Web aplikacija za online dogovaranje konzultacija

Džoić, Antonio

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Web aplikacija za online dogovaranje konzultacija

Diplomski rad

Antonio Džoić

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 18.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Antonio Džoić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D-1199R, 07.10.2021.
OIB studenta:	01526217278
Mentor:	izv. prof. dr. sc. Ivica Lukić
Sumentor:	,
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Mirko Köhler
Član Povjerenstva 1:	izv. prof. dr. sc. Ivica Lukić
Član Povjerenstva 2:	Miljenko Švarcmajer, mag. ing. comp.
Naslov diplomskog rada:	Web aplikacija za online dogovaranje konzultacija
Znanstvena grana diplomskog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Web aplikacija omogućava studentu da pošalje zahtjev profesoru za konzultacije pri čemu ima mogućnost odabira hoće li konzultacije biti uživo ili online te odabire željeni termin konzultacija. Studentu su prikazani termini kada profesor Ferit-a nema predavanja. Nakon što profesor prihvati zahtjev za konzultacije, profesoru i studentu se termin upisuje u google kalendar te se postavlja podsjetnik za događaj.Tema rezervirana za: Antonio Džoić
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	18.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTIRADA

Osijek, 29.09.2023.

Ime i prezime studenta:	Antonio Džoić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1199R, 07.10.2021.
Turnitin podudaranje [%]:	4

Ovom izjavom izjavljujem da je rad pod nazivom **: Web aplikacija za online dogovaranje konzultacija**

izrađen pod vodstvom mentora izv. prof. dr. sc. Ivica Lukić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD **4**

1.1. Zadatak završnog rada **4**

2. PREGLED PODRUČJA TEME RADA **5**

2.1. Primjeri web aplikacija za zakazivanje termina **6**

3. KORIŠTENI ALATI I TEHNOLOGIJE **12**

3.1. Angular **12**

3.2. Angular material **12**

3.3. Laravel **13**

3.4. Microsoft Visual studio Code **15**

3.5. Git **15**

3.6. Google Kalendar API **15**

3.7. Google Meet **16**

4. PROGRAMSKO RJEŠENJE WEB APLIKACIJE **17**

4.1. Dizajn web aplikacije **18**

4.2. Dizajn baze podataka **19**

4.3. Izrada poslužitelja **21**

4.4. Izrada klijentske strane **30**

5. ZAKLJUČAK **36**

LITERATURA **38**

SAŽETAK **40**

ABSTRACT **41**

PRILOZI **42**

1. UVOD

U današnjem digitalnom dobu, gdje se tehnologija brzo razvija i postaje sveprisutna, učinkovitost i fleksibilnost u obrazovnom sektoru postaju ključni faktori uspjeha. Dok se mnogi aspekti obrazovanja moderniziraju kroz digitalne platforme, jedan od segmenata koji često ostaje zanemaren je organizacija konzultacija između studenata i profesora. Tradicionalni načini organizacije konzultacija često su neučinkoviti, vremenski zahtjevni i podložni greškama.Ovaj rad se bavi razvojem web aplikacije koja ima za cilj riješiti ovaj problem.

Cilj ovog rada je razviti web aplikaciju koja će omogućiti studentima i profesorima da jednostavno i učinkovito dogovaraju termine za konzultacije, bez nepotrebnih komplikacija i gubitka vremena. Implementacija ovakve aplikacije ima potencijal značajno smanjiti vrijeme i napor potreban za dogovaranje i održavanje konzultacija, čime se olakšava administrativni rad i poboljšava ukupno iskustvo kako za studente, tako i za profesore. Da bi se postigla ova svrha, koristit će se moderne tehnologije kao što su Angular za klijentsku stranu i Laravel za poslužiteljsku stranu. Ove tehnologije su odabrane zbog njihove skalabilnosti, sigurnosti i široke primjene u industriji, što osigurava dugoročnu učinkovitost i održivost aplikacije. Dodatno, aplikacija je integrirana s Google kalendarom i Google Meet platformom, čime se dodatno povećava njezina funkcionalnost i korisnička vrijednost. Ovaj rad će pružiti detaljan uvid u arhitekturu aplikacije, tehnologije koje su korištene, kao i ključne aspekte koji su razmotreni tijekom razvoja.

Kroz ovaj rad, se želi unaprijediti način na koji se konzultacije dogovaraju i upravljaju, čime se ne samo olakšava administrativni rad, već se i poboljšava kvaliteta obrazovanja i iskustvo svih uključenih strana.

1.1. Zadatak završnog rada

Web aplikacija omogućava studentu da pošalje zahtjev profesoru za konzultacije pri čemu ima mogućnost odabira hoće li konzultacije biti uživo ili online te odabire željeni termin konzultacija. Studentu su prikazani termini kada profesor FERIT-a nema predavanja. Nakon što profesor prihvati zahtjev za konzultacije, profesoru i studentu se termin upisuje u Google kalendar te se postavlja podsjetnik za događaj.

2. PREGLED PODRUČJA TEME RADA

U današnjem brzom i dinamičnom akademskom okruženju, učinkovita organizacija i upravljanje vremenom postaju sve važniji. Tradicionalni načini organizacije konzultacija često su neučinkoviti i podložni greškama.

Zbog boljeg razumijevanja problematike i identificiranja mogućih rješenja, potrebno je izvršiti analizu nedostataka koji prate tradicionalne načine dogovaranja konzultacija. Sagledavanjem ovih izazova, postat će jasno zašto je potrebno razviti novo, automatizirano rješenje koje će ovaj proces učiniti učinkovitijim, sigurnijim i prilagodljivijim. Nakon toga, slijedi pregled postojećih rješenja koja se koriste za automatizaciju zakazivanja konzultacija ili drugih sličnih usluga. Ovim će se istražiti kako se novo rješenje može pozicionirati u odnosu na njih, te koje jedinstvene prednosti može ponuditi.

Nedostaci tradicionalnog pristupa dogovaranja konzultacija u odnosu na online način su sljedeći:

- Traženje e-mail adrese, sastavljanje poruke i čekanje na odgovor može biti vremenski zahtjevno za oba sudionika.
- Teško je održavati pregled nad svim zakazanim i predloženim terminima, što povećava rizik od dvostrukog zakazivanja ili propuštenih sastanaka.
- Profesori često imaju zatrpane e-mail pretince i mogu previdjeti ili zakasniti s odgovorom na zahtjev za konzultacijama.
- Studenti mogu predložiti termine koji nisu u skladu s dostupnošću profesora, što zahtijeva dodatnu komunikaciju i koordinaciju.
- Različiti studenti mogu koristiti različite formate i predloške za e-mail, što može otežati organizaciju za profesora.
- Česti e-mailovi o konzultacijama mogu ometati druge važne komunikacije u e-mail pretincu profesora.
- Bez automatskih podsjetnika, postoji veći rizik da će se jedna ili obje strane zaboraviti na zakazane konzultacije.

Razumijevanje ovih nedostataka može pomoći u identifikaciji ključnih područja koja online rješenje za zakazivanje konzultacija treba adresirati.

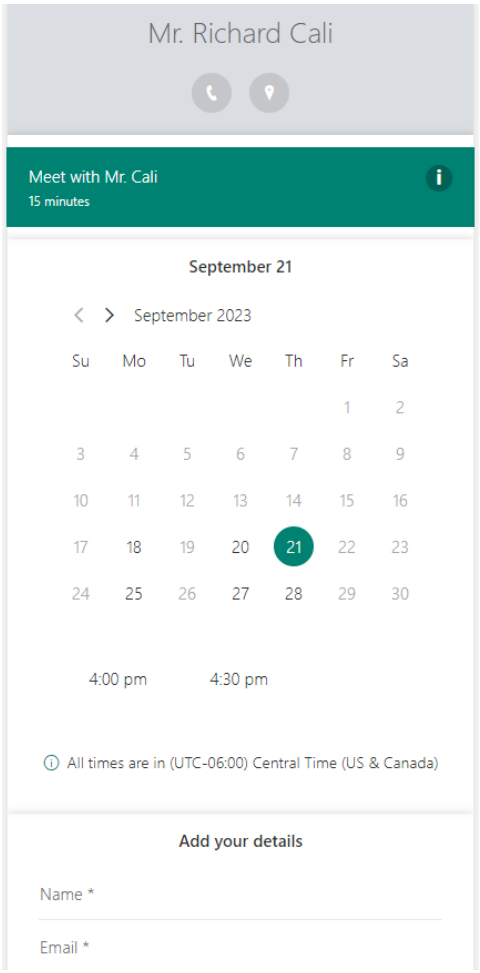
Nakon uočenih nedostataka tradicionalnog pristupa zakazivanju konzultacija potrebno je odrediti prednosti koje novo rješenje donosi, a to su:

- Značajno je smanjeno vrijeme potrebno za dogovor između studenta i profesora zahvaljujući automatizaciji procesa zakazivanja.
- Omogućen je lakši pregled svih zakazanih i dostupnih termina putem centralizirane platforme, čime je smanjen rizik od dvostrukog zakazivanja ili propuštenih sastanaka.
- Uklonjena je potreba za komunikacijom putem e-maila, čime je omogućeno brže i učinkovitije zakazivanje.
- Automatskom sinkronizacijom dostupnosti profesora omogućeno je preciznije zakazivanje.
- Smanjen je rizik od zaborava konzultacija zahvaljujući automatskim podsjetnicima koji se šalju objema stranama.
- Proširena je funkcionalnost i olakšana organizacija *online* konzultacija zahvaljujući mogućnosti integracije s Google kalendarom i Google Meet platformom.
- Omogućeno je lako dodavanje novih funkcionalnosti i prilagodba rastućim potrebama korisnika zahvaljujući skalabilnoj arhitekturi aplikacije.

2.1. Primjeri web aplikacija za zakazivanje termina

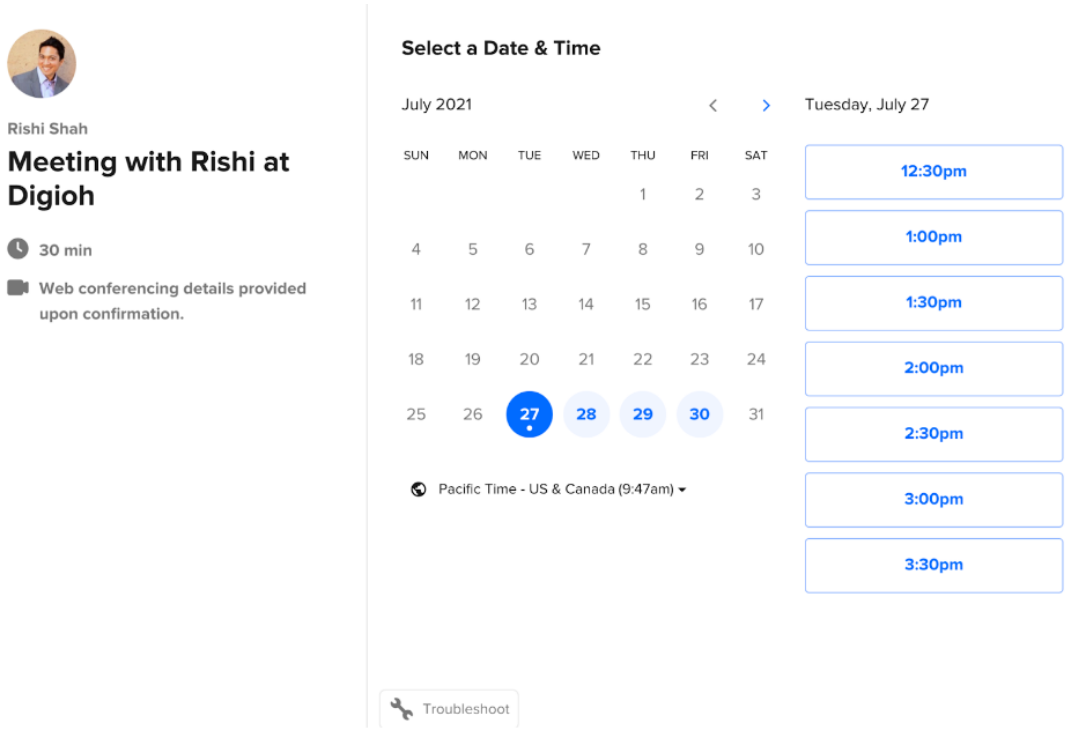
Nakon uočenih nedostataka tradicionalnog zakazivanja konzultacija i prednosti online rješenja potrebno je pronaći nekoliko postojećih rješenja za zakazivanje konzultacija s profesorima. Zbog manjka postojećih rješenja za zakazivanje termina konzultacija s profesorima navedena su slična rješenja koja služe za zakazivanje termina. Primjeri postojećih rješenja su:

1. Benedictine University [1]: Web stranica Benedictine sveučilišta koja služi za zakazivanje konzultacija s profesorima. Potrebno je odabrati jedan od ponuđenih fakulteta nakon čega se prikazuje popis svih profesora tog fakulteta dostupnih za konzultacije. Nije potrebna registracija, dovoljno je pri zakazivanju upisati ime i email adresu. Forma za zakazivanje konzultacija se razlikuje ovisno o odabranom fakultetu. Jednu od forma je moguće vidjeti na slici 2.1..



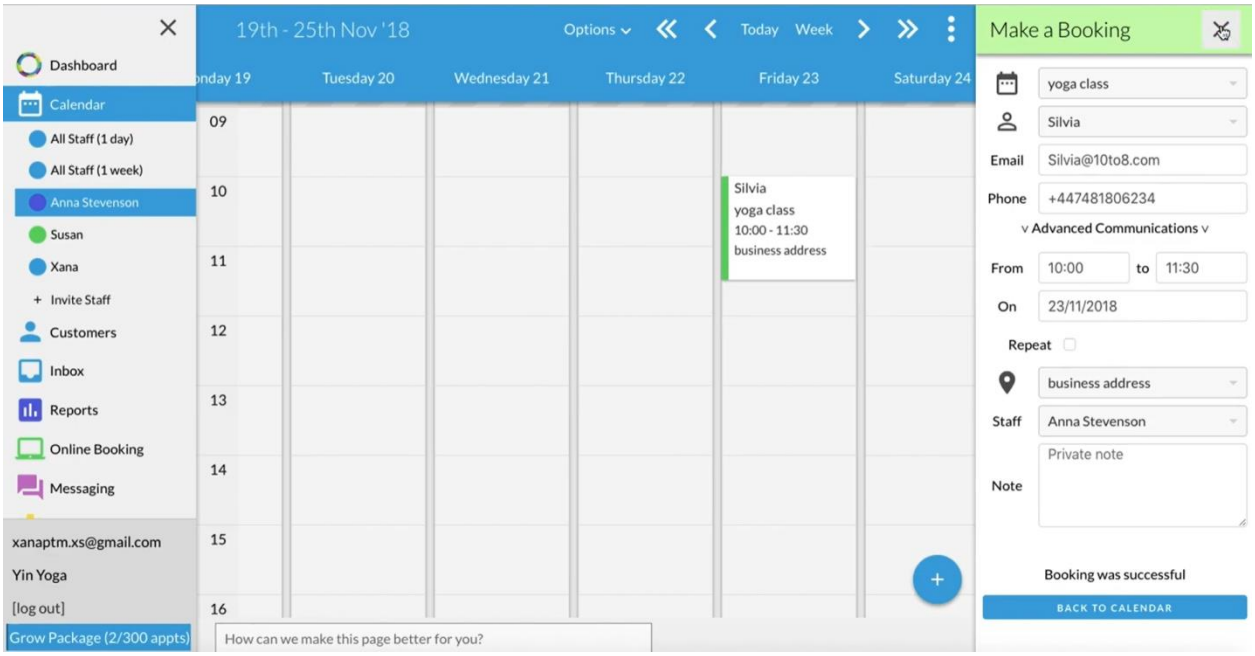
Slika 2.1. Forma za zakazivanje konzultacija na Benedictine sveučilištu

2. Calendly [2]: Calendly (Slika 2.2.) je popularna web aplikacija za zakazivanje sastanaka i konzultacija koja automatizira proces dogovaranja termina između dvije ili više strana. Ova platforma omogućuje korisnicima da postave svoje dostupne termine i zatim omogući drugima da izaberu jedan od tih termina za sastanak ili konzultaciju. Calendly se može integrirati s popularnim kalendarima kao što su Google kalendar, Outlook i Apple kalendar, omogućujući automatsko ažuriranje i izbjegavanje dvostrukih rezervacija.



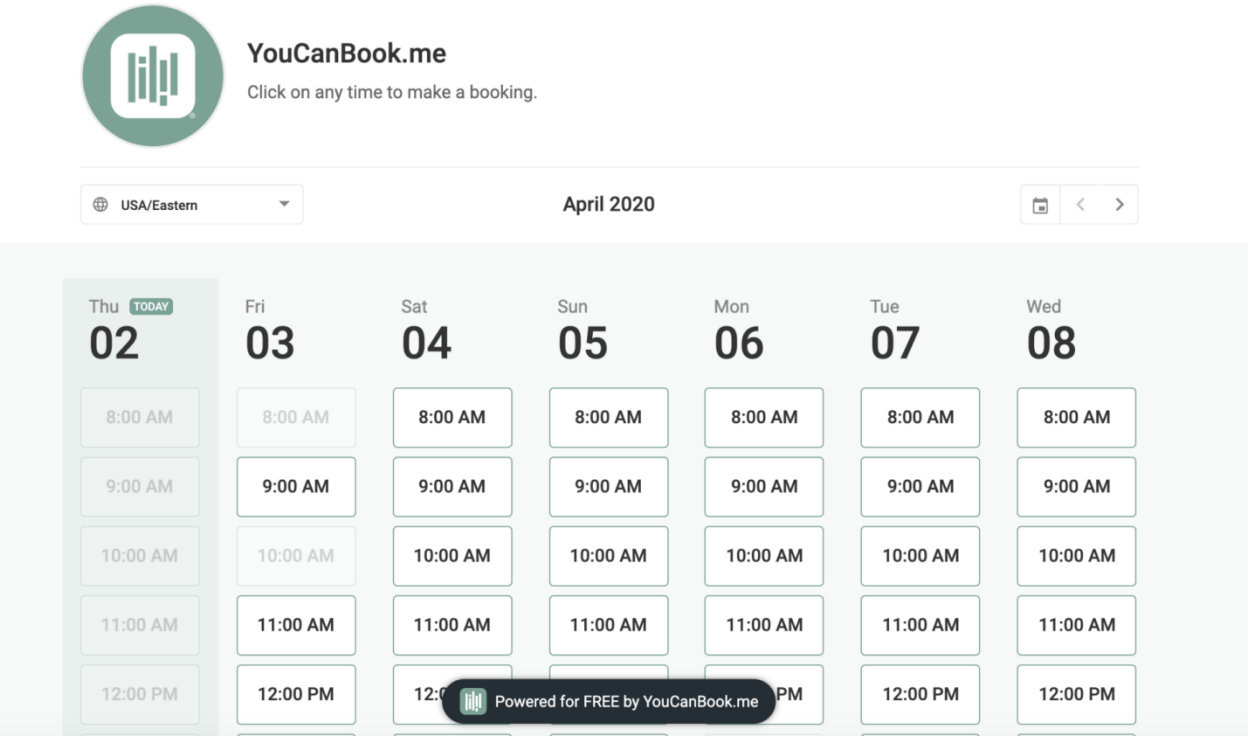
Slika 2.2. Calendly izgled forme za zakazivanje konzultacija

3. Sign in scheduling [3]: Web platforma za upravljanje zakazivanjem i sastancima koja je dizajnirana da olakša koordinaciju između različitih strana, prije je bila poznata pod imenom 10to8 (Slika 2.3.). Ova aplikacija je posebno korisna za poslovne korisnike, uključujući klinike, salone, konzultante i druge profesionalce koji redovito zakazuju sastanke s klijentima ili pacijentima. Platforma se može integrirati s različitim kalendarima, uključujući Google kalendar, Outlook i drugi, čime se omogućuje automatsko ažuriranje i sprečavanje dvostrukih rezervacija. Nudi različite opcije za komunikaciju, uključujući automatske email i SMS podsjetnike, kao i mogućnost vođenja bilješki i povijesti komunikacije s klijentima. Prilagodljiv je i može se koristiti u različitim sektorima i industrijskim vertikalama. Nudi i mogućnost prilagodbe izgleda i funkcionalnosti kako bi se bolje uklopila u specifične potrebe korisnika. Platforma nudi različite analitičke alate koji omogućuju korisnicima da prate učinkovitost zakazivanja, stopu otkazivanja, angažman klijenata i druge ključne metrike.



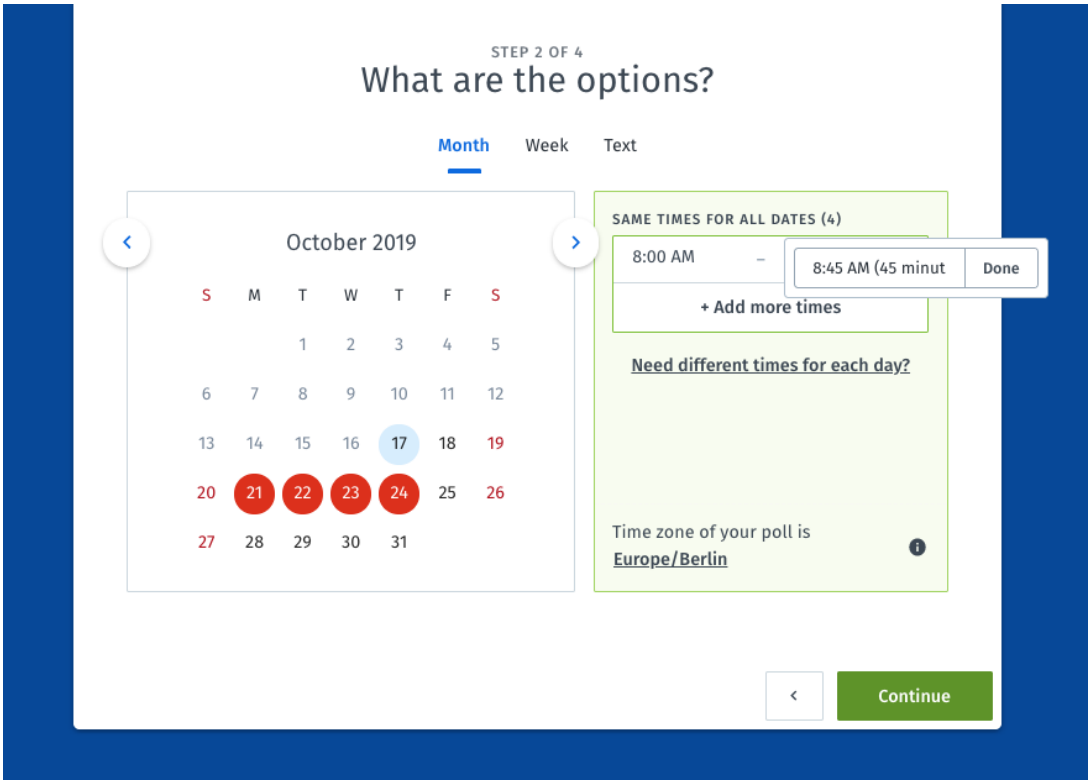
Slika 2.3. Sign in scheduling forma za zakazivanje termina

4. YouCanBook [4]: YouCanBook je online platforma za upravljanje zakazivanjem koja omogućuje pojedincima i organizacijama da automatiziraju i pojednostave proces zakazivanja sastanaka, konzultacija ili drugih vrsta događanja (Slika 2.4.). Omogućuje korisnicima da postave dostupne termine i uvjete zakazivanja, dok klijenti mogu samostalno izabrati željeni termin putem korisničkog sučelja, što smanjuje potrebu za ručnim upravljanjem. Platforma se može integrirati s popularnim kalendarima kao što su Google kalendar i Microsoft Outlook, čime se automatski ažuriraju zakazani termini i sprečavaju dvostruke rezervacije. Nudi funkcionalnost slanja automatskih email i SMS podsjetnika kako korisnicima, tako i klijentima, s ciljem smanjenja broja propuštenih ili zaboravljenih sastanaka. Platforma je vrlo prilagodljiva i omogućuje korisnicima da prilagode izgled stranice za zakazivanje, dodaju logotipe i boje te postave različite uvjete i opcije zakazivanja. Platforma nudi različite alate za praćenje i analizu, uključujući izvještaje o učinkovitosti, stopi otkazivanja i drugim relevantnim metrikama. Također podržava više jezika, što ga čini pogodnim za međunarodne korisnike i organizacije koje posluju u više zemalja,



Slika 2.4. You can book forma za zakazivanje konzultacija

5. Doodle Poll [5]: Doodle Poll je online alat za zakazivanje koji omogućuje korisnicima da brzo i učinkovito koordiniraju sastanke, događanja ili bilo koju drugu vrstu okupljanja s više ljudi. Ovaj alat je posebno koristan kada je potrebno uskladiti rasporede više osoba, što može biti izazovno putem tradicionalnih metoda kao što su e-mail ili telefonski pozivi (Slika 2.5.). Korisnik može brzo kreirati anketu s više mogućih termina i poslati je učesnicima putem e-maila ili direktnog linka. Korisnici mogu predložiti više termina i datuma, a učesnici mogu glasati za one koji im najviše odgovaraju. Također je moguće postaviti različite vrste anketa, kao što su ankete s jednim ili više mogućih odgovora. Učesnici mogu vidjeti odgovore drugih osoba, što olakšava koordinaciju, ali također imaju opciju da ostanu anonimni ako to žele. Doodle Poll se može integrirati s popularnim kalendarima kao što su Google kalendar, Microsoft Outlook i iCal, omogućujući automatsko ažuriranje zakazanih termina.



Slika 2.5. Doodle Poll forma za kreiranje ankete

3. KORIŠTENI ALATI I TEHNOLOGIJE

3.1. Angular

Angular je jedan od najpopularnijih okvira za razvoj web aplikacija i koristi se za izgradnju klijentske strane projekta. Razvijen od strane Googlea, ovaj okvir omogućuje brz i učinkovit razvoj visoko interaktivnih web stranica [6]. Angular koristi TypeScript jezik, koji je nadogradnja JavaScripta i omogućuje bolju tipizaciju i objektno-orijentirano programiranje.

Arhitektura Angulara je modularna i sastoji se od komponenti, modula, usluga i direktiva. Komponente su osnovni građevni blokovi aplikacije i koriste se za definiranje pogleda i logike koji ih upravlja. Moduli omogućuju grupiranje komponenti i drugih Angular resursa. Usluge se koriste za izolaciju poslovne logike i omogućuju ponovnu upotrebu koda. Direktive omogućuju manipulaciju DOM-a (engl. *Document Object Model*) [7].

U ovom projektu, Angular je korišten za izgradnju korisničkog sučelja koje je dinamičko i responzivno. Korištenje Angular CLI (engl. *Command Line Interface*) omogućilo je brzo generiranje komponenti, modula i usluga, čime je ubrzan razvojni proces.

3.2. Angular material

Za izgradnju korisničkog sučelja, pored Angulara, korišten je i Angular Material. Ovaj okvir pruža niz prethodno dizajniranih komponenti koje su u skladu s Material Design specifikacijama koje je Google postavio. Angular Material omogućuje brzu i jednostavnu integraciju komponenti kao što su dugmadi, kartice, trake za navigaciju i dijaloški okviri.

Korištenjem Angular Materiala, postignuta je konzistentnost u dizajnu i poboljšana korisnička iskustva. Također, ovaj okvir omogućuje laku prilagodbu za različite uređaje, što je ključno za responzivni dizajn [8].

U projektu je Angular Material korišten za izradu forme za prijavu, navigacijske trake i kartica koje prikazuju informacije. Korištenjem ovih prethodno dizajniranih komponenti, smanjeno je vrijeme potrebno za razvoj i testiranje korisničkog sučelja.

3.3. Laravel

Laravel je jedan od najpopularnijih radnih okvira za razvoj web aplikacija. Razvijen je s ciljem pojednostavljenja raznih zadataka koji su uobičajeni u većini web projekata, kao što su autentifikacija, usmjeravanje, sesije i keširanje. Laravel je napisan u PHP-u i omogućuje brže pisanje koda i sa manje grešaka [9].

Arhitektura Laravela temelji se na modelu MVC (engl. *Model-View-Controller*), što omogućuje organizaciju koda na način koji olakšava razvoj, održavanje i testiranje aplikacija. Model je srž svake MVC aplikacije. On predstavlja poslovnu logiku i podatke. Model je odgovoran za komunikaciju s bazom podataka i manipulaciju podacima. Kao primjer može poslužiti web trgovina, model bi bio odgovoran za sve što se tiče proizvoda: dodavanje novih proizvoda, ažuriranje postojećih, brisanje i tako dalje. Model je neovisan o korisničkom sučelju i ne bavi se prikazivanjem podataka. Pogled je odgovoran za prikazivanje podataka korisniku. On prima podatke iz modela i prikazuje ih na korisničkom sučelju. Pogled može biti bilo što: web stranica, mobilna aplikacija ili čak konzolna aplikacija. Pogled je pasivan i ne zna ništa o poslovnoj logici ili manipulaciji podacima. Njegova jedina svrha je prikazati podatke na način koji je razumljiv korisniku. Upravljač je komponenta koja povezuje model i pogled. On prima korisničke zahtjeve, obrađuje ih kroz model i ažurira pogled. Upravljač zna kada treba pokrenuti koju komponentu i kako. Na primjer, kada korisnik klikne na gumb za dodavanje proizvoda u košaricu, Upravljač će primijetiti taj zahtjev, zatražiti od Modela da ažurira količinu proizvoda u košarici i potom ažurirati Pogled kako bi korisnik vidio novu količinu u košarici [10]. MVC arhitektura ima mnoge prednosti, uključujući modularnost, fleksibilnost i lakše testiranje. Međutim, ona također može biti složena za razumijevanje i implementaciju, posebno za početnike. Ključna je za razvoj skalabilnih, održivih i testabilnih web aplikacija. Kroz jasnu separaciju odgovornosti između modela, pogleda i upravljača, omogućuje razvojnim timovima da brže i učinkovitije razvijaju aplikacije [11].

Eloquent ORM (engl. *Object-relational mapping*) je jedna od najistaknutijih komponenti Laravel okvira za razvoj web aplikacija. Ovaj alat omogućuje jednostavno i elegantno upravljanje interakcijom između aplikacije i relacijske baze podataka. Zaslužan je za pojednostavljenje mnogih aspekata rada s bazom podataka, uključujući dohvaćanje, ažuriranje i brisanje podataka [11]. Koristi aktivni zapis dizajn obrazac za rad s bazom podataka. Svaki model u Eloquentu odgovara jednoj tablici u bazi podataka, a svaka instanca modela predstavlja redak u toj tablici. Ovo omogućuje intuitivan i objektno-orijentiran pristup upravljanju podacima [12]. Njegovim

korištenjem, programeri mogu lako definirati modele i njihove relacije. Na primjer, ako postoje model Student i model *Course*, može se definirati njihova međusobna relacija koristeći Eloquentove metode kao što su *hasMany* i *belongsTo*. Ovo omogućuje jednostavno dohvaćanje povezanih podataka. Omogućuje jednostavno izvršavanje osnovnih CRUD (engl. *create, read, update, delete*) operacija. Na primjer, za stvaranje novog zapisa, može se koristiti metoda *creat*e, dok se za ažuriranje koristi metoda *updat*e. Brisanje zapisa je jednako jednostavno i izvodi se pomoću metode *delete*. Eloquent dolazi s ugrađenim upitačem (engl. *Query builder*) koji omogućuje složeno i dinamično kreiranje SQL upita. Ovo je posebno korisno za izvršavanje kompleksnih operacija na bazi podataka koje nisu lako ostvarive koristeći samo osnovne CRUD metode [13].

Laravel koristi Blade, vlastiti sustav za rad s predlošcima. Iznimno je snažan i omogućuje uobičajene operacije kao što su nasljeđivanje predložaka i uvjetno prikazivanje podataka. Osim toga, Blade je proširiv, što znači da programeri mogu dodavati vlastite direktive. Sigurnost je jedan od ključnih aspekata svake web aplikacije, a Laravel nudi niz opcija za održavanje sigurne aplikacije.

U Laravelu, međusloj (engl. *Middleware*) predstavlja sloj koji se nalazi između zahtjeva i odgovora u aplikaciji. On omogućuje filtriranje HTTP zahtjeva koji ulaze u aplikaciju. Na primjer, Laravel koristi međusloj za provođenje autentifikacije, logiranje, keširanje i druge zadatke koji se trebaju izvršiti prije ili nakon HTTP zahtjeva [11].

Laravel koristi Composer za upravljanje ovisnostima, što omogućuje jednostavno dodavanje i ažuriranje paketa. Composer također automatski rješava ovisnosti i preuzima potrebne pakete, što znatno ubrzava razvojni proces [13].

Artisan je komandno-linijski alat koji dolazi ugrađen u Laravel okvir za razvoj web aplikacija. Ovaj alat pruža niz korisnih komandi koje olakšavaju razvoj, testiranje i održavanje aplikacija. Jedan je od ključnih elemenata koji čine Laravel jednim od najpopularnijih PHP okvira. Omogućuje programerima da izvršavaju različite zadatke koji su uobičajeni u razvoju web aplikacija. Na primjer, Može se koristiti za generiranje kontrolera, modela i pogleda, migracije baze podataka, izvršavanje testova i mnoge druge zadatke. Ovaj alat je posebno koristan za automatizaciju rutinskih zadataka, čime se štedi vrijeme i resursi. Također nudi širok spektar komandi koje su lako dostupne putem komandne linije. Na primjer, komanda *php artisan make:controller* će generirati novi kontroler u Laravel aplikaciji. Osim toga, omogućuje i kreiranje vlastitih komandi, što dodatno proširuje njegovu funkcionalnost [11]. Artisan je duboko

integriran s ostalim komponentama Laravel okvira. Kada se koristi za kreiranje migracija baze podataka, generirani kod će automatski koristiti Eloquent,, za manipulaciju podacima. Ova duboka integracija ga čini izuzetno korisnim alatom u ekosustavu Laravela. Artisan pruža mnoge prednosti, uključujući brži razvoj, automatizaciju i visoku produktivnost. Međutim, kao i svaki alat, ima svoje nedostatke. Na primjer, za neiskusne programere, može biti malo zastrašujući zbog broja dostupnih komandi i opcija.

3.4. Microsoft Visual studio Code

Visual studio Code ili skraćeno VS Code je uređivač koda koji nudi brojne funkcionalnosti za programiranje. Opremljen je s raznim alatima kao što su otklanjanje pogrešaka, isticanje sintakse, inteligentno dovršavanje koda, refaktoriranje i ugrađeni Git. Osim toga, korisnici mogu prilagoditi VS Code instaliranjem proširenja, mijenjanjem tipkovničkih prečaca i postavki. Uz to, VS Code osigurava osnovnu podršku za većinu programskih jezika, uključujući isticanje sintakse, podudaranje zagrada i preklapanje koda. Dodatno, proširenja pružaju podršku za nove jezike, teme, alate za ispravljanje pogrešaka i analizu koda.

Jedna od ugrađenih značajki VS Code-a je kontrola izvora (engl. *Source control*). Kontrola izvora ima svoju posebnu karticu u traci izbornika, gdje korisnici mogu pristupiti postavkama kontrole verzija i pregledati promjene napravljene na trenutnom projektu. Za iskorištavanje ove značajke, potrebno je da bude povezan s podržanim sustavom za kontrolu verzija, kao što su Git ili Apache Subversion. Ova značajka korisnicima omogućuje stvaranje repozitorija te slanje i povlačenje zahtjeva izravno iz VS Code-a, što je vrlo korisno.

3.5. Git

Za razvoj aplikacije koriste se alati koji olakšavaju i ubrzavaju razvojni proces. Git je korišten kao sustav za kontrolu verzija, što omogućuje timsko radno okruženje i praćenje promjena u kodu. Za upravljanje paketima na frontendu koristi se npm (engl. *Node package manager*), dok se na backendu koristi Composer [14].

3.6. Google Kalendar API

Google Kalendar API (engl. *Application programming interface*) je sučelje koje omogućuje interakciju s Google Kalendarom putem programskih aplikacija. Ovaj API pruža mogućnost čitanja i manipulacije događajima unutar Google Kalendara, omogućujući tako integraciju s različitim aplikacijama i servisima [16]. Često se koristi za razvoj aplikacija koje zahtijevaju

kalendarske funkcionalnosti, kao što su planeri zadataka, upravljači projekata i aplikacije za rezervaciju termina. API koristi RESTful arhitekturu i podržava različite formate podataka, uključujući JSON i Atom. Autentifikacija se obavlja putem OAuth 2.0 protokola, što osigurava visoku razinu sigurnosti. Jedna od najvažnijih funkcionalnosti Google Kalendar API-a je mogućnost upravljanja događajima. Događaji se mogu kreirati, ažurirati, brisati i pretraživati koristeći različite API pozive. Osim upravljanja događajima, API također omogućuje upravljanje samim kalendarima. Moguće je kreirati nove kalendare, dijeliti ih s drugim korisnicima i čak uvoziti ili izvoziti događaje iz drugih kalendarskih aplikacija [16]. Google Kalendar API nudi i niz naprednih mogućnosti, uključujući rad s podsjetnicima, rad s praznicima i specijalnim događanjima, te mogućnost povezivanja s drugim Google servisima kao što su Google Tasks , Google Keep i Google Meet.

3.7. Google Meet

Google Meet je platforma za video konferencije koju je razvio Google kao dio svoje G Suite kolekcije poslovnih aplikacija. Ova platforma omogućuje korisnicima da organiziraju i sudjeluju u video sastancima visoke kvalitete, nudeći niz naprednih funkcionalnosti i opcija za prilagodbu. Google Meet koristi napredne algoritme za kompresiju videa i zvuka, što osigurava čistu sliku i zvuk sa malim brojem zastoja. Platforma je dostupna na različitim uređajima, uključujući računala, tablete i pametne telefone, te podržava više operativnih sustava [17]. Jedna od ključnih funkcionalnosti je mogućnost jednostavne organizacije sastanaka. Korisnici mogu zakazati sastanke unaprijed ili ih odmah započeti. Sastanci se mogu integrirati s Google Kalendarom, što omogućuje automatsko slanje pozivnica i podsjetnika. Kada su u pitanju platforme poput Google Meet-a posebno se pridaje velika važnost sigurnosti i privatnosti. Svi sastanci su enkriptirani, a pristup je moguć samo uz odobrenje organizatora. Također, postoji mogućnost zaključavanja sastanka kako bi se spriječio neautorizirani pristup. Platforma nudi niz naprednih opcija, uključujući mogućnost dijeljenja ekrana, rad u više kamera, te interaktivne ankete i kvizove. Također, postoji opcija za automatsko generiranje transkripata sastanka. Google Meet se široko koristi u poslovnom okruženju, obrazovanju, te za osobne potrebe. Njegova fleksibilnost i skalabilnost čine ga idealnim za različite vrste sastanaka, od malih timskih sastanaka do velikih webinara [18].

4. PROGRAMSKO RJEŠENJE WEB APLIKACIJE

Prije izrade dizajna web aplikacije i ostalih koraka potrebno je definirati funkcionalne zahtjeve koje web aplikacija treba zadovoljiti. Funkcionalni zahtjevi su:

1. Admin može stvarati nove korisnike, uključujući i profesore, ali se studenti i profesori ne mogu sami registrirati.
2. Korisnik se može prijaviti na svoj račun koristeći svoju e-mail adresu i lozinku.
3. Postoje tri vrste korisnika: Admin, Profesor i Student. Admin ima ovlasti stvaranja novih studenata, profesora i kolegija. Profesor može predavati više kolegija i držati konzultacije za svaki kolegij na kojem predaje. Studentu se prikazuju svi kolegiji na koje je upisan, a također može pregledati i profesore koji su izvođači na kolegijima na koje je upisan.
4. Student može pristupiti stranici kolegija i vidjeti popis profesora koji predaju taj kolegij.
5. Student može odabrati profesora i vidjeti osnovne informacije o profesoru, uključujući raspored konzultacija.
6. Student može podnijeti zahtjev za konzultacijama kod odabranog profesora pri čemu se studentu nude samo slobodni termini kada profesor nema zakazane konzultacije. Nakon slanja zahtjeva, student čeka da profesor prihvati konzultacije. Student može odabrati hoće li konzultacije biti uživo ili putem video chat sobe prilikom slanja zahtjeva za konzultacijama.
7. Profesor može prihvatiti ili odbiti zahtjev za konzultacijama od strane studenta. Ako profesor prihvati konzultacije, termin se zapisuje u Google kalendar profesora te se šalje pozivnica za događaj studentu.
8. U terminu konzultacija, profesor i student pristupaju Google Meet sobi putem koje je moguće prikazivati kameru te slati poruke u chat.

Osim funkcionalnih zahtjeva postoje i nefunkcionalni zahtjevi. Nefunkcionalni zahtjevi su zahtjevi koji se odnose na karakteristike i ograničenja sustava te utječu na način na koji sustav funkcionira. Neki od primjera nefunkcionalnih zahtjeva su:

1. Performanse, web aplikacija treba imati brzo vrijeme odziva kako bi korisnici mogli učinkovito koristiti sustav. Sustav treba biti skalabilan kako bi mogao podržavati rastući broj korisnika i podataka.

2. Sigurnost, korisnički podaci uključujući lozinke, trebaju biti sigurno pohranjeni i preneseni putem šifrirane veze (HTTPS). Pristup određenim dijelovima aplikacije, poput administratorskog sučelja, treba biti ograničen samo ovlaštenim korisnicima.
3. Pouzdanost, sustav treba biti pouzdan i stabilan, s minimalnim prekidima u radu. Treba osigurati sigurnosnu kopiju podataka kako bi se izbjegao gubitak podataka u slučaju kvara ili neočekivanog događaja.
4. Korisničko sučelje, treba biti intuitivno i jednostavno za korištenje kako bi korisnici lako navigirali kroz aplikaciju. Dizajn sučelja treba biti responzivan i prilagodljiv različitim uređajima i veličinama zaslona.
5. Integracija s Google kalendarom, sustav treba uspješno integrirati s Google kalendarom kako bi se automatski sinkronizirali termini konzultacija.
6. Održavanje, kod aplikacije treba biti čitljiv, dobro strukturiran i dokumentiran kako bi olakšao održavanje i daljnji razvoj. Potrebno je osigurati ažuriranje sigurnosnih zakrpa i ispravaka pogrešaka.

Nefunkcionalni zahtjevi su važni za osiguravanje ukupne kvalitete i performansi aplikacije te zadovoljavanje korisničkih očekivanja.

4.1. Dizajn web aplikacije

Dizajn web aplikacije odnosi se na proces planiranja i organiziranja kako će korisničko sučelje i korisničko iskustvo izgledati i funkcionirati. Uključuje sve vizualne i interaktivne aspekte aplikacije, uključujući raspored elemenata, stilizaciju, navigaciju, upotrebljivost, protok korisnika i mnoge druge faktore. S obzirom na funkcionalne zahtjeve može se zaključiti da web aplikacija treba sadržavati sljedeće poglede:

1. Prijavljivanje omogućuje korisnicima da unesu svoje pristupne podatke i pristupe svojim korisničkim računima.
2. Naslovna stranica za studenta prikazuje informacije o zakazanim konzultacijama, poslanim zahtjevima za konzultacijama te informacije o odbijenim konzultacijama.Pogled za profesora izgleda jednako kao i pogled za korisnika.Pogled za administratora prikazuje tablicu sa svim korisnicima.
3. Pogled tablice kolegija za studenta prikazuje popis kolegija na koje je student upisan te informacije o kolegiju. Pogled tablice kolegija za administratora prikazuje sve kolegije.

4. Pogled detalja o kolegiju za studenta prikazuje podatke o kolegiju te popis svih izvođača kolegija. Administrator vidi sve kao i student, ali ima i opciju za dodavanje i brisanje izvođača s kolegija.
5. Pogled za pregled i prihvaćanje ili odbijanje zahtjeva omogućuje profesorima pregled i prihvaćanje ili odbijanje zahtjeva za konzultacijama.

4.2. Dizajn baze podataka

Za izradu rješenja potrebno je ispravno dizajnirati bazu podataka. Iz funkcionalnih zahtjeva možemo zaključiti sljedeće relacije:

1. Korisnik (*User*) može imati ulogu profesora (*Professor*), studenta (*Student*) ili administratora (*Admin*). Ova veza se može ostvariti putem polja *role* u tablici *User* koji označava korisničku ulogu. Profesor može imati vezu "jedan naprema jedan" s tablicom *User* putem polja *user_id*, dok student također može imati vezu "jedan naprema jedan" s tablicom *User* putem polja *user_id*.
2. Profesor (*Professor*) predaje više kolegija (*Course*), dok kolegij može biti predmet više profesora. Ovo je primjer veze "više naprema više". Da bi se to postiglo, potrebno je stvoriti dodatnu tablicu *course_professor* koja će sadržavati strane ključeve *course_id* i *professor_id* koji će povezivati ove dvije tablice.
3. Kolegij (*Course*) može imati više studenata (*Student*), dok student može biti upisan na više kolegija. Ovo je također primjer veze "više naprema više".Da bi se to postiglo, potrebno je stvoriti dodatnu tablicu *course_student* koja će sadržavati strane ključeve *course_id* i *student_id* koji će povezivati ove dvije tablice.
4. Raspored (*Schedule*) povezan je s profesorom (*Professor*) stranog ključa *professor_id*. Ovdje je u pitanju veza "jedan naprema više" gdje profesor može imati više rasporeda konzultacija, ali svaki raspored konzultacija pripada samo jednom profesoru.
5. Zahtjev za konzultacije (*Consultation_request*) je povezan sa profesorom (*Professor*), studentom (*Student*) i rasporedom profesora (*Schedule*). Sve su veze "jedan naprema više" gdje profesor može imati više zahtjeva za konzultacije, student može poslati više zahtjeva za konzultacije.

Nakon što su definirane relacije, može se zaključiti koje su tablice potrebne u bazi podataka te koje atribute svaka tablica treba sadržavati. Tablice sa atributima koje su potrebni za web aplikaciju su sljedeće:

Tablica *users* (korisnici):

- *id* (primarni ključ)
- *name* (ime korisnika)
- *last_name* (prezime korisnika)
- *email* (email adresa korisnika)
- *password* (hashirana lozinka)
- *role* (uloga korisnika: admin, profesor, student)

Tablica *professors* (profesori):

- *id* (primarni ključ)
- *user_id* (strani ključ koji se odnosi na korisnika)

Tablica *students* (studenti):

- *id* (primarni ključ)
- *user_id* (strani ključ koji se odnosi na korisnika)

Tablica *courses* (kolegiji):

- *id* (primarni ključ)
- *name* (naziv kolegija)
- *description* (opis kolegija)

Tablica *course_professor* (povezna tablica za kolegije i profesore):

- *id* (primarni ključ)
- *course_id* (strani ključ koji se odnosi na kolegij)
- *professor_id* (strani ključ koji se odnosi na profesora)

Tablica *course_student* (povezna tablica za kolegije i studente):

- *id* (primarni ključ)
- *kolegij_id*: (strani ključ koji se odnosi na kolegij)
- *student_id*: (strani ključ koji se odnosi na studenta)

Tablica *schedule* (raspored profesora):

- *id* (primarni ključ)

- *professor_id* (strani ključ koji se odnosi na profesora)
- *date* (datum rasporeda)

Tablica *consultation_requests* (zahtjevi za konzultacije):

- *id* (primarni ključ)
- *student_id* (strani ključ koji se odnosi na studenta)
- *professor_id* (strani ključ koji se odnosi na profesora)
- *schedule_id* (strani ključ koji se odnosi na raspored profesora)
- *start_time* (početno vrijeme konzultacija)
- *end_time* (vrijeme završetka termina konzultacija)
- *status* (status zahtjeva, npr. čeka se odobrenje, prihvaćeno, odbijeno)
- *type* (vrsta konzultacija: uživo, online)
- *note* (napomena koju student šalje zajedno sa zahtjevom)
- *reason* (u slučaju odbijenog zahtjeva profesor može napisati razlog)
- *link* (predstavlja vezu za Google Meet sobu)

4.3. Izrada poslužitelja

U ovom poglavlju, detaljno će biti opisana izrada poslužiteja koji je ključna komponenta u razvoju web aplikacije. Za izradu poslužitelja, korišten je PHP u kombinaciji s Laravel radnim okruženjem [11]. Ova kombinacija omogućuje brz i učinkovit razvoj, visoku razinu sigurnosti i laku održivost koda.

U kontekstu ovog diplomskog rada, REST (engl. *Representational state transfer*) arhitektura poslužila je kao osnovni model za izgradnju API-ja. REST je arhitekturni stil koji koristi standardne HTTP metode i statusne kodove, URL-ove i MIME tipove. Njegova primjena omogućila je izgradnju skalabilnog i fleksibilnog API-ja koji se lako može integrirati s različitim klijentskim aplikacijama. HTTP metode su temeljne operacije koje omogućuju komunikaciju između klijenta i servera. U ovom projektu, koriste se sljedeće HTTP metode:

- GET: Metoda koja se koristi za dohvaćanje resursa s servera. Na primjer, u *UserController.php*, metoda GET koristi se za dohvaćanje informacija o korisnicima.
- POST: Ova metoda koristi se za slanje podataka na server. U kontekstu *AuthController.php*, POST metoda koristi se za registraciju novih korisnika.

- PUT: Metoda za ažuriranje resursa koji već postoji na serveru. U *CourseController.php*, PUT metoda koristi se za ažuriranje informacija o predavanjima.
- DELETE: Kao što ime sugerira, ova metoda koristi se za brisanje resursa. U *RoleController.php*, DELETE metoda koristi se za uklanjanje uloga.

Rutiranje zahtjeva je implementirano korištenjem Laravelovog rutiranja [13]. Ovo omogućuje jednostavno mapiranje URL-ova na odgovarajuće kontrolere i metode. Za pohranu podataka, korištena je MySQL baza podataka [19]. Eloquent ORM je korišten za interakciju s bazom podataka, što omogućuje jednostavno i učinkovito upravljanje podacima. Validacija podataka je ključna za očuvanje integriteta baze podataka. Laravelov ugrađeni validacijski mehanizam je korišten za ovu svrhu [13].

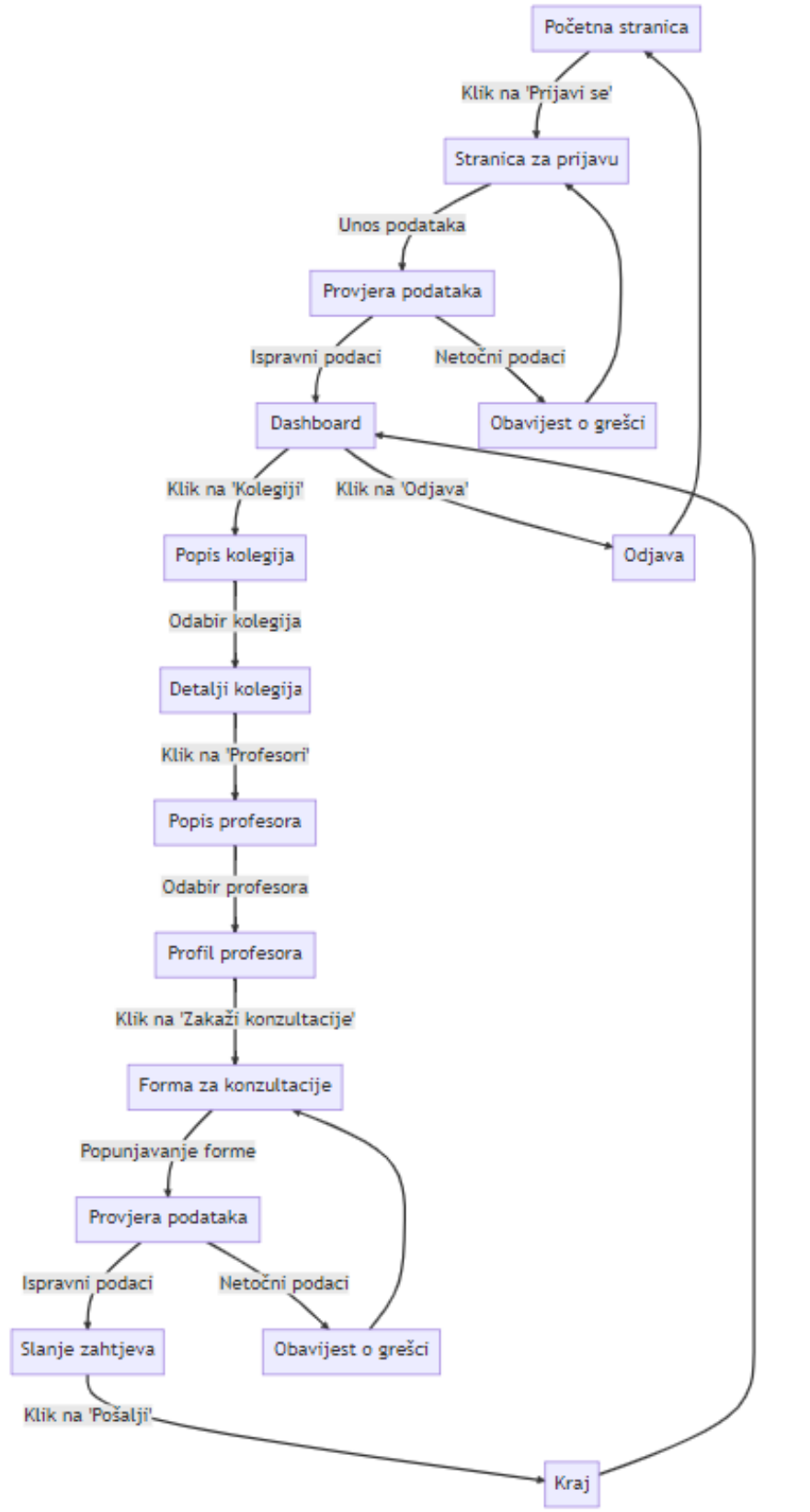
Dijagrami slučaja upotrebe (engl. *Use case diagrams*) koriste se kako bi se vizualno predstavila funkcionalnost sustava iz perspektive korisnika. Oni omogućuju lakše razumijevanje funkcionalnih zahtjeva sustava, interakcije između korisnika i sustava te granice sustava. Ovi dijagrami su posebno korisni u fazi planiranja i analize, gdje je važno razumjeti što korisnici očekuju od sustava. Neki od takvih dijagrama za ovaj projekt su:

- slanje zahtjeva za konzultacije profesoru,
- prihvaćanje ili odbijanje zahtjeva za konzultacije od strane profesora.

U nastavku su pobliže objašnjeni koraci tih dijagrama. Prvi dijagram je "Slanje zahtjeva za konzultacije profesoru". Na slici 4.1 su prikazani koraci:

1. Korisnik se pokušava ulogirati, ako su informacije netočne, korisniku se prikazuje poruka o grešci i vraća se na početni ekran za prijavu. Ako su informacije točne, korisnik se uspješno prijavljuje u sustav.
2. Pregled svih kolegija, korisnik odlazi na stranicu gdje su navedeni svi kolegiji na koje je upisan.
3. Odabir kolegija, korisnik odabire željeni kolegij iz popisa.
4. Odabir profesora, nakon odabira kolegija, korisniku se prikazuje lista profesora vezanih uz taj kolegij.
5. Korisnik odabire željenog profesora.
6. Unos vremena i datuma za konzultacije, korisnik unosi vrijeme početka i završetka konzultacija i odabire datum konzultacija.
7. Unos razloga za konzultacije, korisnik unosi razlog zašto želi konzultacije.

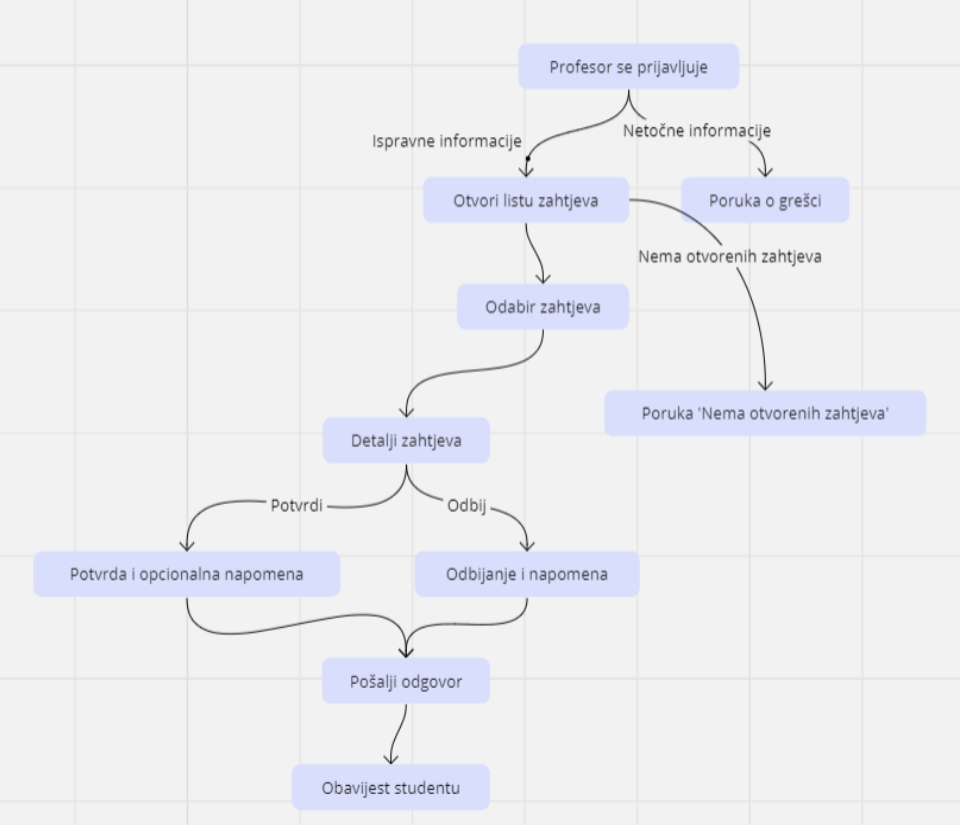
8. Odabir tipa konzultacija, korisnik odabire hoće li konzultacije biti uživo ili online.
9. Slanje zahtjeva, korisnik pritisne gumb "Pošalji zahtjev" za slanje zahtjeva za konzultacijama.Ako je zahtjev uspješno poslan, korisniku se prikazuje poruka o uspjehu. Ako zahtjev nije uspješno poslan, korisniku se prikazuje poruka o grešci.



Slika 4.1. Dijagram slučaja upotrebe slanje zahtjeva za konzultacije profesoru

Drugi dijagram slučaja upotrebe je Prihvaćanje ili odbijanje zahtjeva za konzultacije od strane profesora te je prikazan na slici 4.2. Koraci u ovom slučaju su:

1. Profesor se pokušava ulogirati, ako su informacije netočne, prikazuje se poruka o grešci i vraća se na početni ekran za prijavu. Ako su informacije točne, profesor se uspješno prijavljuje u sustav.
2. Profesor otvara popis zahtjeva za konzultacijama. Sustav prikazuje listu svih zahtjeva za konzultacijama koji čekaju odgovor. Ako nema zahtjeva za konzultacijama, sustav prikazuje poruku "Nema otvorenih zahtjeva".
3. Profesor odabire zahtjev na koji želi odgovoriti.
4. Sustav prikazuje detalje odabranog zahtjeva, uključujući informacije o studentu, predmetu, predloženom vremenu i datumu, te tipu konzultacija (uživo ili online).
5. Profesor bira opciju za potvrdu ili odbijanje zahtjeva.
6. Profesor pritisne gumb "Pošalji odgovor".
7. Sustav šalje odgovor studentu i ažurira status zahtjeva.



Slika 4.2. Dijagram slučaja upotrebe prihvaćanje ili odbijanje zahtjeva za konzultacije od strane profesora

Prvi korak u razvoju bio je postavljanje osnovne strukture projekta. To uključuje:

- Instalacija Laravela
- Postavljanje baze podataka
- Konfiguracija okruženja

Nakon tih koraka bilo je potrebno stvoriti tablice koje su u tom trenutku potrebne u projektu. Taj postupak obavljen je pomoću migracija što predstavlja *Code first* pristup generiranja tablica. *Code first* je pristup u razvoju aplikacija gdje se baza podataka generira iz koda koji je napisan unaprijed. Ovaj pristup je postao popularan u okruženjima kao što su *Entity Framework* i Laravel, gdje se koristi za definiranje i manipulaciju bazom podataka. *Code first* se često koristi u agilnim razvojnim okruženjima gdje je brzina i fleksibilnost ključna. Omogućuje programerima da se fokusiraju na logiku aplikacije, dok se struktura baze podataka generira automatski. Ovo je posebno korisno u ranim fazama razvoja, gdje se često mijenjaju zahtjevi i specifikacije. Prednosti ovog pristupa su brzina razvoja jer programeri mogu brzo prototipirati i iterirati kroz različite verzije aplikacije, jednostavnost jer ne zahtijeva duboko poznavanje SQL-a ili strukture baze podataka. Još jedna od prednosti je i održivost zato što jelakše održavati i ažurirati kod, jer su sve promjene centralizirane na jednom mjestu [21]. Naravno ovakav pristup ima i svoje mane, a to su nedostatak kontrole jer automatska generacija baze podataka može dovesti do problema sa performansama ili sigurnošću te je druga mana kompleksnost za složene baze podataka s mnogo relacija i ograničenja, ovaj pristup može biti neučinkovit.

U kontekstu projekta, *code first* pristup je korišten za stvaranje tablica, a jedna od tih tablica je bila i *consultation_requests*. Migracija koja je stvorila tu tablicu je prikazana na slici 4.3. . U metodi *up()* definirani su svi atributi tablice i njihovi tipovi podataka. Također su dodani i strani ključevi koji se odnose na druge tablice (*students, professors, schedules*). U metodi *down()* navedeno je kako ukloniti tablicu ako je potrebno.

```
0 references | 0 overrides
public function up()
{
    Schema::create('consultation_requests', function (Blueprint $table) {
        $table->id();
        $table->unsignedBigInteger('student_id');
        $table->unsignedBigInteger('professor_id');
        $table->unsignedBigInteger('schedule_id');
        $table->time('start_time');
        $table->time('end_time');
        $table->string('status');
        $table->string('type');
        $table->text('note')->nullable();
        $table->text('reason')->nullable();
        $table->string('link')->nullable();

        $table->foreign('student_id')->references('id')->on('students');
        $table->foreign('professor_id')->references('id')->on('professors');
        $table->foreign('schedule_id')->references('id')->on('schedules');
    });
}

/**
 * Reverse the migrations.
 *
 * @return void
 */
0 references | 0 overrides
public function down()
{
    Schema::dropIfExists('consultation_requests');
}
```

Slika 4.3. Migracija za stvaranje *consultation_requests* tablice

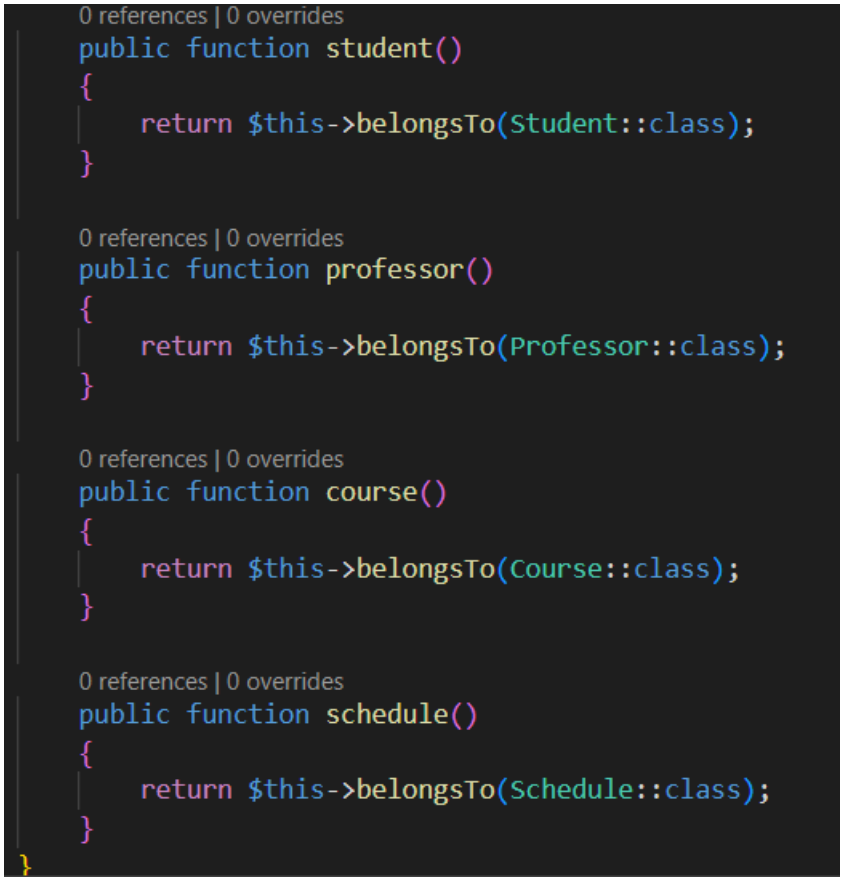
Sljedeći korak je bio omogućiti korisniku da se može prijaviti sa svojom email adresom i zaporkom. Za to je korišten Sanctum međusloj. Sanctum je dio Laravel ekosustava i pruža sveobuhvatno rješenje za API autentifikaciju, kao i zaštite sesija. Odabran je zbog svoje jednostavnosti i fleksibilnosti. On nudi API token autentifikaciju, kao i sesijsku autentifikaciju, što ga čini idealnim za projekte koji koriste kombinaciju tradicionalnih web aplikacija i API-ja. Također dolazi s dodatnim sigurnosnim značajkama, uključujući CSRF zaštitu. Instaliran je korištenjem Composer-a sa naredbom *composer require laravel/sanctum* nakon čega su konfiguracijske datoteke objavljene pomoću naredbe *php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"*. Izvršena je migracija baze podataka sa *php artisan migrate* kako bi se stvorile potrebne tablice za autentifikaciju. U *api.php* datoteci, *auth:sanctum* međusloj je dodan na rute koje zahtijevaju autentifikaciju. U kontrolerima,

auth:sanctum međusloj je dodan kako bi se osiguralo da samo autentificirani korisnici mogu pristupiti određenim resursima.

Kada korisnik uspješno izvrši prijavu, generira se jedinstveni API token koji se šalje korisniku. Ovaj token se koristi za autentifikaciju korisnika pri svakom budućem zahtjevu. Za web aplikacije koje koriste sesije, Sanctum koristi Laravelovu ugrađenu sesijsku autentifikaciju, ali dodaje dodatne sigurnosne mjere, kao što su CSRF tokeni. Može se integrirati s Laravel-ovim ugrađenim sistemom za upravljanje ulogama i dozvolama. Ovo omogućuje granularnu kontrolu nad resursima i akcijama koje korisnik može izvršiti. Mogu definirati pravila autorizacije koja se mogu primijeniti na rute ili kontrolere. Ova pravila se mogu koristiti u kombinaciji s Sanctum-om za dodatnu sigurnost. Prednost ovog međusloja je jednostavnost, što omogućuje brzu implementaciju. Može se koristiti za API autentifikaciju i mobilne aplikacije te ima dodatne sigurnosne mjere kao što su CSRF zaštita i enkripcija tokena. Nedostatak mu je ovisnost o Laravelu jer je specifičan za Laravel, što ga čini manje fleksibilnim za projekte koji koriste druge tehnologije. Za složenije zahtjeve u pogledu autentifikacije i autorizacije, kao što su višestruki faktori autentifikacije, Sanctum može biti ograničen.

Također je bilo potrebno napraviti međusloj čiji je zadatak bio vrlo jednostavan, ovisno o korisničkoj ulozi dopustiti ili odbiti pristup.

Nakon što je omogućena i osigurana prijava korisnika sljedeći korak je bio napraviti sve potrebne modele za generirane tablice i upravljače koji će izvoditi osnovne operacije nad modelima. U Laravelu, svaka tablica u bazi podataka ima svoj odgovarajući model. Modeli se koriste za interakciju s tablicama. Na primjer, za tablicu *consultation_requests*, kreiran je model pod nazivom *ConsultationRequest*. U modelu se također definiraju relacije s drugim modelima. Na slici 4.4. su prikazane sve relacije koje su definirane u *ConsultationRequest* modelu.



Slika 4.4. Relacije "ConsultationRequest" modela

Nakon modela bilo je potrebno generirati upravljač. Artisan naredbeni redak omogućuje brzo generiranje kontrolera pomoću *php artisan make:controller "ConsultationRequestController*". U kontroleru se definiraju metode koje odgovaraju različitim akcijama koje se mogu izvršiti na modelu, kao što su *index*, *create*, *store*, *edit*, *update*, i *destroy*. Nakon što su metode definirane, one se povezuju s odgovarajućim rutama u *api.php* datoteci.

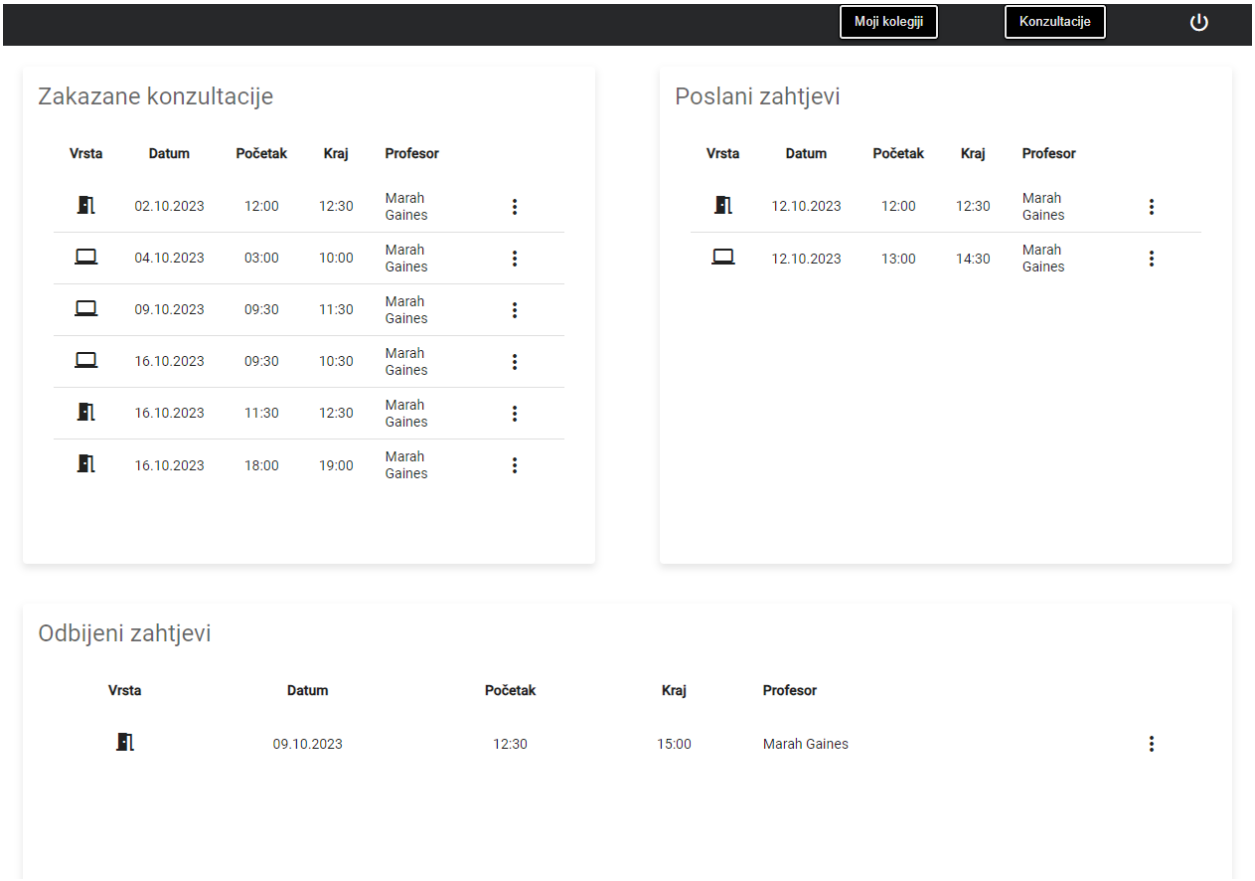
Sljedeći izazov je bio implementirati Google Calendar API u projekt. Prije nego što je moguće koristiti Google Calendar API, potrebno je izvršiti nekoliko pripremnih koraka. To uključuje stvaranje novog projekta u Google Cloud Console, omogućavanje Calendar API-a i preuzimanje JSON konfiguracijske datoteke koja sadrži ključeve za autentifikaciju. Google Calendar API koristi OAuth 2.0 protokol za autentifikaciju. Prvi korak u ovom procesu je preusmjeravanje korisnika na Googleovu OAuth 2.0 stranicu za prijavu. Nakon uspješne prijave, korisnik je preusmjeren natrag na aplikaciju s autorizacijskim kodom, koji se zatim koristi za dobivanje pristupnog tokena. Nakon uspješne autentifikacije, API se može koristiti za izvršavanje različitih operacija na korisnikovom kalendaru. To uključuje stvaranje, ažuriranje i brisanje događanja,

kao i dohvat informacija o postojećim događanjima i kalendarima. Upravljač koji je zadužen za interakciju sa Google Calendar API sadrži konstruktor u kojem se inicijalizira Google klijent i postavlja putanja do JSON datoteke s tajnim klijentskim podacima koje smo dobili pomoću Google Cloud Console. *GetAuthUrl()* metoda generira URL za OAuth2 autentifikaciju i vraća ga kao JSON odgovor. Zatim *postLogin()* metoda se koristi za prijavu korisnika putem Google OAuth2. Ako korisnik već postoji, ažurira se njegov pristupni token. Ako ne, stvara se novi korisnik. Potrebna je bila i metoda *getUserClient()* koja vraća Google klijenta koji je autentificiran za trenutno prijavljenog korisnika. Kada profesor prihvati zahtjev za konzultacije izvršava se *addEvent()* metoda koja dodaje novi događaj u Google kalendar. Ako je online tip konzultacija, dodaje se i Google Meet link.

4.4. Izrada klijentske strane

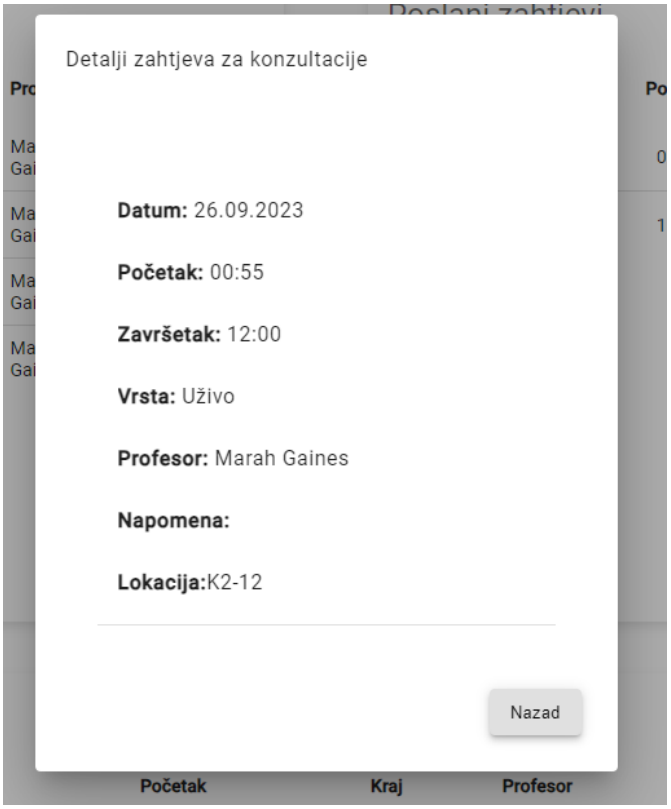
Razvoj klijentske strane je ključna komponenta u izradi modernih web aplikacija, s naglaskom na stvaranju vizualno privlačnog i interaktivnog korisničkog sučelja. Ova komponenta služi kao poveznica između korisnika i tehničke infrastrukture aplikacije, pružajući korisnicima jednostavan i ugodan način za interakciju s dostupnim sadržajem i funkcijama. Razvoj klijentske strane je esencijalna komponenta u izradi modernih web aplikacija, s naglaskom na stvaranju vizualno privlačnog i interaktivnog korisničkog sučelja. Ova komponenta služi kao poveznica između korisnika i tehničke infrastrukture aplikacije, pružajući korisnicima jednostavan i ugodan način za interakciju s dostupnim sadržajem i funkcijama.

Prva komponenta koja je izrađena je navigacijska traka. U ovom slučaju je ona vrlo jednostavna jer se sastoji samo od jednog gumba za odjavu sa korisničkog računa te dva gumba za odlazak na komponentu "Moji kolegiji" i "Konzultacije". Dizajn je minimalistički i moderan, a korištene su bijela, crna i nijanse sive boje. Nakon što je napravljena komponenta za navigaciju, potrebno je bilo napraviti stranicu za prijavu korisnika. Korisnik se može prijaviti unoseći svoju adresu elektronske pošte i zaporke. Prilikom odabira gumba "Prijavi se" dolazi do provjere valjanosti podataka, odnosno jesu li uneseni potrebni podaci, ako jesu tada se šalje zahtjev na API gdje se pokreće metoda login koja provjerava da li u bazi podataka postoji korisnik sa danom kombinacijom email-a i zaporke. Ako postoji tada se sprema token u lokalnu memoriju te se otvara komponenta "*Dashboard*". Ovisno o tome koja je uloga korisnika, mijenjaju se i mogućnosti. "*Dashboard*" komponenta izgleda isto za studenta i profesora, a izgled možemo vidjeti na slici 4.5..



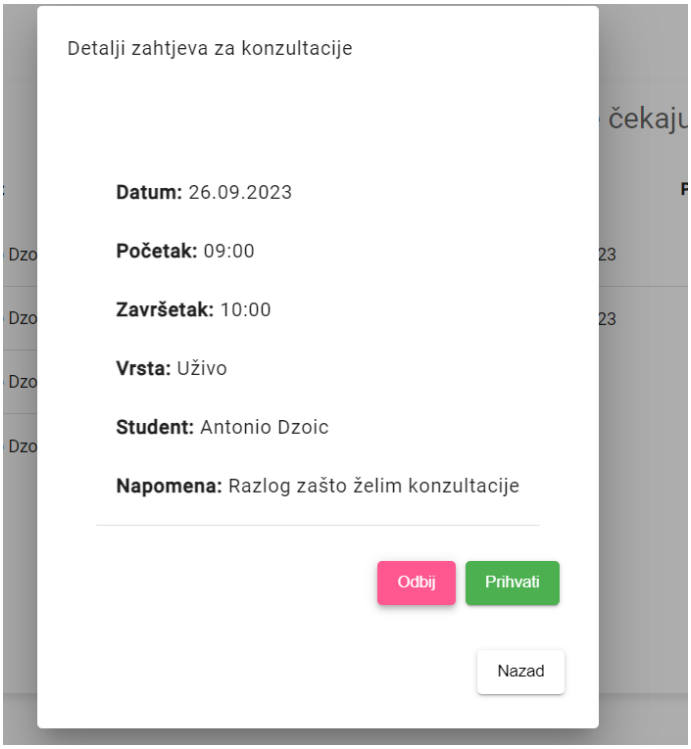
Slika 4.5. Izgled početnog zaslona prijavljenog korisnika

Komponenta je podijeljena na tri dijela, a to su zakazane konzultacije, poslani zahtjevi i odbijeni zahtjevi. Zakazane konzultacije predstavljaju nadolazeće termine dogovorenih konzultacija, klikom na tri točkice, na skroz desnoj strani, se otvara izbornik sa opcijom "Detalji". Izbornik je isti za sve tri tablice. Skroz lijevo možemo vidjeti dvije ikone: vrata i prijenosno računalo.Vrata predstavljaju konzultacije koje će se izvoditi uživo dok prijenosno računalo predstavlja termin konzultacija koje će se izvoditi preko Google Meet platforme. Zatim u tablici imao još osnovne podatke poput datuma, vremena početka konzultacija, vremena završetka konzultacija te profesora kod kojeg su dogovorene konzultacije. Ovakav prikaz vidi i profesor, ali on u tablici umjesto profesora, može vidjeti ime i prezime studenta s kojim ima dogovoren termin konzultacija. Prilikom odabira gumba "Detalji" otvara se dijalog prozor koji prikazuje dodatne informacije o zahtjevu za konzultacije. Ovisno o različitim slučajevima, prikazuju se različite informacije. U slučaju da su dogovorene konzultacije u uredu i korisnik koji otvara detaljni prikaz ima ulogu studenta, vidjet će prikaz kao na slici 4.6.



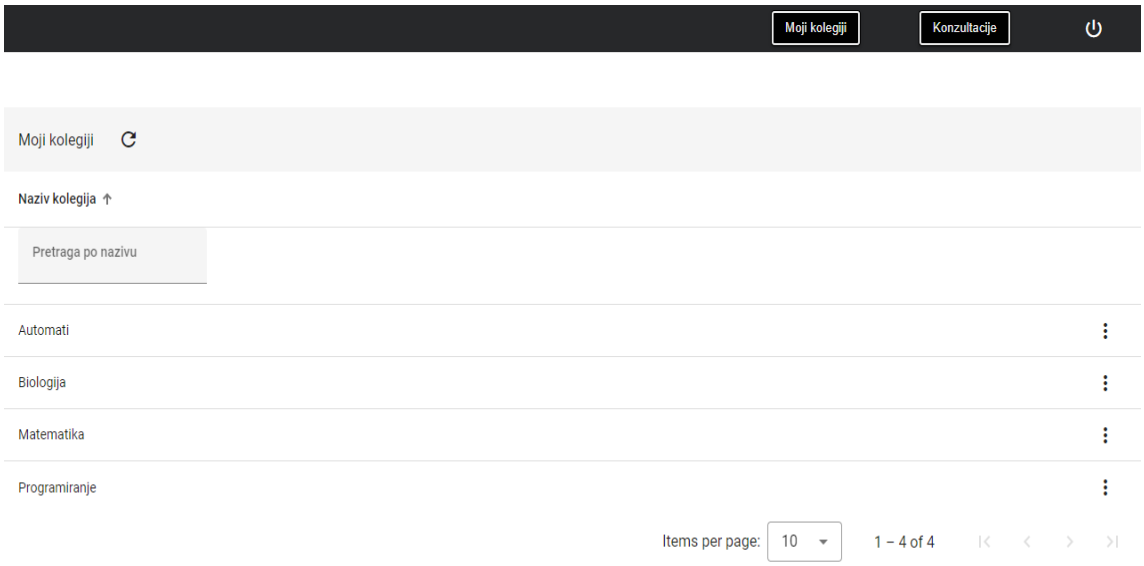
Slika 4.6. Detaljni prikaz zahtjeva za konzultacije

U slučaju da su dogovorene konzultacije online, tada umjesto lokacije će pisati internetska veza za Google Meet sobu. U slučaju da je zahtjev za konzultacije odbijen, umjesto lokacije ili internetske veze za Google Meet sobu pisat će razlog odbijanja. Isti prikaz je i u slučaju profesora osim kada profesor detaljno otvara poslane zahtjeve. U tom slučaju mu se otvara dijalog prozor kao na slici 4.7. gdje se nalaze dva dodatna gumba za prihvaćanje i odbijanje zahtjeva.



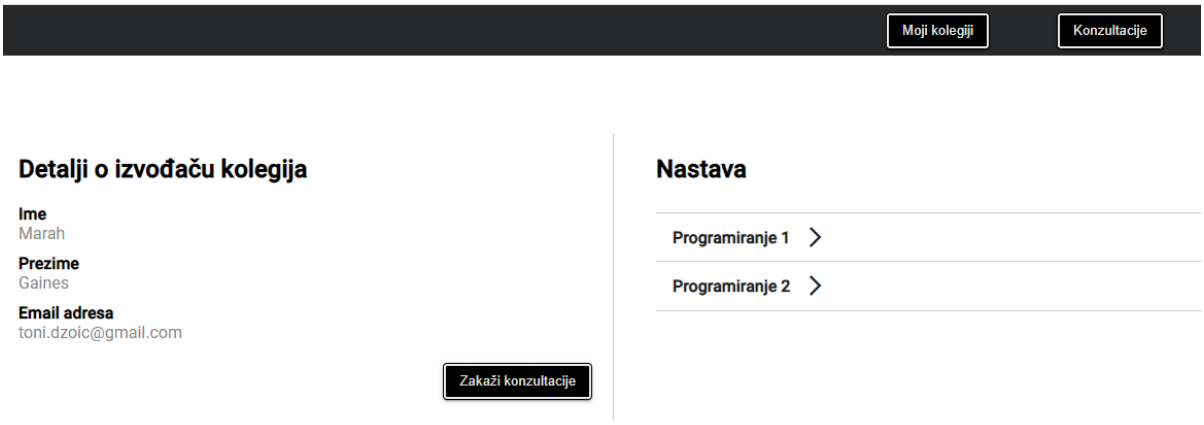
Slika 4.7. Detaljni prikaz neodgovorenog zahtjeva za konzultacije

Student osim što može vidjeti komponentu sa konzultacijama, može vidjeti i komponentu sa tablicom svih kolegija u koje je upisan. Izgled tablice se može vidjeti na slici 4.8..



Slika 4.8. Prikaz tablice "Moji kolegiji"

Ovdje također postoji gumb za prikaz detaljnih informacija o kolegiju koji sadrži i popis izvođača kolegija. Klikom na strelicu pored imena izvođača kolegija otvara se komponenta sa detaljnim informacijama o profesoru, na slici 4.9., gdje se nalazi i gumb "Zakaži konzultacije".

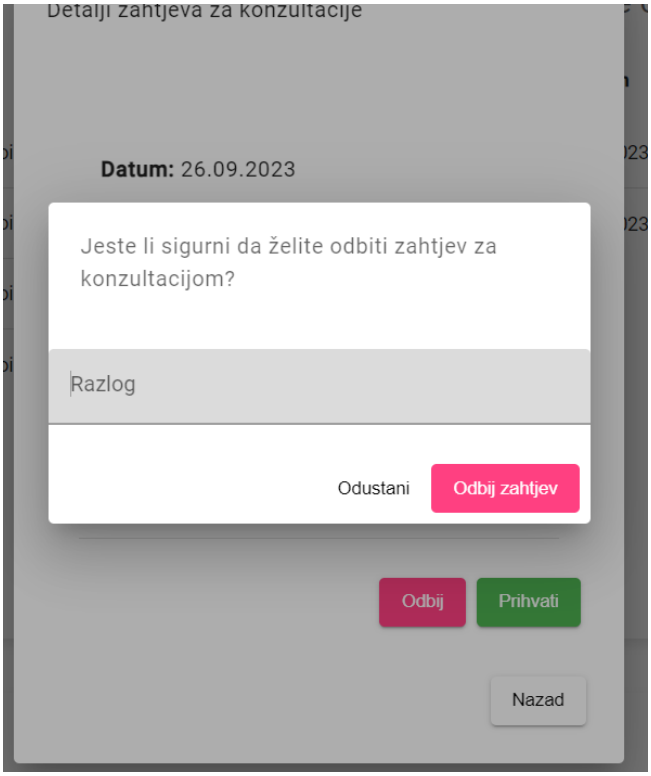


Slika 4.9. Prikaz detalja o izvođaču kolegija

Odabirom tog gumba otvara se dijalog prozor u kojem se odabire datum konzultacija, nakon čega se otvaraju dodatni podaci, a to su: vrijeme početka, vrijeme završetka, vrsta i napomena. Sva su polja obavezna osim polja "Napomena". Prilikom izrade ove komponente trebalo je voditi računa o tome da se filtriraju zauzeti termini kako ne bi došlo do preklapanja termina.

Slika 4.10. Prikaz forme za zakazivanje konzultacija

U slučaju kada profesor želi odbiti termin konzultacija otvara mu se dijalog prozor kao na slici 4.10. . Razlog nije obavezan te se polje može ostaviti prazno, isto vrijedi i za slučaj lokacije kada profesor prihvata termin konzultacija koje će se održavati uživo.



Slika 4.11. Prikaz prozora kada profesor želi odbiti zahtjev

5. ZAKLJUČAK

Web aplikacija za online dogovaranje konzultacija uspješno zadovoljava sve postavljene funkcionalne i nefunkcionalne zahtjeve. Kroz detaljan dizajn i planiranje, uspostavljena je robusna i skalabilna arhitektura koja omogućuje učinkovito upravljanje konzultacijama između studenata i profesora.

Sustav je izrađen korištenjem modernih tehnologija kao što su PHP, Laravel, Angular i MySQL, što ga čini fleksibilnim i lako održivim. Integracija s Google kalendarom i Google Meet platformom dodatno povećava funkcionalnost i korisničko iskustvo.Sigurnost i performanse su također ključni aspekti koji su pažljivo razmotreni tijekom razvoja, osiguravajući tako visoku razinu pouzdanosti i zaštite korisničkih podataka.

U budućnosti, aplikacija može biti proširena s dodatnim funkcionalnostima kao što su grupne konzultacije, integracija s drugim kalendarima i platformama za komunikaciju, te napredne analitičke funkcije za administratore. Kod je pisan uzimajući buduće moguće nadogradnje što je zahtijevalo puno truda kako bi kod ostao čist i lako razumljiv. Također bi se razvojem mobilne aplikacije olakšao pristup aplikaciji te bi povećao korisničku fleksibilnost.

Ovaj projekt predstavlja sveobuhvatno rješenje koje olakšava proces dogovaranja i održavanja konzultacija, čime se unapređuje komunikacija i suradnja između studenata i profesora na akademskim institucijama. Kombiniranjem Google kalendara i Google Meets-a stvorena je web aplikacija koje na jednostavan način uklanja probleme poput punog pretinca električne pošte, smanjuje se mogućnost da profesor zbog velikog broja konzultacija zakaže više termina u isto vrijeme ili da zaboravi na konzultacije zato što će ga podsjetnik podsjetiti 30 minuta ranije što je dovoljno vremena da odgodi konzultacije na vrijeme.

LITERATURA

[1] Benedictine University, Book my professor, dostupno na: https://ben.edu/book-my-professor , [Datum posjete: 15.09.2023].

[2] Calendly, dostupno na: https://calendly.com , [Datum posjete: 15.09.2023].

[3] Sign in Scheduling, dostupno na: https://10to8.com , [Datum posjete: 15.09.2023].

[4] YouCanBook, dostupno na: https://youcanbook.me , [Datum posjete: 15.09.2023].

[5] Doodle Poll, dostupno na: https://doodle.com, [Datum posjete: 15.09.2023].

[6]] M. Cohen, R. Krishnamoorthi, Angular 4: From Theory to Practice, Apress, 2017.

[7] Y. Fain, A. Moiseev, V. Rasputnis, Angular Development with TypeScript, Manning Publications, 2019.

[8] Griffith, Material Design Implementation with AngularJS: UI Component Framework, Apress, 2018.

[9] R. Lerdorf, K. Tatroe, Programming PHP, O'Reilly Media, 2013.

[10] T. Reenskaug, Models-Views-Controllers, Xerox PARC, 1979.

[11] T. Otwell, Laravel: From Apprentice To Artisan, Self-published, 2011.

[12] Laravel Docs, Eloquent: Getting Started, 2023, dostupno na: https://laravel.com/docs/10.x/eloquent [Datum posjete: 21.07.2023].

[13] Nilsen, Laravel 5 Essentials, 2015.

[14] S. Chacon, B. Straub, Pro Git, Apress, 2014.

[15] Google Developers, Google Calendar API Overview, 2023, dostupno na: https://developers.google.com/calendar/api/guides/overview, [Datum posjete: 10.09.2023].

[16] Medium, Querying calendar events via the Google Calendar API, 2022, dostupno na: https://chrieke.medium.com/querying-calendar-events-via-the-google-calendar-api-771782e24f62, [Datum posjete: 10.09.2023].

[17] FitSmallBuisness, Google Meet Review Features & Alternatives for 2023, dostupno na: https://fitsmallbusiness.com/google-meet-review/, [Datum posjete: 11.09.2023].

[18] Denis Sinyukov, How to Integrate Google Calendar API with Laravel, 2022, dostupno na: https://dev.to/dnsinyukov/how-to-integrate-google-calendar-api-and-friendship-with-laravel-4cok, [Datum posjete: 11.09.2023].

[19] P. DuBois, MySQL, Addison-Wesley, 2008.

[20] M. Stauffer, Laravel Up and Running, O'Reilly Media, 2016.

[21] Brown, Agile Development with Code First, Agile Books, London, 2020.

SAŽETAK

Ovaj rad se bavi problematikom organizacije konzultacija između studenata i profesora na akademskim institucijama. U cilju optimizacije ovog procesa, razvijena je web aplikacija koja koristi najsavremenije tehnologije, uključujući PHP, Laravel, Angular i MySQL. Aplikacija je dizajnirana da bude robusna, skalabilna i lako održiva, s posebnim fokusom na sigurnost i performanse. Integracijom s Google kalendarom i Google Meet platformom, aplikacija nudi proširene funkcionalnosti i poboljšava korisničko iskustvo. Osim što zadovoljava sve postavljene funkcionalne i nefunkcionalne zahtjeve, aplikacija pruža osnovu za buduće nadogradnje i proširenja. Kroz ovaj rad, čitatelj će dobiti uvid u ključne aspekte razvoja aplikacije, od arhitekture i tehnologija do sigurnosnih mjera i mogućnosti za buduće nadogradnje.

Ključne riječi: angular, baza podataka, laravel, poslužitelj, web aplikacija

ABSTRACT

Web application for online consultation scheduling

This paper addresses the issue of organizing consultations between students and professors at academic institutions. To optimize this process, a web application has been developed using state-of-the-art technologies, including PHP, Laravel, Angular, and MySQL. The application is designed to be robust, scalable, and easily maintainable, with a special focus on security and performance. By integrating with Google Calendar and the Google Meet platform, the application offers extended functionalities and enhances the user experience. In addition to meeting all the set functional and non-functional requirements, the application provides a foundation for future upgrades and expansions. Through this paper, the reader will gain insight into the key aspects of the application's development, from architecture and technologies to security measures and possibilities for future upgrades.

Keywords: Angular, database, Laravel, server, web application

PRILOZI

Prilog 1: Dokumentacija diplomskog rada na CD-u

Prilog 2: Pristup kodu izrađene aplikacije (klijentska strana): https://github.com/Aplexas99/diplomski/tree/master

Prilog 3: Pristup kodu izrađene aplikacije (API): https://github.com/Aplexas99/diplomski-api/tree/main