

AKADEMIJA STRUKOVNIH STUDIJA JUŽNA SRBIJA

Odsek Visoka poslovna škola Leskovac



Završni rad

WEB APLIKACIJA: OD RAZVOJA DO PRODUKCIJE

Mentor: spec. inž. Lazar Krstić

Student: Lazar Vujovic

Smer: Poslovna informatika i e-biznis

Broj indeksa: 285/23-01

Leskovac, 2026. godine

Web aplikacija: od razvoja do produkcije

Sažetak: *U ovom završnom radu obrađen je proces razvoja i pripreme web aplikacije za rad u serverskom okruženju. Praktični primer predstavlja aplikacija WorkHub, namenjena evidenciji korisnika, organizaciji projekata i praćenju zadataka u manjem poslovnom timu. Pristup funkcionalnostima određen je ulogama administratora, menadžera i korisnika.*

Za razvoj aplikacije koristi se Next.js, dok se podaci čuvaju u PostgreSQL bazi, kojoj aplikacija pristupa pomoću alata Prisma. Prijavljivanje i upravljanje korisničkim sesijama ostvareni su pomoću biblioteke Better Auth. U radu su predstavljeni zahtevi sistema, organizacija baze podataka i realizacija osnovnih funkcionalnosti.

Poseban naglasak stavljen je na infrastrukturu potrebnu za objavljivanje aplikacije. Obrađuju se izbor i zakup virtuelnog servera, udaljeni pristup putem SSH protokola, upravljanje izvornim kodom pomoću Git-a i pokretanje aplikacije i baze u Docker kontejnerima. Predstavljeni su i povezivanje internet domena, podešavanje DNS zapisa, uloga Nginx servera i omogućavanje zaštićene komunikacije putem HTTPS-a. Završni deo obuhvata postupke provere rada, ažuriranja i izrade rezervnih kopija. Rad povezuje razvoj poslovne aplikacije sa pripremom okruženja za njenu dostupnost i održavanje na internetu.

Ključne reči: *web aplikacija, poslovni informacioni sistem, serverska infrastruktura*

Web Application: From Development to Production

Summary: *This final thesis examines the development of a web application and its preparation for operation in a server environment. The practical example is WorkHub, an application designed to manage users, organize projects, and track tasks within a small business team. Access to its features is determined by administrator, manager, and user roles.*

The application uses Next.js, while its data is stored in a PostgreSQL database accessed through Prisma. Better Auth handles sign-in and user sessions. The thesis presents the system requirements, database structure, and implementation of the core features.

Particular emphasis is placed on the infrastructure required to deploy the application. The thesis covers selecting and renting a virtual server, remote access through SSH, source code management with Git, and running the application and database in Docker containers. It also addresses domain configuration, DNS records, the role of Nginx, and secure communication through HTTPS. The final part covers procedures for verification, updates, and database backups. The thesis connects business application development with the preparation of an environment for its availability and maintenance on the internet.

Keywords: *web application, business information system, server infrastructure*

SADRŽAJ

UVOD	2
1. ANALIZA ZAHTEVA I PROJEKTOVANJE APLIKACIJE	3
1.1. Poslovna namena aplikacije	3
1.2. Funkcionalni i nefunkcionalni zahtevi	4
1.3. Korisničke uloge i prava pristupa	5
1.4. Model podataka i veze između entiteta	6
1.5. Arhitektura sistema	8
2. TEHNOLOGIJE I ALATI ZA REALIZACIJU SISTEMA	9
2.1. Next.js	9
2.2. PostgreSQL i Prisma	9
2.3. Better Auth i Zod	10
2.4. Git i udaljeni repozitorijum	10
2.5. Linux i SSH	10
2.6. Docker i Docker Compose	11
2.7. Nginx	11
3. RAZVOJ WEB APLIKACIJE	12
3.1. Priprema razvojnog okruženja i organizacija projekta	12
3.2. Kreiranje baze podataka i povezivanje sa aplikacijom	13
3.3. Prijava korisnika i kontrola pristupa	14
3.4. Evidencija korisnika	15
3.5. Upravljanje projektima	16
3.6. Kreiranje, dodeljivanje i praćenje zadataka	16
3.7. Korisnički interfejs i validacija podataka	18
3.8. Lokalno testiranje aplikacije	18
4. IZBOR I PRIPREMA SERVERSKOG OKRUŽENJA	20
4.1. Zahtevi aplikacije i izbor načina hostovanja	20
4.2. Izbor i zakup VPS servera	20
4.3. Izbor operativnog sistema	22
4.4. Povezivanje sa serverom putem SSH protokola	22
4.5. Podešavanje korisničkih naloga i SSH ključeva	22
5. KONTEJNERIZACIJA I POSTAVLJANJE APLIKACIJE	24
5.1. Preuzimanje izvornog koda (source code) na server	24
5.2. Docker konfiguracija aplikacije i baze podataka	24
5.3. Promenljive okruženja i trajno čuvanje podataka	25
5.4. Pokretanje aplikacije	26
6. POVEZIVANJE DOMENA I OBEZBEĐIVANJE WEB PRISTUPA	27
6.1. Registracija domena i DNS konfiguracija	27
6.2. Podešavanje Nginx obrnutog proksija (reverse proxy)	28
6.3. Izdavanje TLS sertifikata	29
6.4. HTTPS pristup aplikaciji	29
7. TESTIRANJE I ODRŽAVANJE SISTEMA	30
7.1. Funkcionalno testiranje aplikacije	30
7.2. Provera rada Docker servisa i logova	30
7.3. Ažuriranje aplikacije	31
7.4. Rezervne kopije (backup) i mogućnosti unapređenja	31
ZAKLJUČAK	32
LITERATURA	33

UVOD

Organizacija poslovnih aktivnosti zahteva jasan pregled zaposlenih, projekata i zadataka. Kada se podaci o obavezama, rokovima i odgovornim osobama vode u odvojenim tabelama, dokumentima i porukama, njihovo praćenje i ažuriranje može biti otežano. Web aplikacija omogućava objedinjavanje tih podataka i pristup putem internet pregledača, čime olakšava svakodnevnu organizaciju rada.

Predmet ovog završnog rada je razvoj i postavljanje manje poslovne web aplikacije pod nazivom WorkHub, namenjene upravljanju korisnicima, projektima i zadacima. Aplikacija omogućava formiranje projektnih timova, dodeljivanje obaveza i praćenje njihove realizacije. Prava pristupa određena su ulogama administratora, menadžera i korisnika. Administrator upravlja korisničkim nalogima i svim projektima, menadžer organizuje rad na sopstvenim projektima, dok korisnik prati dodeljene zadatke i menja njihov status.

Tema „Web aplikacija: od razvoja do produkcije“ obuhvata i pripremu uslova za korišćenje aplikacije na internetu. Aplikacija koja radi na računaru na kojem je razvijena još nije spremna za pristup krajnjih korisnika. Potrebno je obezbediti server, podesiti pristup, povezati bazu podataka i internet domen i zaštititi komunikaciju. Producersko okruženje u ovom radu označava okruženje u kojem je aplikacija postavljena za korišćenje putem interneta.

Cilj rada je da se kroz praktičan primer prikaže celokupan postupak, od definisanja zahteva i razvoja osnovnih funkcionalnosti do objavljivanja i održavanja aplikacije. Poseban naglasak stavljen je na serversku infrastrukturu i razumevanje uloge njenih komponenti. Obim poslovnih funkcionalnosti ograničen je na potrebe manjeg tima, kako bi dovoljno prostora bilo posvećeno postavljanju sistema i proveru njegovog rada.

Za razvoj aplikacije koristi se Next.js, dok se podaci čuvaju u PostgreSQL bazi. Prisma olakšava pristup podacima i upravljanje promenama strukture baze, a Better Auth omogućava prijavljivanje i upravljanje korisničkim sesijama. Infrastrukturni deo obuhvata zakup virtuelnog servera, podešavanje Linux okruženja i udaljeni pristup putem SSH protokola. Git se koristi za praćenje verzija izvornog koda, dok Docker omogućava pokretanje aplikacije i baze u odvojenim kontejnerima. Povezivanje domena, podešavanje Nginx-a i omogućavanje HTTPS pristupa zaokružuju postupak objavljivanja.

U radu se primenjuju analiza zahteva, projektovanje baze podataka, praktična implementacija i testiranje. Postupci se objašnjavaju kroz primere iz aplikacije, dijagrame, snimke ekrana i relevantne delove konfiguracije. Provera obuhvata osnovne funkcionalnosti, korisnička ovlašćenja i rad u serverskom okruženju. Razmatraju se i ažuriranje aplikacije, čuvanje rezervnih kopija i oporavak podataka, kao aktivnosti potrebne za njeno održavanje.

Rad je organizovan tako da prati tok realizacije sistema. Nakon analize zahteva predstavljene su korišćene tehnologije i razvoj aplikacije. Zatim se obrađuju priprema servera, kontejnerizacija, povezivanje domena i zaštita web pristupa. Završni deo posvećen je testiranju, održavanju, troškovima i ograničenjima rešenja. Na taj način povezuju se znanja iz poslovne informatike, razvoja informacionih sistema i administracije serverskog okruženja.

1. ANALIZA ZAHTEVA I PROJEKTOVANJE APLIKACIJE

Projektovanje aplikacije obuhvata određivanje njene poslovne namene, funkcionalnosti, korisničkih ovlašćenja i načina organizacije podataka. Ovim postupkom definišu se pravila koja sistem treba da primenjuje, kao i osnova za razvoj, testiranje i postavljanje aplikacije u produkcijsko okruženje.

Aplikacija obrađena u ovom radu namenjena je upravljanju korisnicima, projektima i zadacima u manjem poslovnom timu. Sistem omogućava formiranje projektnih timova, dodeljivanje zadataka i praćenje njihove realizacije. Pristup podacima i mogućnost izvršavanja pojedinih radnji zavise od korisničke uloge, odgovornosti za projekat i zaduženja za konkretan zadatak.

Posebna pažnja posvećena je serverskoj proveri identiteta i ovlašćenja, validaciji podataka i odvajanju poslovnih podataka od konfiguracije okruženja. Ovi zahtevi povezuju projektovanje aplikacije sa njenim kasnijim postavljanjem na server.

1.1. Poslovna namena aplikacije

Organizacija rada na projektima zahteva pregled učesnika, raspodelu odgovornosti i praćenje rokova. Kada se podaci o ovim aktivnostima vode u nepovezanim tabelama, dokumentima i porukama, otežano je utvrđivanje trenutnog stanja projekta. Promene zaduženja ili rokova mogu ostati nedostupne pojedinim članovima tima, što otežava koordinaciju rada.

Poslovna namena aplikacije jeste objedinjavanje ovih podataka u jedinstven sistem. Svaki projekat ima odgovornog menadžera i članove, dok se zadaci dodeljuju pojedinačnim izvršiocima. Korisnici na taj način dobijaju pregled svojih obaveza, a menadžeri pregled aktivnosti na projektima za koje su odgovorni.

Osnovni poslovni proces počinje kreiranjem projekta i određivanjem njegovog menadžera. Administrator može da odredi menadžera projekta, dok menadžer kreira projekat za koji je sam odgovoran. Nakon toga se određuju članovi projektnog tima i kreiraju zadaci. Izvršilac zadatka mora biti aktivan korisnik i član odgovarajućeg projekta.

Tokom realizacije korisnik ažurira status svog zadatka. Menadžer prati rad na sopstvenim projektima i po potrebi menja podatke o zadacima, dok administrator ima pregled i mogućnost upravljanja svim projektima.

Status projekta određuje se ručno, a napredak se izračunava na osnovu zadataka. Time su razdvojeni poslovna odluka o stanju projekta i pokazatelj realizacije njegovih aktivnosti. Završavanje zadataka zato ne mora automatski da promeni status projekta.

Obim aplikacije ograničen je na upravljanje korisnicima, projektima, članstvom i zadacima. Slanje elektronske pošte, javna registracija, prilozi i razmena poruka nisu obuhvaćeni realizovanim rešenjem.

1.2. Funkcionalni i nefunkcionalni zahtevi

Funkcionalni zahtevi određuju operacije koje sistem omogućava korisnicima. Njihovo izvršavanje zavisi od korisničke uloge i odnosa korisnika prema konkretnom projektu ili zadatku.

Tabela 1.1. Funkcionalni zahtevi aplikacije

Oznaka	Zahtev	Opis
FZ-01	Prijava i odjava	Prijavljivanje postojećim nalogom, uspostavljanje sesije i odjava iz sistema.
FZ-02	Upravljanje korisnicima	Administrator pregleda korisnike, kreira naloge i menja njihove uloge i aktivnost.
FZ-03	Zaštita poslednjeg administratora	Sistem sprečava promene kojima bi ostao bez aktivnog administratora.
FZ-04	Upravljanje projektima	Administrator upravlja svim projektima, a menadžer samo sopstvenim projektima.
FZ-05	Upravljanje članstvom	Administrator i nadležni menadžer upravljaju članovima projekta.
FZ-06	Upravljanje zadacima	Ovlašćeni korisnici kreiraju i menjaju zadatke i dodeljuju ih aktivnim članovima projekta.
FZ-07	Pregled dodeljenih zadataka	Korisnik pregleda sopstvene zadatke na projektima kojima ima pristup.
FZ-08	Promena statusa zadatka	Korisnik menja status sopstvenog zadatka, u skladu sa poslovnim pravilima.
FZ-09	Praćenje realizacije	Sistem prikazuje izračunati napredak i prepoznaje zadatke kojima je istekao rok.
FZ-10	Upravljanje profilom	Svaki prijavljeni korisnik može da uređuje dozvoljene podatke svog profila i menja sopstvenu lozinku.
FZ-11	Validacija podataka	Sistem proverava format unosa, rokove, članstvo, aktivnost izvršioca i ograničenja zaključanog projekta.

Nefunkcionalni zahtevi odnose se na zaštitu podataka, pouzdanost i mogućnost održavanja sistema.

Tabela 1.2. Nefunkcionalni zahtevi aplikacije

Oznaka	Zahtev	Opis
NZ-01	Jednostavnost korišćenja	Interfejs omogućava pregled projekata i zadataka kroz dosledne stranice, forme i poruke.
NZ-02	Serverska kontrola pristupa	Identitet i ovlašćenja proveravaju se pri učitavanju zaštićenih podataka i izvršavanju izmena.
NZ-03	Zaštita lozinki i tajni	Lozinke se čuvaju kao heš vrednosti, a tajne konfiguracije izdvajaju se iz repozitorijuma.
NZ-04	Zaštićena komunikacija	Produkcijski pristup predviđen je preko HTTPS-a, uz odgovarajuću konfiguraciju sesijskih kolačića.
NZ-05	Integritet podataka	Validacija i ograničenja baze sprečavaju neispravne zapise i nedozvoljene veze.
NZ-06	Dosledno računanje datuma	Kalendarski datumi i kašnjenja obrađuju se prema vremenskoj zoni Europe/Belgrade.
NZ-07	Trajno čuvanje podataka	Podaci baze čuvaju se nezavisno od životnog ciklusa kontejnera.
NZ-08	Ponovljivost postavljanja	Docker konfiguracija i migracije omogućavaju dokumentovano postavljanje sistema.
NZ-09	Održavanje	Organizacija koda i konfiguracije omogućava ažuriranje aplikacije i praćenje njenog rada.

Validacija se sprovodi na više nivoa. U pregledaču se koriste osnovna ograničenja obrazaca, kao što su obavezna polja i minimalna dužina unosa. Serverska validacija koristi Zod šeme, nakon čega se proveravaju poslovna pravila. Ispravan format podataka sam po sebi ne znači da je operacija dozvoljena: korisnik, na primer, može poslati ispravan identifikator zadatka za čiju izmenu nema ovlašćenje.

1.3. Korisničke uloge i prava pristupa

U sistemu postoje tri korisničke uloge: administrator, menadžer i korisnik. Ovlašćenja se određuju kombinovanjem uloge i odnosa prema konkretnim podacima.

Administrator upravlja korisničkim nalogima i ima pristup svim projektima i zadacima. Može da menja korisničke uloge i aktivnost naloga, uz ograničenje kojim se čuva najmanje jedan aktivan administrator.

Menadžer upravlja projektima za koje je odgovoran. U okviru tih projekata može da uređuje podatke, upravlja članstvom i zadacima. Sama menadžerska uloga ne daje mu pravo upravljanja projektima drugih menadžera.

Korisnik pristupa projektu ako je njegov član. Može da pregleda članove projekta, dok je pregled zadataka ograničen na zadatke koji su mu dodeljeni. Kod sopstvenih zadataka može da menja status, ali ne i ostala polja.

Tabela 1.3. Matrica korisničkih ovlašćenja

Radnja	Administrator	Menadžer	Korisnik
Pregled svih korisnika i kreiranje naloga	Da	Ne	Ne
Izmena uloge i aktivnosti naloga	Da, uz zaštitu poslednjeg administratora	Ne	Ne
Kreiranje projekta	Da, uz izbor menadžera	Da, kao sopstvenog projekta	Ne
Izmena i brisanje projekta	Svi projekti	Sopstveni projekti	Ne
Pregled projekta	Svi projekti	Sopstveni projekti	Projekti čiji je član
Upravljanje članovima i zadacima	Svi projekti	Sopstveni projekti	Ne
Pregled članova projekta	Svi projekti	Sopstveni projekti	Projekti čiji je član
Pregled zadataka	Svi zadaci	Zadaci na sopstvenim projektima	Sopstveni zadaci
Izmena svih polja zadatka	Da	Na sopstvenim projektima	Ne
Izmena statusa zadatka	Da	Na sopstvenim projektima	Samo sopstveni zadaci
Izmena profila i sopstvene lozinke	Da	Da	Da

Kontrola pristupa sprovodi se pri učitavanju stranica i u serverskim akcijama. Prikaz ili skrivanje elemenata interfejsa služi prilagođavanju dostupnih opcija korisniku, dok se odluka o dozvoljenosti operacije donosi na serveru.

Za proveru odgovornosti menadžera upoređuju se identifikator prijavljenog korisnika i identifikator menadžera projekta. Pri dodeljivanju zadatka proveravaju se aktivnost izvršioca i njegovo članstvo u projektu. Pri promeni statusa koju zahteva korisnik proverava se da li je on izvršilac konkretnog zadatka.

Korisnici se deaktiviraju umesto brisanja. Time se zadržavaju veze sa prethodnim projektima i zadacima, dok deaktivirani nalog gubi pravo pristupa aplikaciji.

1.4. Model podataka i veze između entiteta

Model podataka obuhvata poslovne entitete i podatke potrebne za prijavljivanje korisnika. Poslovni deo čine korisnici, projekti, članstva na projektima i zadaci. Podaci za autentifikaciju obuhvataju naloge i sesije kojima upravlja Better Auth.

Tabela u nastavku prikazuje logički model i ulogu podataka, bez navođenja svih implementacionih polja.

Tabela 1.4. Osnovne celine modela podataka

Entitet	Namena i ključni podaci
Korisnik	Identitet korisnika, podaci profila, uloga i aktivnost.
Projekat	Podaci o projektu, odgovorni menadžer, status i rokovi.
Članstvo na projektu	Veza između korisnika i projekta kojem pripada.
Zadatak	Podaci o zadatku, pripadajući projekat, izvršilac, status, prioritet i rok.
Nalog	Podaci potrebni za autentifikaciju, uključujući heš lozinke kod prijave lozinkom.
Sesija	Podaci kojima se prati važenje prijave i povezanost sesije sa korisnikom.

Veza između menadžera i projekta je tipa jedan prema više. Jedan menadžer može biti odgovoran za više projekata, dok svaki projekat ima jednog odgovornog menadžera, određenog poljem `menadzerId`.

Veza između korisnika i projekata preko članstva je tipa više prema više. Korisnik može biti član više projekata, a jedan projekat može imati više članova. Članstvo se zato evidentira odvojeno od zadataka. Korisnik može pripadati projektnom timu i pre nego što mu je dodeljen prvi zadatak.

Jedan projekat može sadržati više zadataka, dok svaki zadatak pripada jednom projektu. Zadatak ima jednog izvršioca, a isti korisnik može biti izvršilac više zadataka. Pri dodeljivanju mora biti ispunjeno pravilo da je izvršilac aktivan i da pripada projektnom timu.

Podaci o sesijama povezani su sa korisnikom. Sesija omogućava aplikaciji da prepozna prijavljenog korisnika, ali se za donošenje odluke o pristupu proverava i trenutno stanje njegovog naloga. Time deaktivacija naloga utiče na pristup i kada korisnik već ima uspostavljenu sesiju.

Za integritet podataka posebno su značajna sledeća pravila:

- Projekat mora imati odgovornog menadžera.
- Zadatak mora pripadati postojećem projektu.
- Izvršilac zadatka mora biti aktivan član tog projekta.
- Izmena korisničkih uloga i aktivnosti ne sme ukloniti poslednjeg aktivnog administratora.
- Operacije nad zaključanim projektom podležu posebnim poslovnim ograničenjima.
- Datumi i rokovi proveravaju se pre upisa ili izmene podataka.

Napredak projekta predstavlja izvedenu vrednost koja se računa iz zadataka. Kašnjenje se određuje poređenjem kalendarskih datuma u obliku `YYYY-MM-DD`, uz primenu vremenske zone Europe/Belgrade i pravila o statusu zadatka. Na taj način se razlikuju podaci koje korisnik unosi od pokazatelja koje sistem izračunava.

1.5. Arhitektura sistema

Aplikacija je razvijena korišćenjem Next.js tehnologije i App Router-a. Organizovana je kroz tri logičke celine: korisnički interfejs, serversku poslovnu logiku i sloj podataka.

Korisnički interfejs čine serverske i klijentske komponente. Serverske komponente učitavaju podatke i pripremaju prikaz stranica, dok klijentske omogućavaju interakciju kroz forme, dijaloge i navigaciju. Podaci iz formi prosleđuju se serverskim akcijama, koje proveravaju sesiju, korisnička ovlašćenja i ispravnost unosa pre izvršavanja operacije.

Pristup PostgreSQL bazi ostvaruje se preko Prisma klijenta. Za poslovne operacije koriste se Server Actions, bez posebnog API sloja. Prijavljivanje, sesije i odjava realizovani su pomoću Better Auth biblioteke. Kontrola pristupa izvršava se na serveru, pri učitavanju zaštićenih podataka i njihovoj izmeni.

U serverskom okruženju aplikacija i baza rade u odvojenim Docker kontejnerima, povezanim pomoću Docker Compose konfiguracije. Podaci baze čuvaju se u trajnom volumenu. Nginx prihvata HTTPS zahteve korisnika i prosleđuje ih aplikaciji, dok baza nije neposredno dostupna putem interneta.

Ovakva arhitektura objedinjuje razvoj i postavljanje sistema, uz jasno razdvojene odgovornosti komponenti i centralizovanu kontrolu pristupa podacima.

2. TEHNOLOGIJE I ALATI ZA REALIZACIJU SISTEMA

Za izradu i objavljivanje web aplikacije potrebno je povezati više tehnologija. Jedne omogućavaju prikaz stranica i obradu korisničkih zahteva, druge čuvanje podataka, dok se pojedini alati koriste za postavljanje aplikacije na server i njeno održavanje.

U ovom poglavlju predstavljeni su alati korišćeni u radu i objašnjena njihova uloga. Njihova konkretna podešavanja i postupci korišćenja prikazani su u narednim poglavljima.

2.1. Next.js

Kada korisnik otvori aplikaciju, pred njim se pojavljuju stranice, tabele, dugmad i obrasci za unos podataka. Iza tog prikaza nalaze se postupci koji učitavaju podatke, proveravaju zahteve i čuvaju izmene. Next.js omogućava razvoj korisničkog interfejsa i serverskog dela u okviru jednog projekta.

Next.js je razvojni okvir zasnovan na Javascript biblioteci React. Razvojni okvir predstavlja skup mogućnosti i pravila koji olakšavaju organizaciju aplikacije. React omogućava izradu interfejsa od komponenti, odnosno manjih celina kao što su navigacija, forma za prijavu ili tabela zadataka. Next.js tu osnovu dopunjuje mogućnostima za organizaciju stranica i izvršavanje koda na serveru.

U ovoj aplikaciji Next.js se koristi za prikaz korisnika, projekata i zadataka, kao i za obradu operacija nad tim podacima. Na primer, kada menadžer popuni formu za novi zadatak, serverski deo proverava podatke i njegova ovlašćenja, a zatim pokreće upis u bazu.

Za organizaciju stranica koristi se App Router, deo Next.js okvira koji povezuje strukturu projekta sa adresama aplikacije. Operacije poput kreiranja projekta ili promene statusa zadatka realizovane su pomoću serverskih akcija, odnosno funkcija koje se izvršavaju na serveru.

2.2. PostgreSQL i Prisma

Podaci koje korisnik unese moraju ostati sačuvani i nakon zatvaranja pregledača ili ponovnog pokretanja aplikacije. Za njihovo organizovano čuvanje koristi se baza podataka.

PostgreSQL je open source sistem za upravljanje relacionim bazama podataka. U relacionoj bazi podaci su organizovani u tabele koje mogu biti međusobno povezane. Takva organizacija omogućava čuvanje i povezivanje, na primer, korisnika, projekata i zadataka. PostgreSQL podržava i pravila kojima se sprečava unos neispravnih podataka.

U ovoj aplikaciji svaki zadatak povezan je sa projektom i korisnikom koji je zadužen za njegovo izvršenje. Zahvaljujući tim vezama moguće je prikazati zadatke određenog projekta ili obaveze konkretnog korisnika.

Za pristup bazi koristi se Prisma. Ovaj alat omogućava da se struktura podataka opiše u okviru projekta i da se iz programskog koda izvršavaju operacije čitanja, upisa i izmene. Prisma obuhvata i migracije, odnosno evidentirane promene strukture baze. One omogućavaju da se, na primer, dodavanje novog polja primeni i na serversku bazu prilikom objavljivanja nove verzije aplikacije.

PostgreSQL čuva podatke, dok Prisma olakšava način na koji aplikacija radi sa njima.

2.3. Better Auth i Zod

Pored rada sa poslovnim podacima, aplikacija treba da prepozna prijavljenog korisnika i proveriti podatke koje on unosi. Za ove potrebe koriste se Better Auth i Zod.

Better Auth je biblioteka za autentifikaciju, odnosno proveru identiteta korisnika. Omogućava prijavljivanje i upravljanje sesijama. Sesija predstavlja vezu između uspešne prijave i narednih zahteva korisnika, zahvaljujući kojoj nije potrebno ponovo unositi lozinku pri otvaranju svake stranice.

U aplikaciji Better Auth obrađuje prijavu i odjavu. Nakon utvrđivanja identiteta, aplikacija posebno proverava šta korisnik sme da radi. Na primer, uspešna prijava korisnika ne daje mu pravo da menja projekat drugog menadžera. Time se razlikuju provera identiteta i provera ovlašćenja.

Zod je biblioteka za proveru strukture i ispravnosti podataka tokom izvršavanja programa. Pravila se opisuju kroz šeme, kojima se određuje kakav unos se očekuje.

U ovom sistemu Zod se koristi za serversku proveru podataka iz obrazaca, kao što su obavezna polja i format imejl adrese. Nakon toga aplikacija proverava poslovna pravila, na primer da li izabrani izvršilac pripada projektu.

2.4. Git i udaljeni repozitorijum

Tokom razvoja aplikacije izvorni kod se često menja. Dodaju se funkcionalnosti, ispravljaju greške i prilagođavaju podešavanja. Zbog toga je važno sačuvati pregled promena i mogućnost poređenja različitih verzija.

Git je sistem za kontrolu verzija. Omogućava evidentiranje promena u datotekama, pregled istorije razvoja i rad sa različitim granama projekta. Sačuvana celina promena naziva se *commit* i obično sadrži kratku poruku koja opisuje izvršenu izmenu.

Repozitorijum predstavlja projekat sa istorijom njegovih promena. Pored lokalnog repozitorijuma na računaru, koristi se i udaljeni repozitorijum, dostupan preko mreže.

U ovom radu Git povezuje razvoj aplikacije sa njenim postavljanjem na server. Promene nastaju i proveravaju se u razvojnom okruženju, zatim se šalju u udaljeni repozitorijum i preuzimaju na server. Preuzimanje koda predstavlja jedan korak ažuriranja; nakon njega potrebno je izgraditi i pokrenuti odgovarajuću verziju aplikacije.

Lozinke i druge tajne vrednosti čuvaju se odvojeno od repozitorijuma.

2.5. Linux i SSH

Da bi aplikacija bila dostupna putem interneta, potrebno je računarsko okruženje u kojem će se izvršavati. U ovom radu za tu namenu koristi se VPS, odnosno virtuelni privatni server. Zakupom VPS-a dobija se izdvojeno virtuelno okruženje sa dodeljenim procesorskim resursima, memorijom i prostorom za podatke.

Na serveru se koristi operativni sistem zasnovan na Linux-u. Njegova uloga je da upravlja resursima računara i omogući izvršavanje aplikacije, baze i ostalih potrebnih programa.

Pošto se server nalazi na udaljenoj lokaciji, administriranje se obavlja preko SSH protokola. SSH omogućava zaštićeno povezivanje i izvršavanje komandi na udaljenom računaru. OpenSSH je skup alata koji omogućava ovakav pristup, uključujući prijavljivanje pomoću SSH ključeva.

U ovom projektu SSH se koristi za početno podešavanje servera, instalaciju softvera, preuzimanje koda i proveru rada aplikacije. SSH ključevi predstavljaju par povezanih ključeva: javni se postavlja na server, dok privatni ostaje kod osobe koja pristupa serveru.

2.6. Docker i Docker Compose

Aplikaciji su za rad potrebni odgovarajući programski paketi i podešavanja. Ako se okruženje na računaru za razvoj razlikuje od okruženja na serveru, mogu se pojaviti problemi pri pokretanju. Docker olakšava pripremu i prenošenje potrebnog izvršnog okruženja.

Docker koristi kontejnere, odnosno izolovana okruženja u kojima se izvršavaju programi. Kontejner se pokreće na osnovu unapred pripremljenog paketa, koji se u Docker terminologiji naziva image. Taj paket sadrži aplikaciju i potrebne zavisnosti, dok se način njegove izrade definiše u datoteci `Dockerfile`.

U ovom radu aplikacija i PostgreSQL baza izvršavaju se u odvojenim kontejnerima. Njihovo zajedničko podešavanje obavlja se pomoću Docker Compose alata. U jednoj konfiguracionoj datoteci definišu se servisi, njihove mrežne veze i prostor za trajno čuvanje podataka.

Podaci baze čuvaju se u Docker volumenu, prostoru koji ostaje sačuvan i kada se kontejner zameni. To omogućava ažuriranje kontejnera bez automatskog gubitka poslovnih podataka. Volumen, međutim, ne zamenjuje rezervnu kopiju, koja se izrađuje posebnim postupkom.

2.7. Nginx

Kada korisnik otvori adresu aplikacije, njegov pregledač šalje zahtev serveru. Nginx može da prihvati taj zahtev i prosledi ga programu koji treba da ga obradi. U toj ulozi naziva se obrnuti proksi (reverse proxy).

U predloženom serverskom okruženju Nginx se nalazi ispred Next.js aplikacije. Korisnik pristupa internet domenu, Nginx prima zahtev i prosleđuje ga aplikaciji, a zatim vraća njen odgovor pregledaču.

Nginx se koristi i za podešavanje HTTPS pristupa uz TLS sertifikat. HTTPS omogućava šifrovanu komunikaciju između pregledača i servera, što je posebno značajno pri slanju lozinke i drugih korisničkih podataka.

Povezivanje domena, instalacija sertifikata i konkretna Nginx konfiguracija prikazani su u poglavlju posvećenom omogućavanju web pristupa.

3. RAZVOJ WEB APLIKACIJE

Praktični deo rada obuhvata razvoj web aplikacije WorkHub, namenjene organizaciji projekata i praćenju zadataka u manjem poslovnim timu. Aplikacija objedinjuje podatke o korisnicima, projektnim timovima i obavezama, pri čemu dostupne mogućnosti zavise od korisničke uloge.

Razvoj obuhvata organizaciju projekta, kreiranje baze podataka, prijavljivanje korisnika i realizaciju poslovnih funkcionalnosti. U ovom poglavlju prikazano je kako su te celine povezane u aplikaciju koja se najpre pokreće lokalno, a zatim priprema za postavljanje na server.

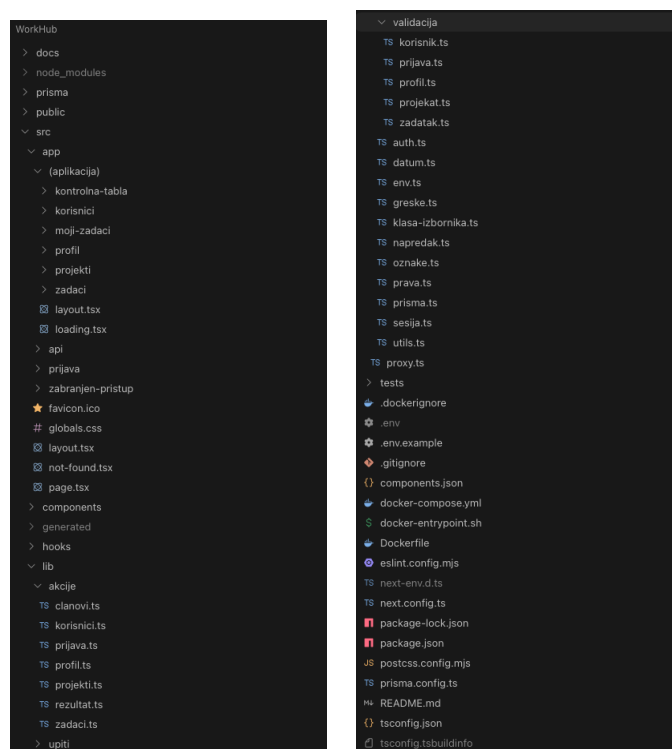
3.1. Priprema razvojnog okruženja i organizacija projekta

Razvojno okruženje predstavlja skup programa i podešavanja potrebnih za izradu i proveru aplikacije na lokalnom računaru. U okviru projekta povezuju se Next.js aplikacija, PostgreSQL baza i alati za upravljanje podacima i korisničkim prijavama.

Izvorni kod organizovan je prema odgovornostima. Stranice i elementi korisničkog interfejsa odvojeni su od serverskih operacija, pravila pristupa i provere podataka. Takva podela olakšava pronalaženje koda i smanjuje ponavljanje istih pravila na više mesta.

Serverske akcije smeštene su u direktorijum `src/lib/akcije/`, dok se šeme za proveru unosa nalaze u `src/lib/validacija/`. Pravila pristupa izdvojena su u modul `prava.ts`, a obrada datuma i izračunavanje napretka u odgovarajuće pomoćne module.

Podešavanja koja zavise od okruženja, kao što su adresa baze i tajna za autentifikaciju, čuvaju se u `.env` datoteci.



Slika 3.1. Organizacija izvornog koda aplikacije WorkHub

3.2. Kreiranje baze podataka i povezivanje sa aplikacijom

WorkHub koristi PostgreSQL bazu podataka. Njena struktura opisana je pomoću Prisma modela, dok se promene strukture evidentiraju kroz migracije. Migracija predstavlja sačuvanu promenu, kao što je kreiranje tabele ili dodavanje polja, koja se kasnije može primeniti i u serverskom okruženju.

Poslovni deo baze čine četiri tabele: `korisnici`, `projekti`, `clanovi_projekta` i `zadaci`. Njihove veze omogućavaju da se za svaki projekat utvrde menadžer, članovi tima i pripadajući zadaci.

Tabela 3.1. Poslovne tabele baze podataka

Tabela	Namena	Najvažniji podaci
<code>korisnici</code>	Evidencija korisnika i njihovih uloga	Ime, prezime, imejl adresa, uloga i aktivnost
<code>projekti</code>	Evidencija projekata	Naziv, opis, menadžer, datum početka, rok i status
<code>clanovi_projekta</code>	Povezivanje korisnika sa projektima	Projekat, korisnik i datum dodavanja
<code>zadaci</code>	Evidencija pojedinačnih obaveza	Projekat, izvršilac, naziv, opis, rok, prioritet i status

Svaki zapis ima jedinstveni identifikator. Veze između tabela ostvaruju se pomoću ključeva (foreign key), odnosno polja koja upućuju na zapis u drugoj tabeli. Na primer, zadatak sadrži identifikator projekta kojem pripada i identifikator korisnika koji ga izvršava.

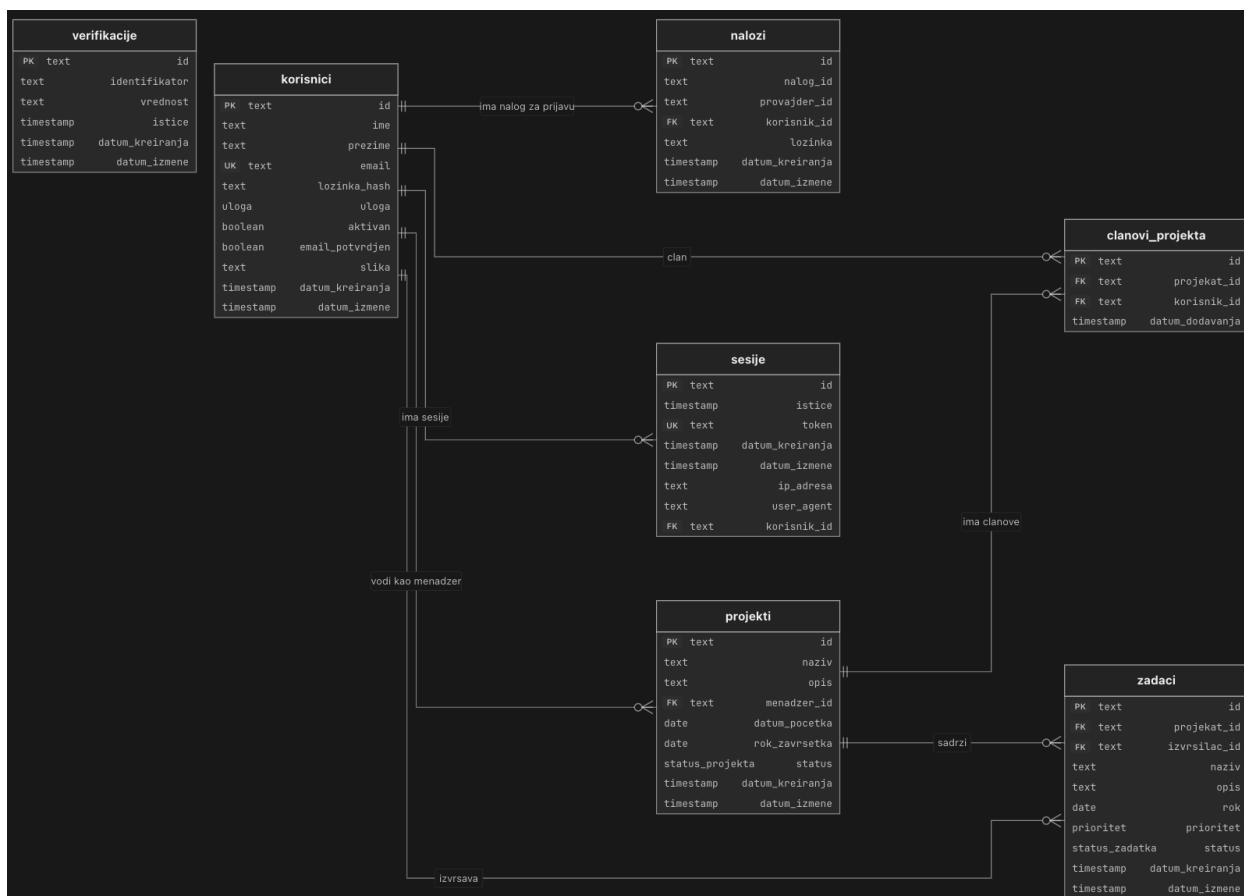
Tabela `clanovi_projekta` omogućava da jedan korisnik bude član više projekata i da jedan projekat ima više članova. Jedinstveno ograničenje nad parom projekta i korisnika sprečava da ista osoba bude dva puta upisana kao član istog projekta. Menadžer se pri kreiranju projekta automatski dodaje i kao član, kako bi i njemu mogao da bude dodeljen zadatak.

Baza sadrži i tehničke tabele `sesije`, `nalozi` i `verifikacije`, koje pripadaju mehanizmu prijavljivanja. One su odvojene od poslovne evidencije. Tabela `verifikacije` u predviđenom načinu rada ostaje neiskorišćena jer aplikacija ne šalje poruke za potvrdu imejl adrese.

Posebna pažnja posvećena je pravilima brisanja. Projekat koji ima zadatke ne može biti obrisani dok ti zadaci postoje. Bisanje projekta bez zadataka uklanja i njegova članstva. Korisnici se u redovnom radu deaktiviraju, čime se čuvaju njihove veze sa ranijim aktivnostima.

Statusi, prioriteti i uloge imaju unapred određene dozvoljene vrednosti. Time se sprečava proizvoljan unos različitih naziva za isto stanje. Napredak projekta i oznaka kašnjenja izračunavaju se iz postojećih podataka, pa za njih nisu potrebne posebne kolone.

Aplikacija pristupa bazi preko Prisma klijenta. Na primer, pri prikazu projekta učitavaju se njegovi podaci i povezani zapisi o članovima i zadacima.



Slika 3.2. Dijagram PostgreSQL baze podataka aplikacije WorkHub

3.3. Prijava korisnika i kontrola pristupa

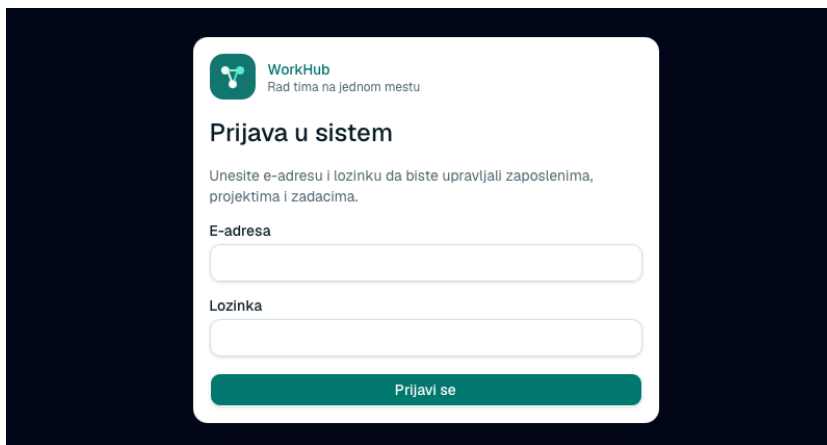
Pristup aplikaciji počinje na stranici za prijavu. Korisnik unosi imejl adresu i lozinku, nakon čega se podaci proveravaju na serveru. Za prijavljivanje i upravljanje sesijama koristi se Better Auth.

Nakon uspešne prijave uspostavlja se sesija, koja omogućava prepoznavanje korisnika pri narednim zahtevima. U pregledač se postavlja HTTP-only kolačić, kojem klijentski JavaScript ne može neposredno da pristupi. Pri odjavi se završava sesija i uklanja odgovarajući kolačić.

Aplikacija proverava i aktivnost korisničkog naloga. Deaktivirani korisnik nema pravo pristupa. Poruka pri neuspešnoj prijavi ostaje opšta, bez posebnog otkrivanja da li je problem u lozinci ili aktivnosti naloga.

Nakon prijave primenjuju se pravila za tri korisničke uloge. Administrator upravlja korisnicima i svim projektima, menadžer sopstvenim projektima, dok korisnik pristupa projektima čiji je član i zadacima koji su mu dodeljeni.

Provera ovlašćenja izvršava se i pri učitavanju podataka i pri njihovoj izmeni. Na primer, korisnik ne može da promeni tuđi zadatak slanjem njegovog identifikatora, jer serverska akcija proverava kome zadatak pripada.



Slika 3.3. Prijava korisnika u aplikaciju

3.4. Evidencija korisnika

Evidencija zaposlenih u aplikaciji ostvarena je kroz korisničke zapise. Za korisnika se čuvaju ime, prezime, imejl adresa, uloga i podatak o aktivnosti. Imejl adresa je jedinstvena i koristi se za prijavljivanje.

Administrator kreira korisničke naloge i upravlja njihovim ulogama i aktivnošću. Javna registracija nije predviđena, pa se pristup aplikaciji odobrava kroz administrativni deo sistema.

Korisnici se deaktiviraju umesto fizičkog brisanja. Na taj način prethodno dodeljeni zadaci i veze sa projektima ostaju sačuvani, dok se onemogućava dalji pristup nalogu. Sistem dodatno sprečava deaktivaciju ili promenu uloge poslednjeg aktivnog administratora.

Svim prijavljenim korisnicima dostupna je izmena dozvoljenih podataka sopstvenog profila i promena lozinke. Ove mogućnosti odvojene su od administrativnog upravljanja drugim korisnicima.

Ime	E-adresa	Uloga	Status	Kreiran	Akcije
Jovana Jovanović	jovana.jovanovic@workhub.test	Korisnik	Aktivan	06.09.2026.	Izmeni Obriši
Marija Marić	marija.maric@workhub.test	Korisnik	Aktivan	06.09.2026.	Izmeni Obriši
Nikola Nikolić	nikola.nikolic@workhub.test	Menadžer	Aktivan	06.09.2026.	Izmeni Obriši
Petar Petrović	petar.petrovic@workhub.test	Korisnik	Aktivan	06.09.2026.	Izmeni Obriši
Stefan Stefanović	stefan.stefanovic@workhub.test	Korisnik	Deaktiviran	06.09.2026.	Izmeni Obriši
Ivana Stojanovic	ivana.stojanovic@workhub.test	Menadžer	Aktivan	06.09.2026.	Izmeni Obriši
Lazar Vujovic	admin@lazarvujovic.com	Administrator	Aktivan	06.09.2026.	Izmeni

Slika 3.4. Pregled korisnika i njihovih uloga

3.5. Upravljanje projektima

Projekat predstavlja celinu koja okuplja članove tima i njihove zadatke. Pri kreiranju unose se naziv, opis, datum početka i rok završetka, a određuje se i odgovorni menadžer.

Administrator može da kreira projekat i dodeli ga menadžeru, dok menadžer kreira projekat za koji je sam odgovoran. U okviru dozvoljenih projekata moguće je uređivati podatke i upravljati članovima.

Projekat može imati status „Planiran“, „U toku“, „Završen“ ili „Otkazan“. Status se određuje ručno, dok se napredak računa na osnovu zadataka. Ova razlika omogućava da se odvojeno prate stanje projekta i realizacija njegovih aktivnosti.

Članstvo određuje koji korisnici pripadaju timu. Izvršioci zadataka biraju se među aktivnim članovima, a uklanjanje člana koji ima zadatke nije dozvoljeno. Time se sprečava da postojeći zadaci ostanu vezani za izvršioca koji više nije član projekta.

Brisanje projekta ograničeno je postojanjem zadataka. Na taj način se izbegava slučajno uklanjanje projekta sa evidentiranim obavezama.

The screenshot displays the 'WorkHub' interface. On the left is a dark sidebar with navigation links: 'Kontrolna tabla', 'Projekti' (highlighted), 'Moji zadaci', 'Korisnici', and 'Profil'. The main content area is titled 'Migracija službenog mejla' and includes a progress bar at 100% (2 tasks completed). It features three panels: 'Osnovni podaci' (Basic data) with project description, manager (Ivana Stojanovic), and period (01.03.2026. — 30.06.2026.); 'Članovi tima' (Team members) listing Ivana Stojanovic (manager), Jovana Jovanović, and Petar Petrović, each with an 'Ukloni' (Remove) button; and 'Zadaci' (Tasks) table.

Naziv	Izvršilac	Status	Prioritet	Rok
Prenos sandučića	Jovana Jovanović	Završen	Visok	15.05.2026.
Obuka korisnika	Petar Petrović	Završen	Srednji	10.06.2026.

Slika 3.5. Pregled projekta i njegovog tima

3.6. Kreiranje, dodeljivanje i praćenje zadataka

Zadatak predstavlja pojedinačnu obavezu u okviru projekta. Sadrži naziv, opis, rok, prioritet, status i jednog izvršioca. Jedan projekat može imati više zadataka, a korisnik može biti zadužen za više obaveza.

Zadatke kreiraju administrator i menadžer odgovarajućeg projekta. Pre upisa proverava se da li izabrani izvršilac postoji, da li je aktivan i da li je član projekta. Ove provere sprovodi serverska logika aplikacije.

Predviđena su tri nivoa prioriteta: nizak, srednji i visok. Statusi zadatka su „Za rad“, „U toku“ i „Završen“. Korisnik može da menja status sopstvenog zadatka, dok administrator i nadležni menadžer mogu da menjaju i ostale podatke.

Kreiranje zadatka odvija se kroz nekoliko povezanih koraka. Korisnik popunjava formu, a serverska akcija proverava prijavu, ovlašćenja i ispravnost unosa. Nakon provere poslovnih pravila, Prisma klijent upisuje zadatak u bazu i on postaje dostupan u odgovarajućem pregledu.

Oznaka „Kasni“ nije poseban status u bazi. Ona se određuje na osnovu roka i stanja zadatka. Za poređenje se koriste kalendarski datumi prema vremenskoj zoni Europe/Belgrade, kako bi se kašnjenje dosledno prikazivalo.

WorkHub
Rad tima na jednom mestu

Kontrolna tabla
Projekti
Moji zadaci
Korisnici
Profil

Novi zadatak

Projekat: Zamena starog CRM sistema

Naziv

Opis

Izvršilac
Izaberite člana

Rok
09 / 07 / 2026

Prioritet
Srednji

Status
Za rad

Sačuvaj zadatak

Slika 3.6. Kreiranje i dodeljivanje zadatka

WorkHub
Rad tima na jednom mestu

Kontrolna tabla
Projekti
Moji zadaci
Korisnici
Profil

Moji zadaci

Zadaci koji kasne prikazuju se prvi, zatim oni sa najbližim rokom.

Pretraga po nazivu Svi statusi Svi prioriteti Svi izvršioci

Filtriraj

Naziv	Projekat	Izvršilac	Status	Prioritet	Rok
Analiza postojećih kartica klijenata	Zamena starog CRM sistema	Marija Marić	Završen	Srednji	28.02.2026.
Prenos sandučića	Migracija službenog mejla	Jovana Jovanović	Završen	Visok	15.05.2026.
Obuka korisnika	Migracija službenog mejla	Petar Petrović	Završen	Srednji	10.06.2026.

Slika 3.7. Praćenje statusa i rokova zadataka

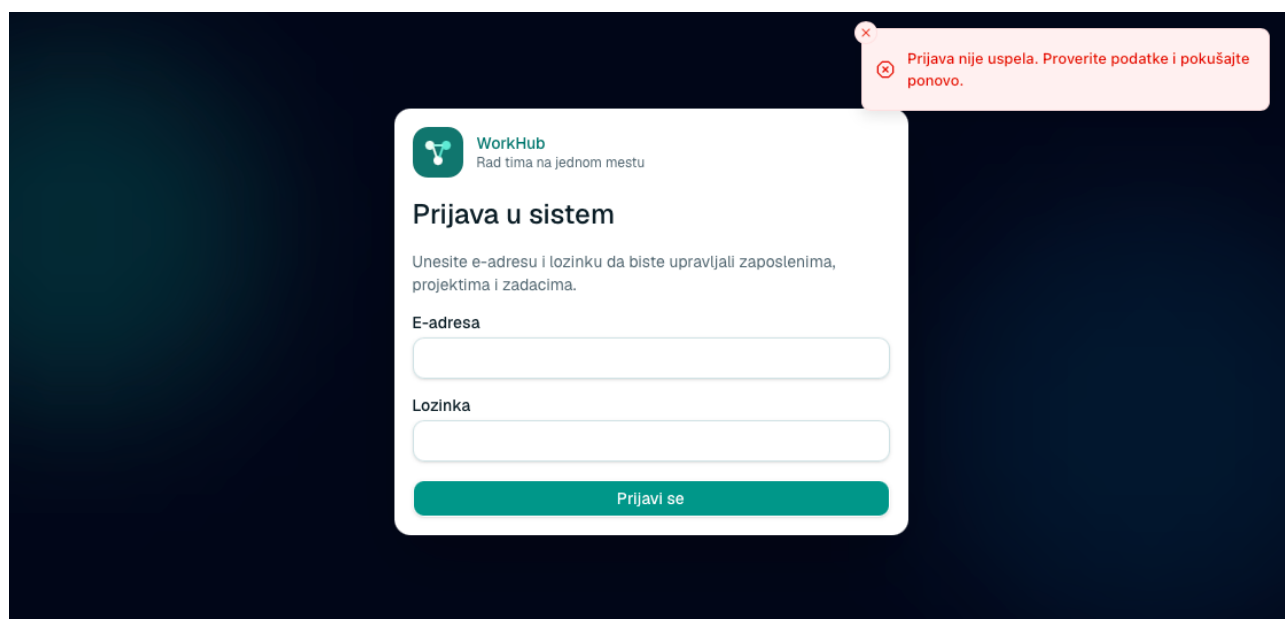
3.7. Korisnički interfejs i validacija podataka

Korisnički interfejs omogućava pregled podataka i izvršavanje dostupnih operacija kroz stranice, tabele, forme i dijaloge. Opcije se prilagođavaju ulozi korisnika, tako da su administrativne mogućnosti dostupne administratoru, a korisnički pregled usmeren na dodeljene obaveze.

Pri unosu podataka primenjuju se provere u pregledaču i na serveru. Pregledač može odmah da upozori na prazno obavezno polje ili prekratak unos. Takve provere olakšavaju korišćenje, dok se konačna odluka o prihvatanju podataka donosi na serveru.

Serverska validacija koristi Zod šeme za proveru oblika i sadržaja unosa. Nakon toga se proveravaju poslovni uslovi, kao što su ispravnost rokova, članstvo izvršioca i ograničenja zaključanog projekta.

Poruka o grešci treba da objasni šta korisnik treba da ispravi. Na primer, pri pokušaju dodeljivanja zadatka neodgovarajućem izvršiocu potrebno je ukazati da izvršilac mora biti aktivan član projekta. Podaci se upisuju tek kada su sve potrebne provere uspešne.



Slika 3.8. Prikaz poruke pri neispravnom unosu

3.8. Lokalno testiranje aplikacije

Pre postavljanja na server potrebno je proveriti osnovne funkcionalnosti u lokalnom okruženju. Cilj je da se utvrdi da li se podaci pravilno čuvaju i prikazuju i da li se korisnička ovlašćenja primenjuju prema definisanim pravilima.

Za proveru se koriste probni nalozi administratora, menadžera i korisnika, kao i projekti sa različitim članovima i zadacima. Početni podaci mogu se pripremiti ručnim pokretanjem postupka za unos probnih podataka, odnosno *seed* postupka.

Testiranje obuhvata uspešne operacije i pokušaje radnji koje sistem treba da odbije. Posebno je značajno proveriti pristup tuđim projektima i zadacima, kao i ograničenja koja štite povezane podatke.

Tabela 3.2. Plan lokalne provere aplikacije

Provera	Očekivani rezultat
Prijava aktivnim nalogom i ispravnim podacima	Korisniku je omogućen pristup.
Prijava deaktiviranim nalogom	Pristup je odbijen.
Kreiranje projekta od strane menadžera	Menadžer postaje odgovoran za projekat i njegov član.
Izmena projekta drugog menadžera	Operacija je odbijena.
Dodeljivanje zadatka aktivnom članu	Zadatak je uspešno sačuvan.
Dodeljivanje zadatka korisniku koji nije član	Operacija je odbijena.
Promena statusa sopstvenog zadatka	Novi status je sačuvan.
Pokušaj korisnika da izmeni tuđi zadatak	Operacija je odbijena.
Uklanjanje člana koji ima zadatke	Operacija je odbijena.
Brisanje projekta sa zadacima	Operacija je odbijena.
Deaktivacija poslednjeg aktivnog administratora	Operacija je odbijena.
Ponovno učitavanje stranice nakon izmene	Prikazani su prethodno sačuvani podaci.

Rezultati se evidentiraju nakon izvođenja provera, uz beleženje uočenih grešaka i izvršenih ispravki. Provere domena, HTTPS pristupa i ponašanja nakon ponovnog pokretanja servera pripadaju testiranju produkcijskog okruženja i obrađuju se u kasnijem poglavlju.

4. IZBOR I PRIPREMA SERVERSKOG OKRUŽENJA

Nakon razvoja aplikacije potrebno je obezbediti okruženje u kojem će ona biti dostupna putem interneta. Za aplikaciju WorkHub zakupljen je virtualni server kod provajdera Hetzner. Na njemu se izvršavaju aplikacija, baza podataka i programi potrebni za prihvatanje korisničkih zahteva.

Priprema obuhvata izbor resursa, operativnog sistema i načina pristupa, kao i podešavanje zaštite i instalaciju potrebnog softvera.

4.1. Zahtevi aplikacije i izbor načina hostovanja

Hostovanje predstavlja obezbeđivanje računarskih resursa na kojima se aplikacija izvršava. Za WorkHub je potrebno okruženje koje omogućava pokretanje Next.js aplikacije i PostgreSQL baze, korišćenje Docker-a i podešavanje Nginx-a.

Izabran je VPS (Virtual Private Server), odnosno virtualni privatni server. On omogućava samostalnu instalaciju programa i upravljanje konfiguracijom, što odgovara cilju rada da se prikaže ceo postupak postavljanja aplikacije.

Pošto je sistem namenjen manjem poslovnom timu, aplikacija i baza smeštene su na jednom serveru. Ovakvo rešenje pojednostavljuje administriranje, ali znači i da prekid rada servera utiče na dostupnost celog sistema.

4.2. Izbor i zakup VPS servera

Server je zakupljen preko Hetzner kontrolnog panela. Postupak kreiranja obuhvata izbor paketa, lokacije, operativnog sistema, mrežnih opcija i načina pristupa.

Za realizaciju je korišćen paket CPX22. Osnovne karakteristike aktivnog servera prikazane su u tabeli.

Tabela 4.1. Karakteristike zakupljenog servera

Karakteristika	Vrednost
Provajder	Hetzner
Paket	CPX22
Naziv servera	V-App
Virtuelni procesori	2 vCPU
Radna memorija	4 GB
Prostor na disku	80 GB
Lokacija	Nirnberg, Nemačka

Virtuelni procesori obavljaju obradu zahteva, radna memorija služi izvršavanju programa, a disk čuva aplikaciju, bazu i ostale datoteke. Navedena konfiguracija odabrana je kao početno okruženje za demonstraciju sistema. Njena dovoljnost pri većem broju korisnika procenjuje se kroz praćenje opterećenja i testiranje.

Kontrolni panel omogućava pregled stanja servera i upravljanje mrežom, napajanjem i rezervnim kopijama. Na prikazu aktivnog servera uključena je opcija rezervnih kopija. Njihovu upotrebu i proveru oporavka potrebno je razmotriti u okviru održavanja sistema.

The screenshot displays the Hetzner control panel for a server named CPX22 V-App. The interface is dark-themed with red accents. At the top, there's a navigation bar with tabs: Overview (selected), Graphs, Backups, Snapshots, Load Balancers, Networking, Firewalls, Volumes, Power, and Res>U. Below the navigation bar, a summary row shows server specifications: 2 VCPU, 4 GB RAM, 80 GB DISK LOCAL, € 2.10 USAGE, 0/20 TB TRAFFIC OUT, and € 7.99 /mo PRICE. A button labeled 'Add labels' is positioned below this row. The main content area is divided into three sections. The 'ACTIVITIES' section on the left features a sad face icon and the text 'No activities within the last 90 days.' with a 'View all >' link. The 'OPTIONS' section on the right includes settings for BACKUPS (Disable), PLACEMENT GROUP (Select group), and PUBLIC NETWORK (Disable). The 'LOCATION' section at the bottom right lists the NETWORK ZONE as eu-central, CITY as Nuremberg, and COUNTRY as Germany, accompanied by a map of Germany with a red location pin.

Slika 4.1. Pregled zakupljenog servera u Hetzner kontrolnom panelu

4.3. Izbor operativnog sistema

Operativni sistem upravlja resursima servera i omogućava izvršavanje potrebnih programa. U postupku kreiranja Hetzner nudi više Linux distribucija, među kojima su Ubuntu, Debian i druge.

Izabrana je verzija Ubuntu 26.04. Verziju instaliranu na aktivnom serveru potrebno je potvrditi na samom sistemu, jer izbor prikazan u obrascu ne potvrđuje da je taj server kasnije kreiran sa istim podešavanjima.

Podaci o instaliranom sistemu mogu se pregledati komandom:

```
cat /etc/os-release
```

Tačna verzija važna je pri izboru uputstava za instalaciju i proveru kompatibilnosti softvera. U nastavku se serversko okruženje koristi za rad sa Docker-om, Git-om i Nginx-om.

4.4. Povezivanje sa serverom putem SSH protokola

Administriranje udaljenog servera obavlja se putem SSH protokola. On omogućava zaštićeno povezivanje sa lokalnog računara i izvršavanje komandi u terminalu servera.

Opšti oblik komande za povezivanje je:

```
ssh korisnik@IP_ADRESA_SERVERA
```

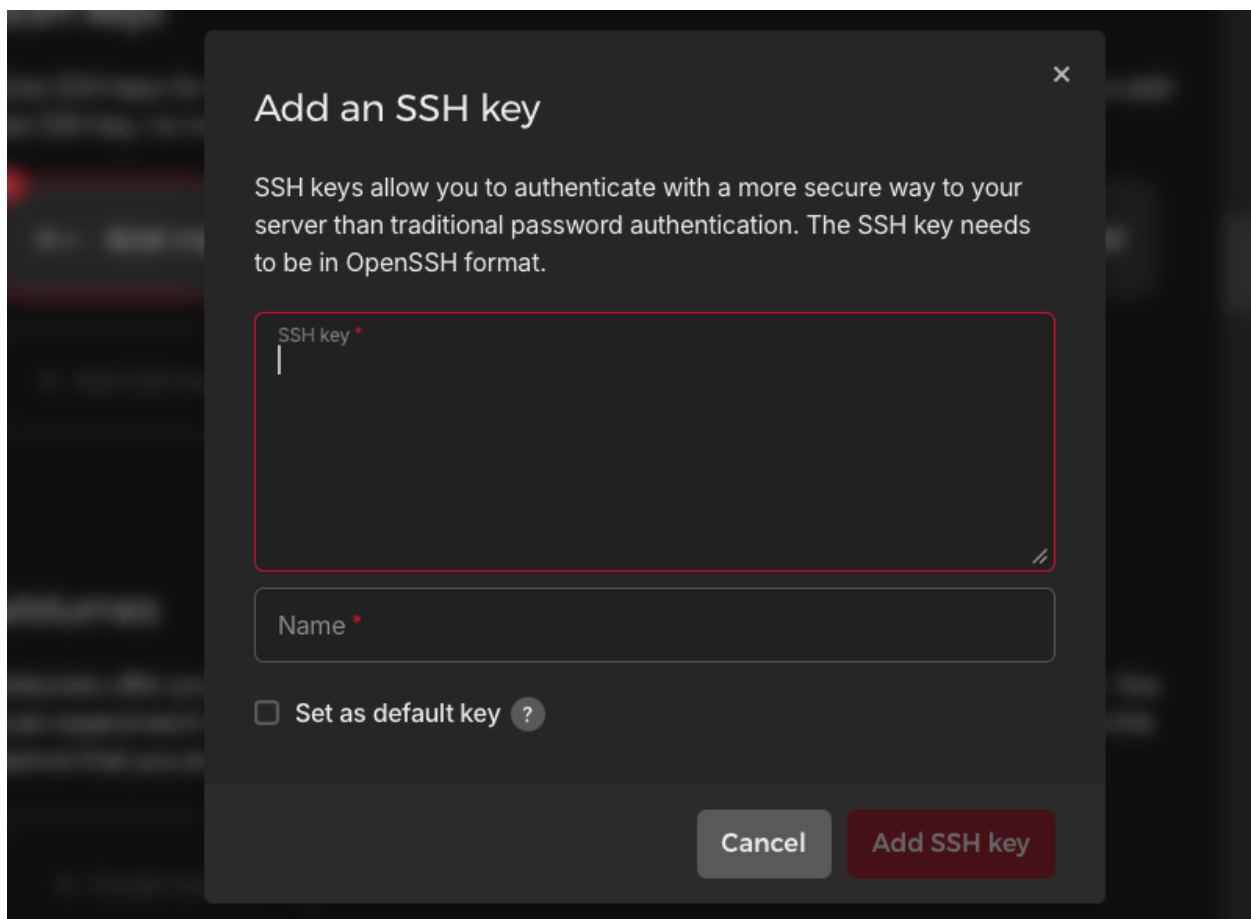
Korisničko ime određuje nalog na serveru, a IP adresa računar kojem se pristupa. Obično je `root` nalog koji ima puna administratorska ovlašćenja.

4.5. Podešavanje korisničkih naloga i SSH ključeva

SSH pristup može se zasnivati na paru ključeva. Javni ključ dodaje se serveru, dok privatni ostaje na računaru administratora i koristi se pri povezivanju.

Hetzner omogućava unos javnog ključa tokom kreiranja servera. U prikazanom postupku odabran je jedan SSH ključ. Time se unapred priprema pristup serveru za vlasnika odgovarajućeg privatnog ključa.

Za svakodnevno administriranje poželjno je koristiti poseban nalog i dodeliti mu mogućnost izvršavanja administratorskih komandi pomoću `sudo (SuperUser Do)`. Tako se puna ovlašćenja koriste samo kada su potrebna.



Slika 4.2. Dodavanje javnog SSH ključa pri kreiranju servera

5. KONTEJNERIZACIJA I POSTAVLJANJE APLIKACIJE

Nakon pripreme serverskog okruženja pristupilo se postavljanju WorkHub aplikacije i baze podataka. Za njihovo pokretanje korišćeni su Docker i Docker Compose, čime su aplikacija i PostgreSQL baza izdvojene u zasebne kontejnere i povezane u okviru zajedničkog okruženja.

5.1. Preuzimanje izvornog koda (source code) na server

Izvorni kod aplikacije čuva se u udaljenom Git repozitorijumu. Nakon povezivanja sa serverom prvo je napravljen direktorijum namenjen WorkHub aplikaciji, a zatim je u njega kloniran repozitorijum.

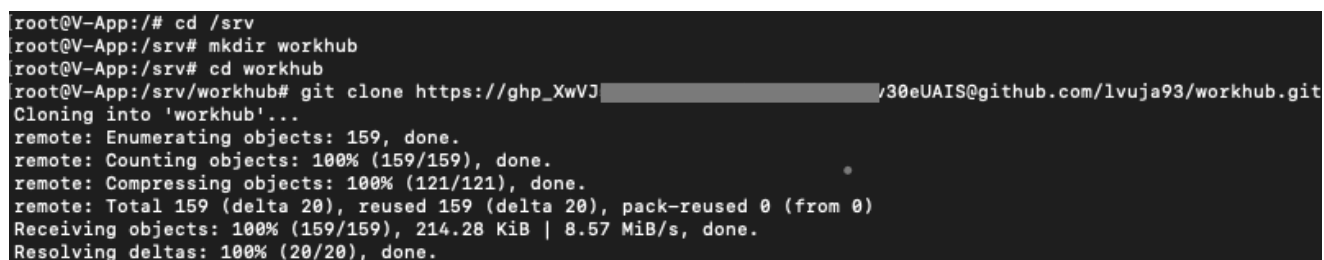
Postupak se može prikazati sledećim komandama:

```
mkdir workhub
cd workhub
git clone <adresa-repozitorijuma> .
```

Na ovaj način kompletan projekat preuzet je direktno na server. Kod kasnijih izmena nije potrebno ponovo klonirati repozitorijum, već se nova verzija preuzima komandom:

```
git pull
```

Datoteka `.env`, lokalno instalirani paketi i privremene datoteke koje aplikacija sama generiše nisu deo Git repozitorijuma, jer su isključeni pomoću `.gitignore` datoteke.



```
root@V-App:/# cd /srv
root@V-App:/srv# mkdir workhub
root@V-App:/srv# cd workhub
root@V-App:/srv/workhub# git clone https://ghp_XwVJ[REDACTED]/30eUAIS@github.com/lvuja93/workhub.git
Cloning into 'workhub'...
remote: Enumerating objects: 159, done.
remote: Counting objects: 100% (159/159), done.
remote: Compressing objects: 100% (121/121), done.
remote: Total 159 (delta 20), reused 159 (delta 20), pack-reused 0 (from 0)
Receiving objects: 100% (159/159), 214.28 KiB | 8.57 MiB/s, done.
Resolving deltas: 100% (20/20), done.
```

Slika 5.1. Preuzimanje izvornog koda WorkHub aplikacije na server

5.2. Docker konfiguracija aplikacije i baze podataka

Za pokretanje WorkHub sistema napravljen je `Dockerfile` za Next.js aplikaciju i Docker Compose konfiguracija koja povezuje aplikaciju sa PostgreSQL bazom.

Dockerfile koristi višefaznu izgradnju zasnovanu na Node.js 22 Alpine okruženju. U odvojenim fazama instaliraju se zavisnosti, izgrađuje Next.js aplikacija i pripremaju Prisma alati. Završni kontejner pokreće aplikaciju pod korisnikom `nextjs`, a ne kao administratorski korisnik.

Docker Compose konfiguracija sadrži dva servisa:

- `app` — Next.js aplikacija;
- `db` — PostgreSQL 16 baza podataka.

Oba servisa povezana su na Docker mrežu `workhub`. Aplikacija zato bazi pristupa preko interne adrese `db:5432`, bez korišćenja javne IP adrese servera.

PostgreSQL baza je na samom serveru povezana sa 127.0.0.1:5435, dok je Next.js aplikacija povezana sa 127.0.0.1:3015. Vezivanje za localhost znači da ovi portovi nisu namenjeni direktnom pristupu preko interneta.

Pre pokretanja aplikacije proverava se spremnost PostgreSQL baze pomoću `pg_isready`. Aplikacioni servis je podešen da sačeka dok baza ne bude spremna za prihvatanje konekcija.

```
1 services:
2   db:
3     container_name: workhub-db
4     image: postgres:22-alpine
5     restart: unless-stopped
6     env_file:
7       - .env
8     environment:
9       POSTGRES_USER: ${POSTGRES_USER}
10      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
11      POSTGRES_DB: ${POSTGRES_DB}
12    ports:
13      - '127.0.0.1:5435:5432'
14    volumes:
15      - workhub_postgres:/var/lib/postgresql/data
16    networks:
17      - workhub
18    healthcheck:
19      test: ['CMD-SHELL', 'pg_isready -U ${POSTGRES_USER} -d ${POSTGRES_DB}']
20      interval: 5s
21      timeout: 5s
22      retries: 10
23
24  app:
25    container_name: workhub-app
26    build:
27      context: .
28      args:
29        DATABASE_URL: postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@db:5432/${POSTGRES_DB}
30        BETTER_AUTH_SECRET: ${BETTER_AUTH_SECRET}
31        BETTER_AUTH_URL: ${BETTER_AUTH_URL}
32    restart: unless-stopped
33    env_file:
34      - .env
35    ports:
36      - '127.0.0.1:3015:3000'
37    environment:
38      NODE_ENV: production
39      PORT: 3000
40      HOSTNAME: 0.0.0.0
41      DATABASE_URL: postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@db:5432/${POSTGRES_DB}
42      BETTER_AUTH_SECRET: ${BETTER_AUTH_SECRET}
43      BETTER_AUTH_URL: ${BETTER_AUTH_URL}
44    networks:
45      - workhub
46    depends_on:
47      db:
48        condition: service_healthy
49
50  volumes:
51    workhub_postgres:
52      name: workhub_postgres
53
54  networks:
55    workhub:
56      name: workhub
```

Slika 5.2. Docker Compose konfiguracija WorkHub sistema

5.3. Promenljive okruženja i trajno čuvanje podataka

Podaci koji ne treba da budu deo izvornog koda čuvaju se u `.env` datoteci na serveru. Tu se nalaze podaci za povezivanje sa bazom, adresa aplikacije i tajna vrednost potrebna sistemu za autentifikaciju.

Produksijska javna adresa aplikacije definisana je kao:

`https://workhub.lazarvujovic.com`

Sama `.env` datoteka nije deo Git repozitorijuma, čime se sprečava objavljivanje lozinki i drugih poverljivih podataka.

Podaci PostgreSQL baze čuvaju se u Docker volumenu `workhub_postgres`. Zbog toga ponovno kreiranje kontejnera ne podrazumeva brisanje podataka iz baze.

Pre pokretanja aplikacije izvršava se skripta `docker-entrypoint.sh`, koja pokreće postupak za primenu migracija baze. Početni podaci se ne unose automatski. Seed je ručno pokrenut komandom:

```
docker compose run --rm app seed
```

5.4. Pokretanje aplikacije

Kada su izvorni kod, Docker konfiguracija i promenljive okruženja pripremljeni, WorkHub je izgrađen i pokrenut pomoću komande:

```
docker compose up --build -d
```

Opcija `--build` omogućava izgradnju nove verzije aplikacionog imidža, dok `-d` pokreće kontejnere u pozadini.

Za oba servisa podešeno je pravilo `restart: unless-stopped`, tako da se kontejneri mogu ponovo pokrenuti nakon prekida rada, osim kada su prethodno namerno zaustavljeni.

Prilikom kasnijeg ažuriranja aplikacije nova verzija izvornog koda preuzima se komandom `git pull`, nakon čega se Docker aplikacija ponovo izgradi i pokrene.

U ovoj fazi aplikacija je dostupna lokalno na serveru preko adrese `127.0.0.1:3015`. Da bi korisnici mogli da joj pristupe putem javnog domena i bezbedne HTTPS veze, potrebno je podesiti Nginx, DNS i TLS sertifikat, što je obrađeno u narednom poglavlju.

6. POVEZIVANJE DOMENA I OBEZBEĐIVANJE WEB PRISTUPA

Nakon što je WorkHub aplikacija pokrenuta na serveru, bilo je potrebno omogućiti pristup putem internet domena i zaštititi komunikaciju između korisnika i servera. Za upravljanje domenom i DNS zapisima korišćen je Cloudflare, dok je Nginx podešen kao obrnuti proksi ispred aplikacije. HTTPS pristup omogućen je TLS sertifikatom izdatim pomoću Certbot alata.

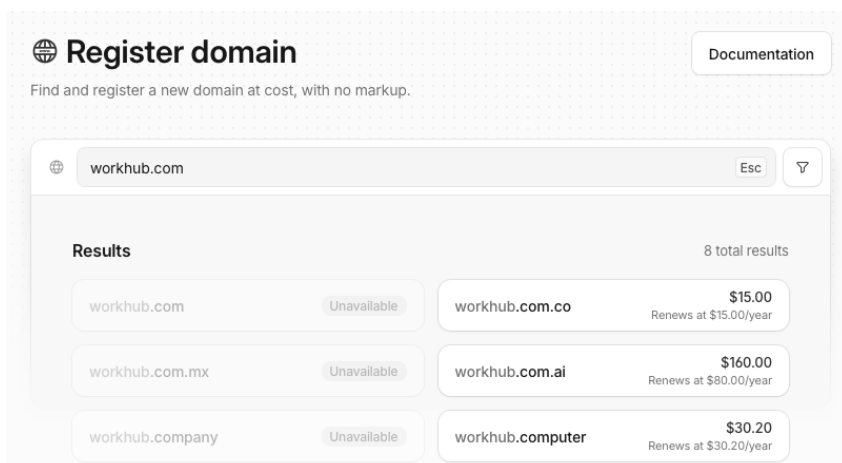
6.1. Registracija domena i DNS konfiguracija

Osnovni domen `lazarvujovic.com` registrovan je preko Cloudflare servisa. Za WorkHub aplikaciju korišćen je poddomen:

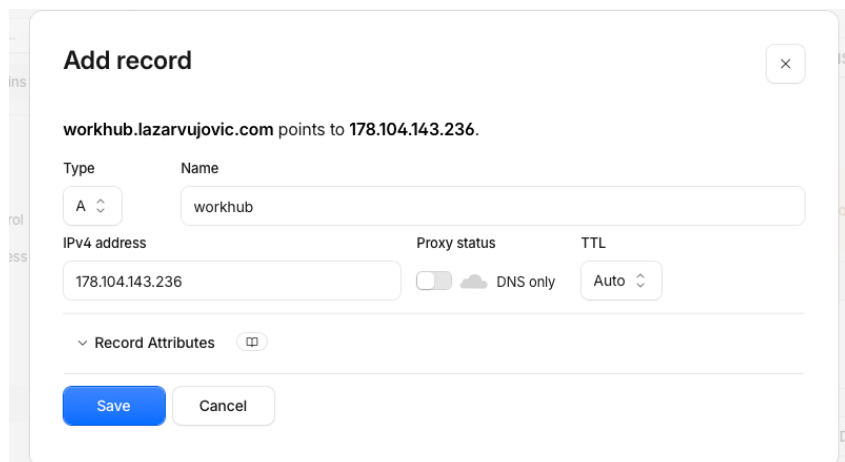
`workhub.lazarvujovic.com`

Da bi se zahtevi za ovaj poddomen usmerili ka VPS serveru, u Cloudflare DNS podešavanjima kreiran je A zapis. A zapis povezuje naziv domena sa javnom IPv4 adresom servera.

Na taj način, kada korisnik pristupi adresi `workhub.lazarvujovic.com`, DNS sistem određuje IP adresu VPS servera na kojem je aplikacija postavljena.



Slika 6.1. Pretraga domena u Cloudflare servisu



Slika 6.2. DNS zapis za WorkHub poddomen

6.2. Podešavanje Nginx obrnutog proksija (reverse proxy)

WorkHub aplikacija unutar servera dostupna je na lokalnoj adresi 127.0.0.1:3015. Krajnji korisnici, međutim, ne pristupaju direktno ovom portu. Ispred aplikacije postavljen je Nginx, koji prima zahteve sa interneta i prosleđuje ih Next.js aplikaciji.

Za WorkHub je napravljena posebna Nginx konfiguracija u kojoj je kao naziv servera definisan domen:

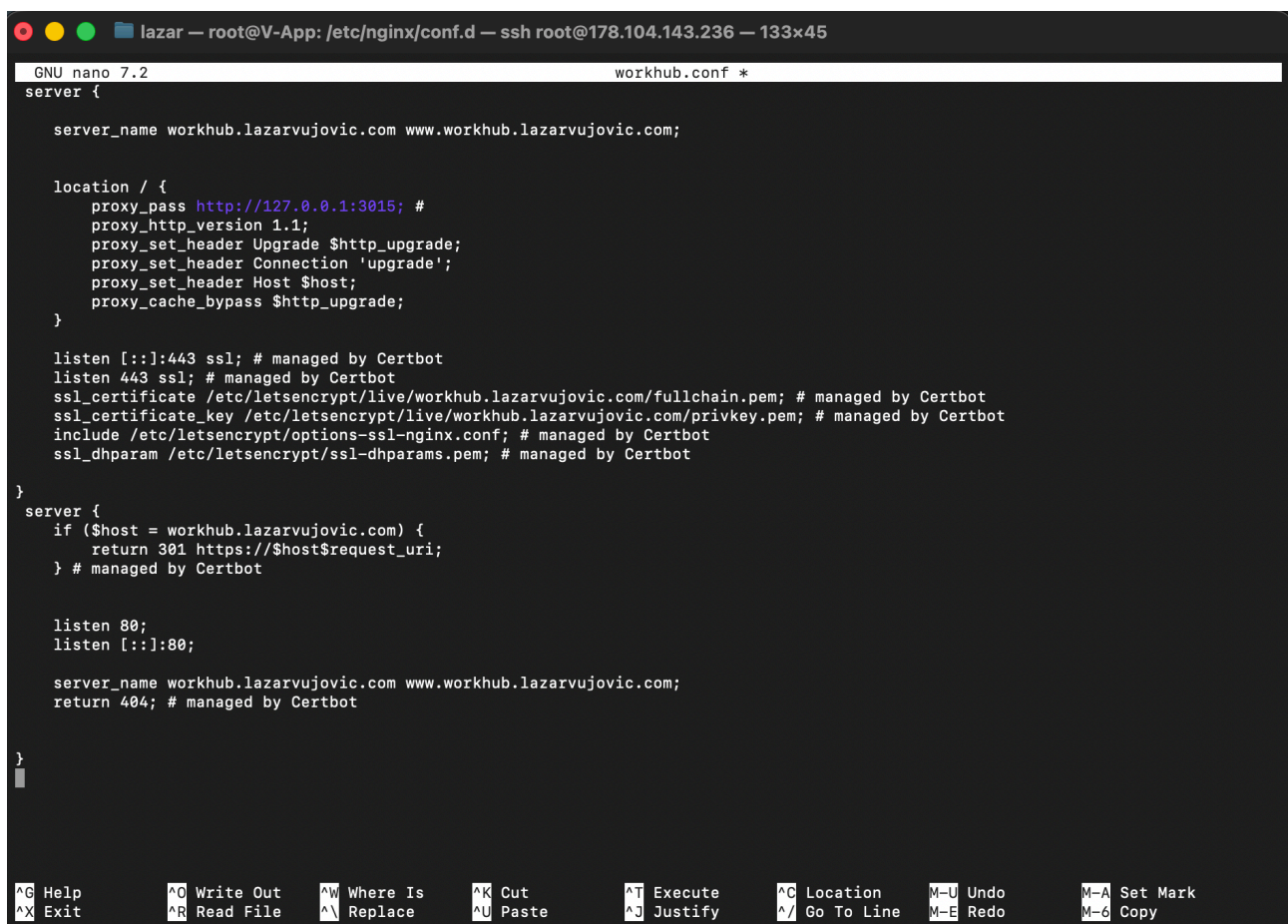
```
server_name workhub.lazarvujovic.com;
```

Zahtevi koji stignu na ovaj domen prosleđuju se aplikaciji pomoću direktive:

```
proxy_pass http://127.0.0.1:3015;
```

Nginx tako ima ulogu posrednika između javnog internet pristupa i aplikacije koja radi unutar servera. Ovakav pristup omogućava da sama aplikacija ostane dostupna samo lokalno, dok Nginx upravlja javnim web saobraćajem.

U konfiguraciji se prosleđuje i originalni naziv domena, kao i potrebna HTTP zaglavlja za komunikaciju između klijenta, Nginx servera i Next.js aplikacije.



```
GNU nano 7.2 workhub.conf *
server {

    server_name workhub.lazarvujovic.com www.workhub.lazarvujovic.com;

    location / {
        proxy_pass http://127.0.0.1:3015; #
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    listen [::]:443 ssl; # managed by Certbot
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/workhub.lazarvujovic.com/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/workhub.lazarvujovic.com/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = workhub.lazarvujovic.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    listen [::]:80;

    server_name workhub.lazarvujovic.com www.workhub.lazarvujovic.com;
    return 404; # managed by Certbot
}

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo     M-A Set Mark
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line M-E Redo     M-6 Copy
```

Slika 6.3. Nginx konfiguracija za WorkHub aplikaciju

6.3. Izdavanje TLS sertifikata

Samo povezivanje domena sa serverom ne obezbeđuje šifrovanu komunikaciju. Za zaštitu podataka koji se razmenjuju između korisničkog pregledača i servera koristi se HTTPS protokol, koji se zasniva na TLS sertifikatu.

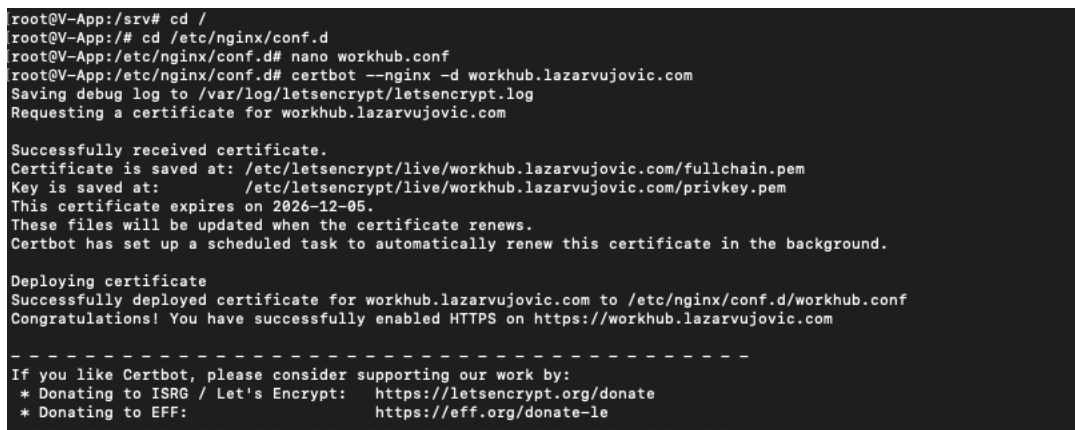
Za WorkHub je sertifikat izdat pomoću Certbot alata i njegove integracije sa Nginx serverom. Korišćena je komanda:

```
certbot --nginx -d workhub.lazarvujovic.com
```

Certbot je nakon provere domena izdao sertifikat i prilagodio Nginx konfiguraciju za HTTPS pristup. U konfiguraciji su nakon toga definisani port 443 i putanje do sertifikata i privatnog ključa:

```
ssl_certificate /etc/letsencrypt/live/workhub.lazarvujovic.com/fullchain.pem;  
ssl_certificate_key /etc/letsencrypt/live/workhub.lazarvujovic.com/privkey.pem;
```

Na slici izvršenja Certbot komande potvrđeno je uspešno izdavanje sertifikata, kao i podešavanje automatskog obnavljanja sertifikata.



```
root@V-App:~# cd /  
root@V-App:~# cd /etc/nginx/conf.d  
root@V-App:~# cd /etc/nginx/conf.d# nano workhub.conf  
root@V-App:~# cd /etc/nginx/conf.d# certbot --nginx -d workhub.lazarvujovic.com  
Saving debug log to /var/log/letsencrypt/letsencrypt.log  
Requesting a certificate for workhub.lazarvujovic.com  
  
Successfully received certificate.  
Certificate is saved at: /etc/letsencrypt/live/workhub.lazarvujovic.com/fullchain.pem  
Key is saved at: /etc/letsencrypt/live/workhub.lazarvujovic.com/privkey.pem  
This certificate expires on 2026-12-05.  
These files will be updated when the certificate renews.  
Certbot has set up a scheduled task to automatically renew this certificate in the background.  
  
Deploying certificate  
Successfully deployed certificate for workhub.lazarvujovic.com to /etc/nginx/conf.d/workhub.conf  
Congratulations! You have successfully enabled HTTPS on https://workhub.lazarvujovic.com  
  
-----  
If you like Certbot, please consider supporting our work by:  
* Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate  
* Donating to EFF: https://eff.org/donate-le  
-----
```

Slika 6.4. Izdavanje TLS sertifikata za WorkHub domen

6.4. HTTPS pristup aplikaciji

Nakon podešavanja sertifikata Nginx prihvata HTTPS zahteve na standardnom portu 443 i prosleđuje ih WorkHub aplikaciji na lokalni port 3015.

Certbot je u Nginx konfiguraciju dodao i pravilo za preusmeravanje zahteva sa HTTP verzije domena na HTTPS:

```
return 301 https://$host$request_uri;
```

Time se pristup adresi preko običnog HTTP protokola preusmerava na šifrovanu HTTPS vezu. Konačna javna adresa aplikacije je:

```
https://workhub.lazarvujovic.com
```

Ista adresa koristi se i kao vrednost promenljive `BETTER_AUTH_URL`, tako da sistem za autentifikaciju koristi produkcijsku HTTPS adresu aplikacije.

Ovim korakom završeno je povezivanje aplikacije sa javnim domenom i obezbeđen je šifrovan pristup preko interneta. Nakon toga slede provere funkcionalnosti, pristupa, rada kontejnera i postupaka održavanja sistema.

7. TESTIRANJE I ODRŽAVANJE SISTEMA

Nakon postavljanja aplikacije u produkciono okruženje potrebno je proveriti da li su osnovne funkcije sistema dostupne, da li serverske komponente rade i da li je moguće pratiti eventualne greške. Pored samog testiranja, važno je definisati i jednostavan postupak za ažuriranje i održavanje aplikacije.

7.1. Funkcionalno testiranje aplikacije

Funkcionalno testiranje WorkHub aplikacije obuhvata proveru najvažnijih poslovnih funkcija i kontrole pristupa. Posebnu pažnju treba posvetiti prijavi korisnika, radu sa projektima i zadacima, kao i razlikama između administratorske, menadžerske i korisničke uloge.

Plan osnovnih testova može se prikazati na sledeći način:

Tabela 7.1. Plan funkcionalnog testiranja WorkHub aplikacije

Test	Očekivani rezultat
Prijava ispravnim korisničkim podacima	Korisnik uspešno pristupa aplikaciji
Unos pogrešne lozinke	Prijava se odbija
Prijava deaktiviranim nalogom	Pristup se odbija
Kreiranje projekta	Projekat se evidentira u sistemu
Dodavanje člana projekta	Aktivni korisnik postaje član projekta
Kreiranje i dodela zadatka	Zadatak se dodeljuje članu projekta
Promena statusa sopstvenog zadatka	Novi status se čuva
Pokušaj neovlašćene izmene	Operacija se odbija
Brisanje projekta koji ima zadatke	Brisanje se odbija
Pristup aplikaciji putem HTTPS-a	Aplikacija se otvara preko zaštićene veze

Ova tabela predstavlja plan provere funkcionalnosti. Rezultate pojedinačnih testova treba unositi samo za provere koje su stvarno izvršene.

7.2. Provera rada Docker servisa i logova

Stanje WorkHub servisa provereno je pomoću komande:

```
docker compose ps
```

Rezultat je pokazao da su oba servisa aktivna. Aplikacioni kontejner `workhub-app` bio je u stanju `Up`, dok je PostgreSQL kontejner `workhub-db` bio označen kao `Up` i `healthy`.

Prikaz portova takođe potvrđuje prethodno definisanu konfiguraciju: aplikacija je povezana sa 127.0.0.1:3015, dok je PostgreSQL baza povezana sa 127.0.0.1:5435.

```
root@V-App:~# cd /srv/workhub/workhub
root@V-App:/srv/workhub/workhub# docker compose ps
```

NAME	IMAGE	COMMAND	SERVICE	CREATED	STATUS	PORTS
workhub-app	workhub-app	"/docker-entrypoint..."	app	19 hours ago	Up 19 hours	127.0.0.1:3015->3000/tcp
workhub-db	postgres:16-alpine	"docker-entrypoint.s..."	db	44 hours ago	Up 44 hours (healthy)	127.0.0.1:5435->5432/tcp

Slika 7.1. Stanje WorkHub Docker servisa

Za dodatnu proveru korišćeni su logovi aplikacije i baze. U aplikacionom logu zabeleženo je da su migracije primenjene i da je Next.js server pokrenut na portu 3000.

U istom logu postoje i zabeleženi neuspešni pokušaji prijave, označeni porukom `Invalid password`, kao i ranije greške povezane sa Server Action zahtevima. Pošto samo na osnovu loga nije moguće utvrditi uzrok niti trenutni uticaj ovih poruka, one se koriste kao primer značaja praćenja serverskih logova.

Pregled logova pokazuje značaj ovog mehanizma tokom postavljanja i kasnijeg održavanja sistema, jer omogućava brzo uočavanje problema bez direktnog pristupa samoj aplikaciji.

7.3. Ažuriranje aplikacije

Kada se u razvojnoj verziji aplikacije izvrše izmene, nova verzija se prvo čuva u udaljenom Git repozitorijumu. Na produkcionom serveru promene se zatim preuzimaju komandom:

```
git pull
```

Nakon preuzimanja nove verzije aplikacioni Docker imidž se ponovo izgrađuje i servisi pokreću komandom:

```
docker compose up --build -d
```

Ovaj postupak omogućava da se ažuriranje aplikacije obavi bez ručnog kopiranja pojedinačnih datoteka. Nakon pokretanja moguće je ponovo koristiti `docker compose ps` i logove radi provere stanja servisa.

7.4. Rezervne kopije (backup) i mogućnosti unapređenja

Na Hetzner serveru uključena je opcija rezervnih kopija servera. Ona predstavlja dodatni nivo zaštite u slučaju problema sa celim serverskim okruženjem.

Međutim, rezervna kopija celog servera nije isto što i posebna rezervna kopija PostgreSQL baze. Za precizniju zaštitu podataka moguće je dodatno koristiti PostgreSQL alat `pg_dump` i čuvati kopije baze na odvojenoj lokaciji.

Kao moguća dalja unapređenja mogu se izdvojiti automatizovane rezervne kopije baze, detaljnije praćenje resursa servera, centralizovano prikupljanje logova i automatizovano postavljanje novih verzija aplikacije.

Na ovaj način WorkHub predstavlja kompletan primer puta web aplikacije od izvornog koda do produkcionog sistema, uz osnovne mehanizme za proveru rada i održavanje.

ZAKLJUČAK

Izrada web aplikacije ne završava se trenutkom kada su njene funkcionalnosti razvijene i kada ona uspešno radi u lokalnom razvojnom okruženju. Da bi aplikacija mogla da se koristi u realnim uslovima, potrebno je razumeti i način njenog postavljanja, povezivanja sa bazom podataka, zaštite pristupa i kasnijeg održavanja. Upravo povezivanje ovih oblasti predstavlja jednu od najvažnijih vrednosti praktične realizacije prikazane u ovom radu.

Rad na aplikaciji WorkHub omogućio je sagledavanje web sistema kao jedne povezane celine. Razvoj poslovne logike i korisničkih funkcionalnosti predstavlja samo jedan njen deo. Jednako su važni organizacija podataka, upravljanje korisničkim pravima, čuvanje izvornog koda, priprema serverskog okruženja, povezivanje različitih servisa i obezbeđivanje sigurnog pristupa putem interneta. Greška ili nepravilno podešavanje samo jedne od ovih komponenti može uticati na rad celog sistema.

Poseban značaj ovog rada ogleda se u praktičnom upoznavanju sa putem aplikacije od razvojnog računara do produkcionog servera. Kroz taj proces postaje jasnije zbog čega su potrebni odvojeno razvojno i produkciono okruženje, na koji način aplikacija komunicira sa bazom, kako se izvorni kod prenosi i ažurira i zbog čega javna dostupnost aplikacije zahteva dodatne slojeve zaštite i konfiguracije.

Realizacija projekta pokazala je i značaj automatizacije i ponovljivosti postupaka. Korišćenjem unapred definisanih konfiguracija moguće je smanjiti broj ručnih koraka prilikom postavljanja i ažuriranja sistema. Istovremeno, praćenje logova, provera stanja servisa i planiranje rezervnih kopija pokazuju da održavanje nije odvojena aktivnost koja počinje tek kada se pojavi problem, već sastavni deo životnog ciklusa aplikacije.

Stechena znanja imaju širu primenu od samog WorkHub projekta. Razumevanje odnosa između aplikacije, baze podataka, operativnog sistema, mreže, domena i bezbednog web pristupa predstavlja važnu osnovu za razvoj i administraciju savremenih informacionih sistema. Ovakav način rada omogućava da se problem ne posmatra samo iz ugla programiranja, već i iz ugla pouzdanosti, dostupnosti i praktične upotrebe sistema.

WorkHub u trenutnom obliku predstavlja manju poslovnu aplikaciju, ali je kroz njegovu realizaciju prikazan proces koji se u sličnom obliku primenjuje i kod složenijih sistema. Daljim razvojem moguće je proširiti poslovne funkcionalnosti, automatizovati rezervne kopije i objavljivanje novih verzija, unaprediti praćenje rada servera i dodatno povećati otpornost sistema. Na taj način rad predstavlja osnovu ne samo za konkretno rešenje, već i za dalje usavršavanje znanja iz razvoja web aplikacija i upravljanja produkcionim okruženjem.

LITERATURA

1. Vercel, *Next.js Documentation*, [online], dostupno na: <https://nextjs.org/docs>, pristupljeno 27. 8. 2026.
2. PostgreSQL Global Development Group, *PostgreSQL 16 Documentation*, [online], dostupno na: <https://www.postgresql.org/docs/16/>, pristupljeno 30. 8. 2026.
3. Prisma Data, Inc., *Prisma ORM 7 Documentation*, [online], dostupno na: <https://docs.prisma.io/docs/orm/v7>, pristupljeno 2. 9. 2026.
4. Better Auth, *Better Auth Documentation*, [online], dostupno na: <https://www.better-auth.com/docs>, pristupljeno 29. 8. 2026.
5. Zod, *Zod Documentation*, [online], dostupno na: <https://zod.dev/>, pristupljeno 31. 8. 2026.
6. Git Project, *Git Documentation*, [online], dostupno na: <https://git-scm.com/doc>, pristupljeno 26. 8. 2026.
7. Canonical Ltd., *Ubuntu Server Documentation – OpenSSH Server*, [online], dostupno na: <https://ubuntu.com/server/docs/how-to/security/openssh-server/>, pristupljeno 3. 9. 2026.
8. Docker, Inc., *Docker Documentation*, [online], dostupno na: <https://docs.docker.com/>, pristupljeno 1. 9. 2026.
9. NGINX, *NGINX Documentation*, [online], dostupno na: <https://nginx.org/en/docs/>, pristupljeno 4. 9. 2026.
10. Cloudflare, Inc., *Cloudflare Registrar and DNS Documentation*, [online], dostupno na: <https://developers.cloudflare.com/>, pristupljeno 28. 8. 2026.
11. Electronic Frontier Foundation, *Certbot Documentation*, [online], dostupno na: <https://certbot.eff.org/instructions>, pristupljeno 6. 9. 2026.
12. Let's Encrypt, *Let's Encrypt Documentation*, [online], dostupno na: <https://letsencrypt.org/docs/>, pristupljeno 5. 9. 2026.
13. Hetzner Online GmbH, *Hetzner Cloud Documentation – Servers*, [online], dostupno na: <https://docs.hetzner.com/cloud/servers/>, pristupljeno 7. 9. 2026.