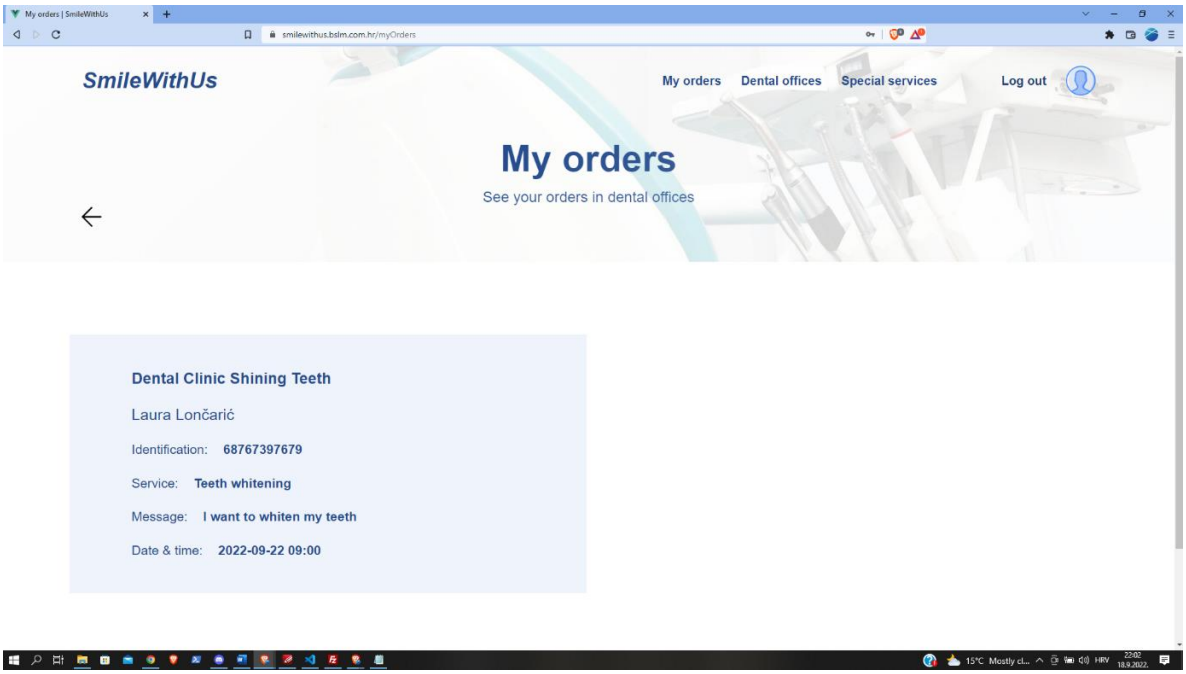
5.1.3 Korisničko sučelje i proces zakazivanja termina

Prijavom u sustav prikazuje se početna stranica s dodatnim značajkama prikazanim na slici 41. Pruža mogućnost zakazivanja termina te pregled zakazanih narudžbi.



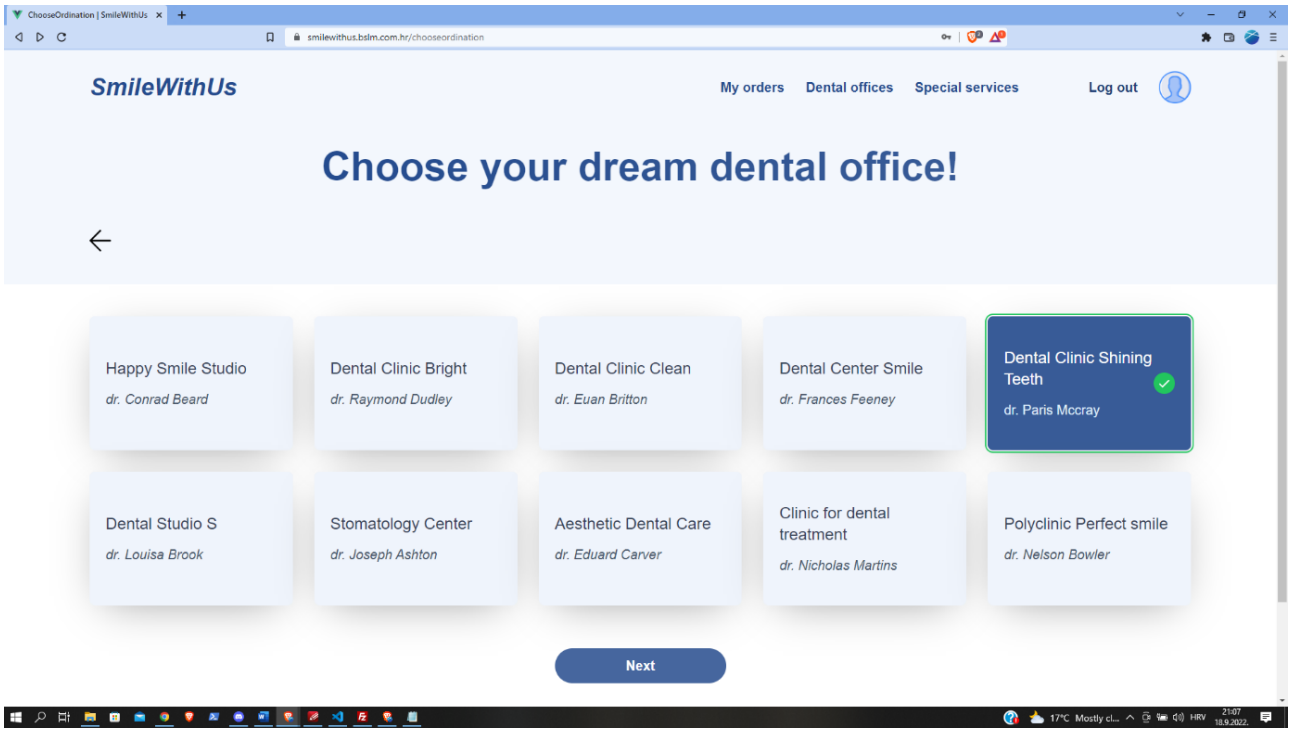
Slika 41. Početna stranica prijavljenog korisnika

Klikom na gumb My orders prikazuju se zakazane narudžbe korisnika sa svim detaljima termina. Pregled zakazanih narudžbi korisnika prikazan je na slici 42.

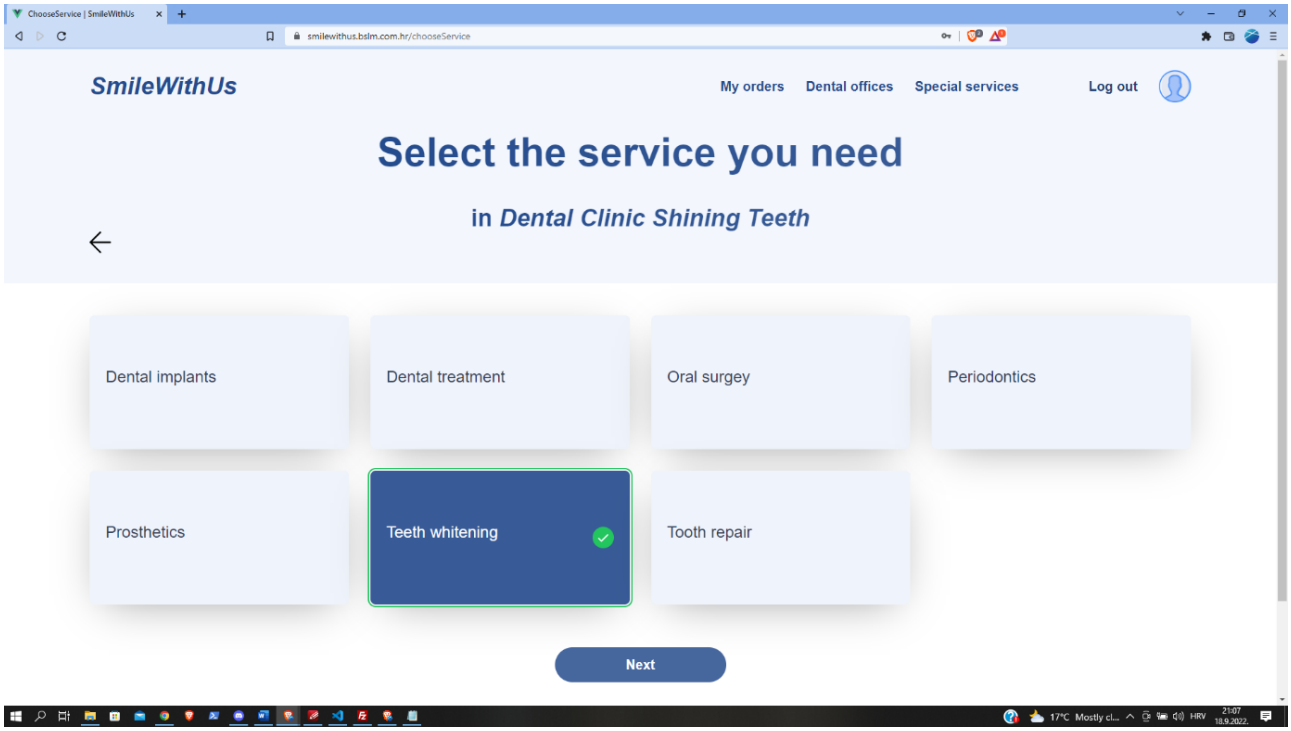


Slika 42. Korisnikove narudžbe

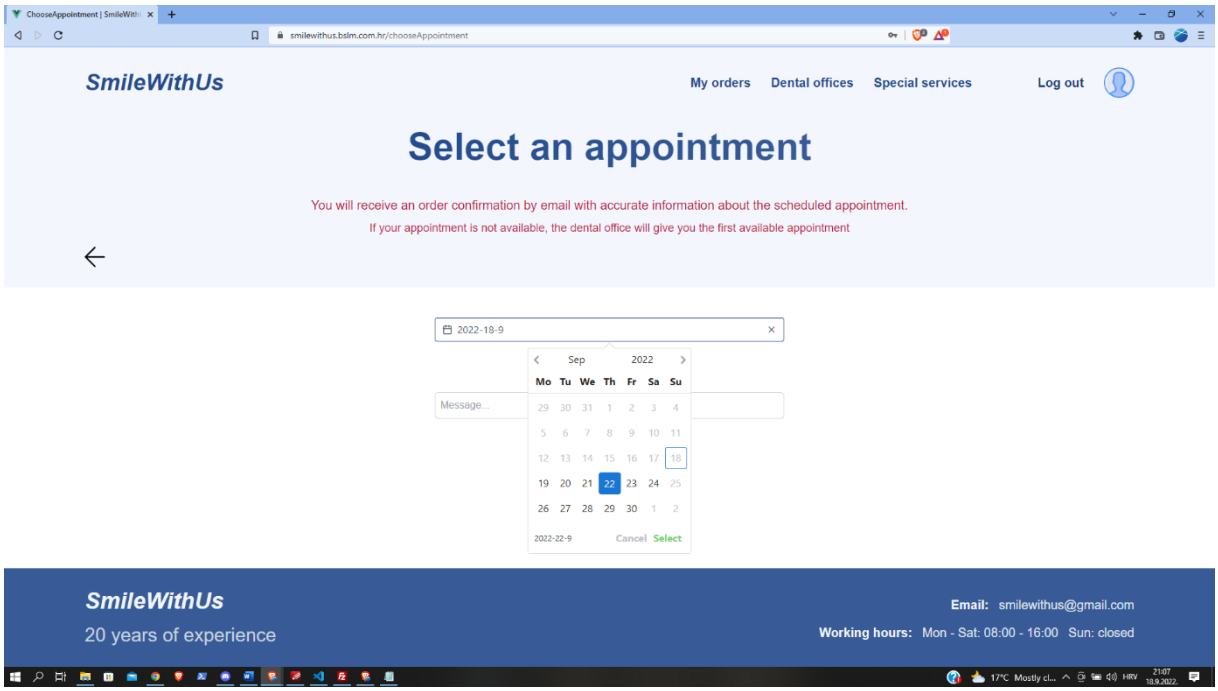
Proces zakazivanja termina sastoji se od odabira ponuđenih ordinacija što je prikazano na slici 43. Zatim korisnik bira željenu uslugu koju odabrana ordinacija pruža. Odabir usluga prikazan je na slici 44. Na kraju korisnik odabire željeni datum termina te ostavlja dodatnu poruku kao napomenu prema želji što je prikazano na slici 45.



Slika 43. Odabir ordinacije



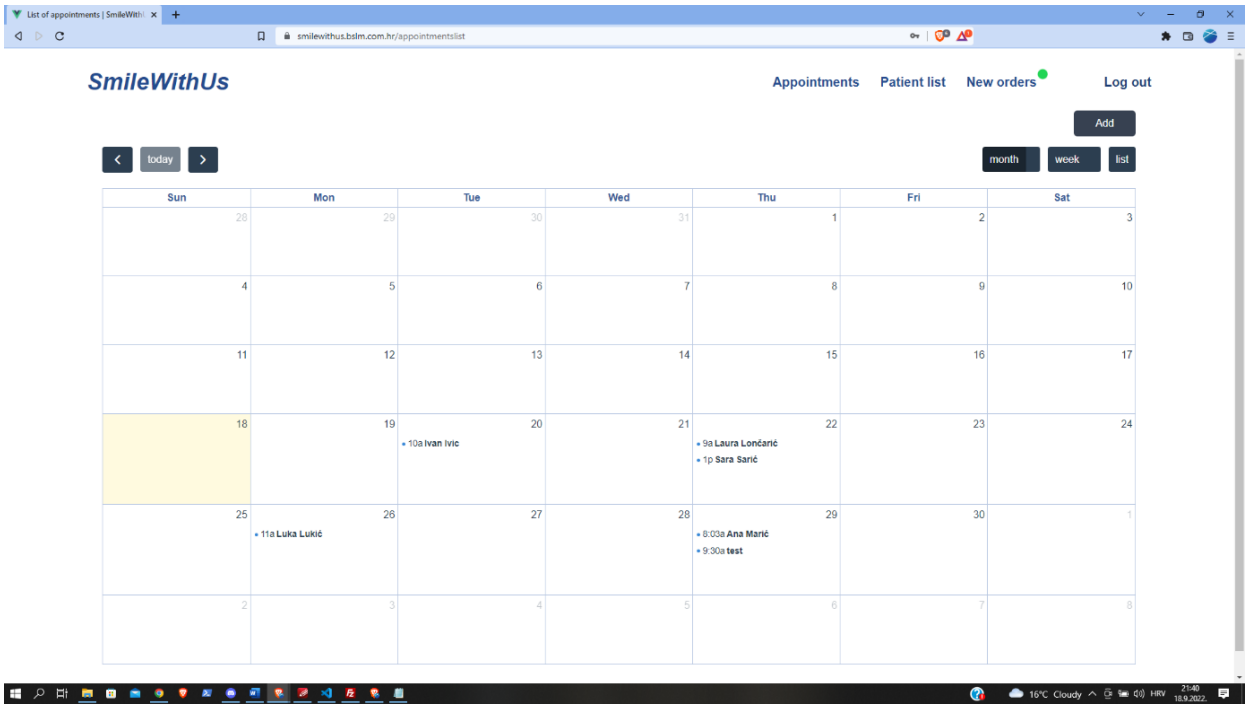
Slika 44. Odabir usluge



Slika 45. Odabir datuma termina

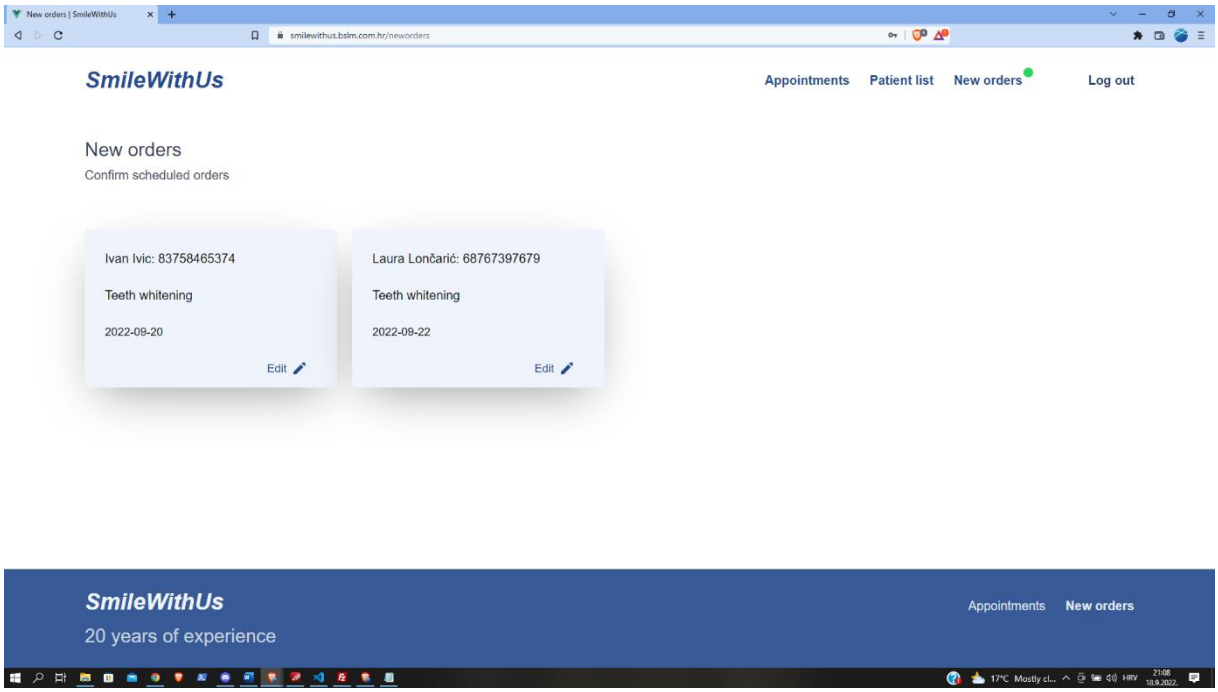
5.1.4 Sučelje ordinacija

Sučelje ordinacija pruža kalendar s rasporedom zakazanih termina koji se može pregledavati po mjesecu, tjednu i danu što je prikazano na slici 46.

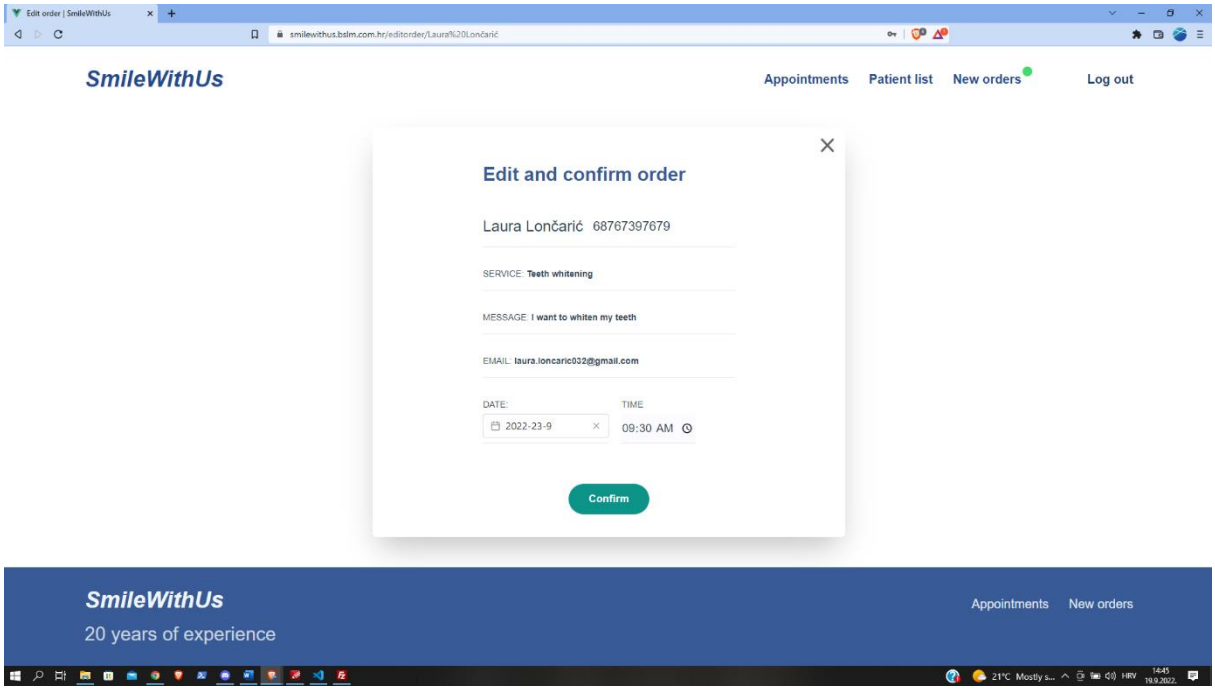


Slika 46. Kalendar s rasporedom zakazanih termina

Ukoliko postoji nova narudžba ordinacija prima obavijest na gumbu New orders. Prikazuje nove narudžbe za stomatološki pregled prikazane na slici 47, kojima zatim određuje točan datum i vrijeme termina što je prikazano na slici 48.

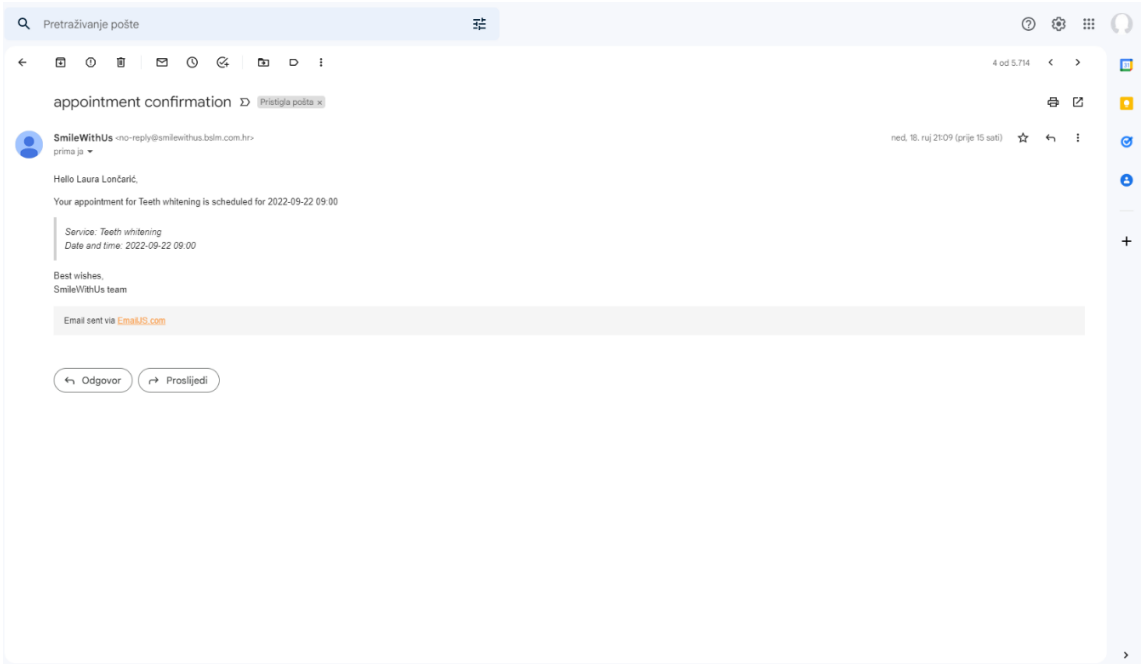


Slika 47. Novi termini korisnika



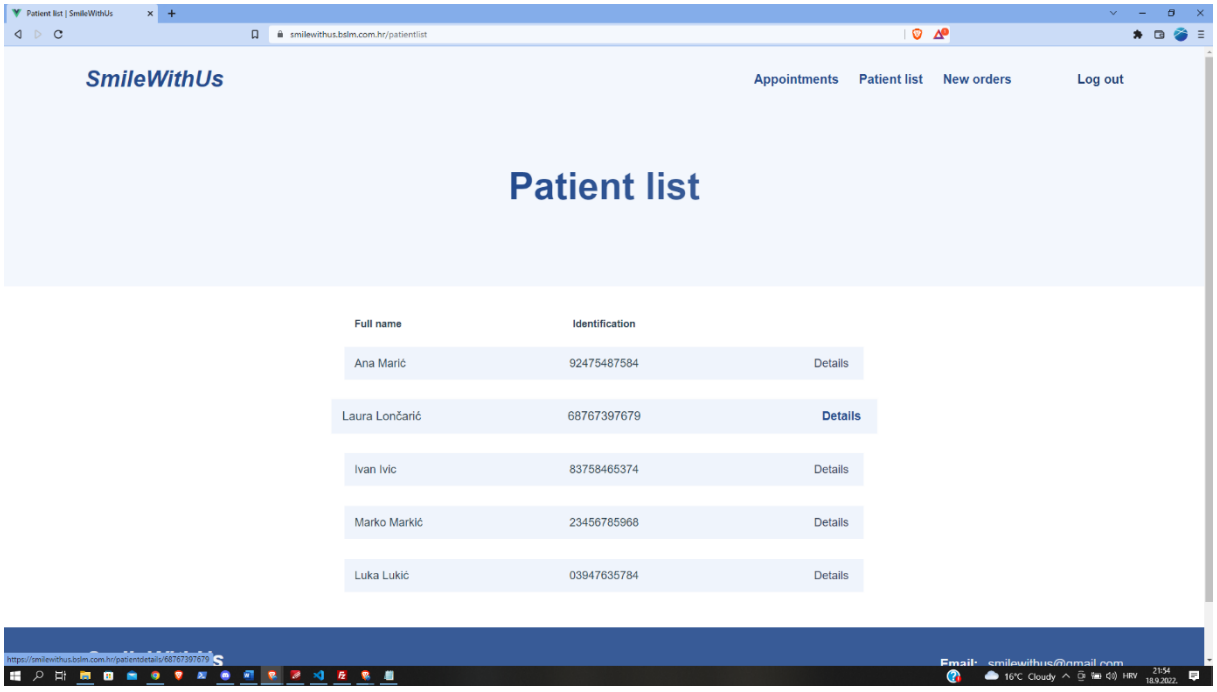
Slika 48. Određivanje točnog termina korisnika

Kada su detalji termina određeni korisnik dobiva potvrdu na email sa svim detaljima termina. Primjer potvrde putem email-a prikazan je na slici 50.



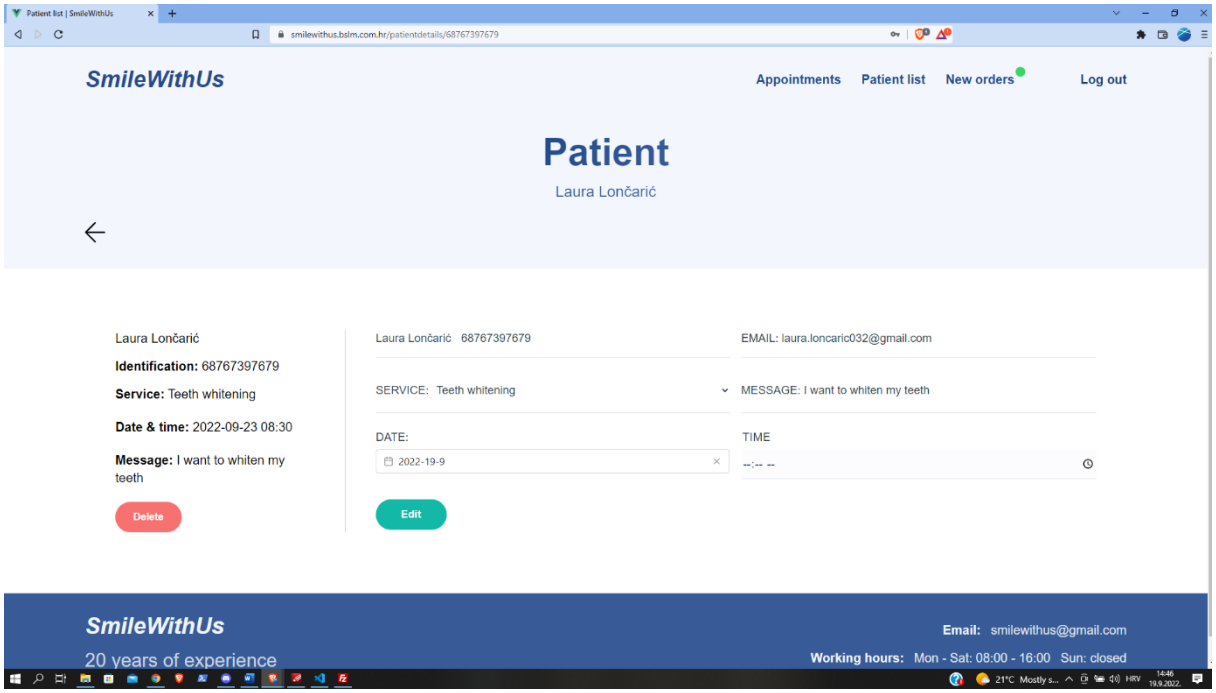
Slika 49. EmailJS – potvrda zakazanog termina korisnika

Sučelje ordinacija također pruža listu pacijanata, tj. njihovih zakazanih termina što je prikazano na slici 50.



Slika 50. Lista pacijenata

Klikom na gumb Details ordinacija može mijenjati detalje zakazanog termina prema potrebi ili ga potpuno opbrisati ukoliko pacijent odustane od termina.



Slika 51. Detalji termina pacijenta s mogućnošću promijene detalja i brisanja

6. Zaključak

Razvoj web aplikacija vrlo je tražen. Razvija se velikom brzinom svake godine. Olakšava proces razvoja i omogućuje tvrtkama brže postizanje ciljeva. Pomoću web aplikacija lakše je održavati ispravnu komunikaciju između potencijalnih kupaca ili korisnika i poslovne organizacije. Time se povećava popularnost organizacije i poboljšava stvaranje novih klijenata. Također web aplikacije pružaju opcije za poboljšanje korisničke podrške. Prednost takvih aplikacija je u tome što im se može pristupiti bilo kada i bilo gdje. S obzirom da svijet postaje digitaliziran, korisnici cijene internetska iskustva.

Ovom dokumentacijom opisan je razvoj web aplikacije koja obuhvaća listu privatnih stomatoloških ordinacija na jednom mjestu. Provedena je kritička analiza srodnih rješenja kojom je utemeljeno da je tehnologija u zdravstvu nije razvijena u skladu s očekivanima korisnika. Gotovo nijedna stomatološka ordinacija u Hrvatskoj ne pruža učinkovit sustav za online rezerviranje termina. Proces zakazivanja termina i dalje se većinom odvija telefonski što uzrokuje naporan, neugodan proces te često rezultira dugim čekanjem. Ovaj nedostatak pristupa stomatološkim ordinacijama predstavlja problem pacijentima. Stoga, ova aplikacija fokusirana je na bolje razumijevanje korisnika, onoga što im je potrebno, kako bi ispunila očekivanja korisnika i pružila im smisleno i relevantno iskustvo. Nudi korisnicima jednostavan, brz i praktičan proces zakazivanja termina u samo tri koraka. Online zakazivanje termina omogućuje pacijentima rezerviranje termina kada im odgovara bez potrebe zvanja ordinacije što znatno smanjuje vrijeme čekanja. Omogućava ordinacijama veću kontrolu nad pacijentima i popunjavanje praznina u terminima. Digitalizacija ordinacija smanjuje poremećaje u rasporedu što podupire postizanje željenog cilja te smanjuje količinu zamornih zadataka zaposlenika pojednostavljivanjem uredskih operacija.

S druge strane, aplikacija još uvijek ima nekoliko nedostataka. Korisnici nemaju prikaz zauzetih i slobodnih termina što može uzrokovati ne slaganja između pružatelja usluga i korisnika. Isto tako, aplikacija ne pruža mogućnost odjavljivanja termina. Ukoliko bi se aplikacija realizirala, dodala bi se mogućnost odjave termina s dodatnom povratnom informacijom putem email-a. Isto tako napravio bi se kalendar bez korištenja biblioteka kako bi se u potpunosti prilagodio potrebama procesa online zakazivanja termina.

Pružanje online sustava velika je prednost. Ima potencijal za razvoj poslovanja dok postojeće klijente održava zadovoljnima. Implementacija web aplikacije za online zakazivanje termina

dobra je investicija u učinkovitosti ordinacije te pruža jedinstvenu platformu za rast ordinacije osmišljenu kako bi pomogla ordinacijama poboljšati sve poveznice s pacijentima.

7. Literatura

[1] Contributor, T. (2019, August 26). Web application (Web app). SearchSoftwareQuality. Retrieved September 16, 2022, from https://www.techtarget.com/searchsoftwarequality/definition/Web-application-Web-app

[2] HGRADES. (2017, November 6). Assessing Online Scheduling as an Emerging Trend in Scheduling Physician Appointments. Healthleaders. Retrieved September 19, 2022, from https://www.healthleadersmedia.com/innovation/assessing-online-scheduling-emerging-trend-scheduling-physician-appointments

[3] Introduction | Vue.js. Retrieved September 16, 2022, from https://vuejs.org/guide/introduction.html

[4] JavaScript | MDN. (2022, September 13). Retrieved September 17, 2022, from https://developer.mozilla.org/en-US/docs/Web/JavaScript

[5] What is JavaScript? - Learn web development | MDN. (2022, September 14). Retrieved September 17, 2022, from https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript

[6] Fitzgerald, A. (2022, May 30). https://blog.hubspot.com/website/what-is-tailwind-css. HubSpot.

[7] Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. (2020, November 15). Tailwind CSS. Retrieved September 13, 2022, from https://tailwindcss.com/

[8] Getting Started | Vuex. (n.d.). Retrieved September 17, 2022, from https://vuex.vuejs.org/guide/

[9] Vue Datepicker | Vue Datepicker. (n.d.-b). Retrieved September 17, 2022, from https://vue3datepicker.com/

[10] FullCalendar - JavaScript Event Calendar. (n.d.). Retrieved September 17, 2022, from https://fullcalendar.io/

[11] Wikipedia contributors. (2022, September 8). Visual Studio Code. Wikipedia. Retrieved September 17, 2022, from https://en.wikipedia.org/wiki/Visual_Studio_Code

[12] https://www.ordinacijaceovic.hr/

[13] ARENA - Vrhunska stomatoloÅ¡ka poliklinika u Zagrebu. (n.d.). Retrieved September 18, 2022, from https://arenapoliklinika.hr/

[14] Stomatološka Ordinacija. (2022, August 4). Retrieved September 18, 2022, from https://skodadent.com/

Popis slika