Web3 aplikacija za vođenje projekata

Buhinjak, Karlo

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK
Stručni studij Računarstvo

WEB3 APLIKACIJA ZA VOĐENJE PROJEKATA

Završni rad

Karlo Buhinjak

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Karlo Buhinjak
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	AR 4773, 27.07.2020.
JMBAG:	0165084861
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, univ. mag. ing. comp.
Naslov završnog rada:	Web3 aplikacija za vođenje projekata
Znanstvena grana završnog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rada:	Aplikacija omogućuje vođenje projekata i treba sadržavati dnevne zadatke za projekt unutar tvrtke koji će biti dostupni svim zaposlenicima. Dnevni zadaci će se zapisivati na blockchain, tako se dobije sljedivost i mogu se pratiti odgovornosti na projektu. Također aplikacija će sadržavati chat za komunikaciju između zaposlenika tvrtke čije će se poruke zapisivati na blockchain. U aplikaciji će biti implementirana to-do lista za svakog zaposlenika radi lakšeg praćenja dnevnih zadataka. Tema je rezervirana za: Karlo buhinjak
Datum ocjene pismenog dijela završnog rada od strane mentora:	27.08.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	02.10.2024.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	07.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 07.10.2024.

Ime i prezime Pristupnika:	Karlo Buhinjak
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	AR 4773, 27.07.2020.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Web3 aplikacija za vođenje projekata**

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 pristupnika:

SADRŽAJ

1. UVOD 1
   1.1. Zadatak završnog rada 2
2. PREGLED PODRUČJA TEME 3
   2.1. Asana 3
   2.2. Trello 4
   2.3. Jira 5
3. TEHNOLOGIJE KORIŠTENE ZA RAZVOJ APLIKACIJE 7
   3.1. HTML 7
   3.2. CSS 8
   3.3. Web3 9
   3.4. JavaScript 10
      3.4.1. Node.js 11
      3.4.2. Express.js 12
      3.4.3. EJS 12
      3.4.4. Ethers.js 13
   3.5. MongoDB 14
   3.6. Solidity 15
   3.7. HardHat 16
   3.8. Volta Testnet 16
   3.9. Metamask 17
4. FUNKCIONALNOST APLIKACIJE 19
   4.1. Arhitektura aplikacije 19
   4.2. Upravljanje korisnicima 20
      4.2.1. Model korisnika 20
      4.2.2. Registracija korisnika 21
      4.2.3. Prijava korisnika 22
      4.2.4. Administratorske funkcije 23
   4.3. Zapisivanje podataka na blockchain 24
      4.3.1. Zapisivanje podataka o dnevnim zadacima 24
      4.3.2. Kreiranje projekata i postavljanje na blockchain 30
      4.3.3. Decentralizirana chat aplikacija 34
5. UPOTREBA APLIKACIJE 37

5.1. Prijava i registracija **37**
5.2. Korisnički dio **38**
5.2.1. Početni prikaz stranice 38
5.2.2. Prikaz trenutnih projekata 40
5.2.3. Korištenje timskog chata 41
5.2.4. Korištenje Todo liste 42
5.3. Administratorski dio **43**
5.3.1. Početni prikaz stranice 43
5.3.2. Stranica za dodavanje i postavljanje projekata na blockchain 43
5.3.3. Stranica za upravljanje zaposlenicima 45
5.3.4. Stranica za dodavanje zaposlenika 46
6. ZAKLJUČAK **47**
LITERATURA **49**
SAŽETAK **50**
ABSTRACT **51**
PRILOZI **52**
P.1. Izvorni kod aplikacije **52**

1. UVOD

U suvremenom digitalnom dobu, upravljanje projektima postaje sve kompleksnije i zahtijeva učinkovite alate i tehnologije kako bi se osigurala uspješna realizacija projekata. Organizacije danas djeluju u brzo promjenjivom okruženju, gdje fleksibilnost, agilnost i brzina donošenja odluka postaju ključni faktori uspjeha. Kako bi se prilagodili ovim zahtjevima, tradicionalni pristupi upravljanju projektima sve se više nadopunjuju modernim tehnološkim rješenjima koja omogućuju bolje planiranje, praćenje i izvršavanje projekata. Glavni izazovi u ovom procesu obuhvaćaju usklađivanje timova na udaljenim lokacijama, osiguranje integriteta podataka, održavanje visoke razine sigurnosti te transparentnost informacija unutar timova. Web3 tehnologija, zasnovana na blockchainu, pruža inovativna rješenja za mnoge industrije, uključujući i područje upravljanja projektima. Blockchain omogućava stvaranje distribuiranih i decentraliziranih sustava u kojima svi sudionici imaju jednak pristup informacijama, dok su podaci osigurani kriptografskim zapisima. Time se smanjuje potreba za centraliziranim posrednicima te se eliminira rizik manipulacije ili neovlaštenog pristupa podacima. Ove značajke čine Web3 tehnologiju izuzetno korisnom za razvoj aplikacija za vođenje projekata, osobito u okruženjima s mnogobrojnim sudionicima ili u industrijama gdje je povjerenje ključno. Jedna od ključnih prednosti Web3 tehnologije u upravljanju projektima jest transparentnost. Svi sudionici imaju uvid u tijek aktivnosti u realnom vremenu, što smanjuje rizik od nesporazuma i poboljšava suradnju. Decentralizirana struktura blockchaina povećava otpornost sustava, smanjujući osjetljivost na tehničke probleme i vanjske napade. Dodatno, blockchain omogućava provođenje pametnih ugovora, što olakšava automatizaciju ključnih procesa unutar projekta, kao što su odobravanje zadataka ili distribucija resursa. Uz tehničke prednosti, Web3 tehnologija uvodi novi pristup upravljanju projektima. Umjesto tradicionalne hijerarhijske strukture, gdje su informacije i odluke koncentrirane kod projektnih menadžera, Web3 potiče decentralizirano upravljanje. Time se povećava odgovornost među članovima tima, jer svi imaju jednaku priliku sudjelovati u donošenju odluka i pristupiti informacijama. Ovaj rad istražuje potencijal Web3 tehnologije u unapređenju upravljanja projektima kroz transparentnost, sigurnost i decentralizaciju. Fokus će biti na razvoju aplikacije koja koristi blockchain tehnologiju, s posebnim naglaskom na tehničke i sigurnosne aspekte njezine primjene. Detaljno će se analizirati postojeća rješenja za upravljanje projektima, kao i mogućnosti i prednosti korištenja blockchaina u ovom kontekstu. Praktični primjer pokazat

će kako Web3 tehnologija može transformirati način vođenja projekata unutar organizacija, uz osvrt na mogućnosti za daljnji razvoj i primjenu u budućnosti.

1.1. Zadatak završnog rada

Cilj ovog završnog rada je istražiti mogućnosti i prednosti upotrebe web3 tehnologije u razvoju aplikacije za vođenje projekata. U svrhu realizacije prethodno navedenog cilja, rad će obuhvatiti pregled postojećih rješenja za upravljanje projektima, istraživanje osnovnih koncepta web3 tehnologije te opis dizajna, arhitekture i implementacije web3 aplikacije za vođenje projekata.

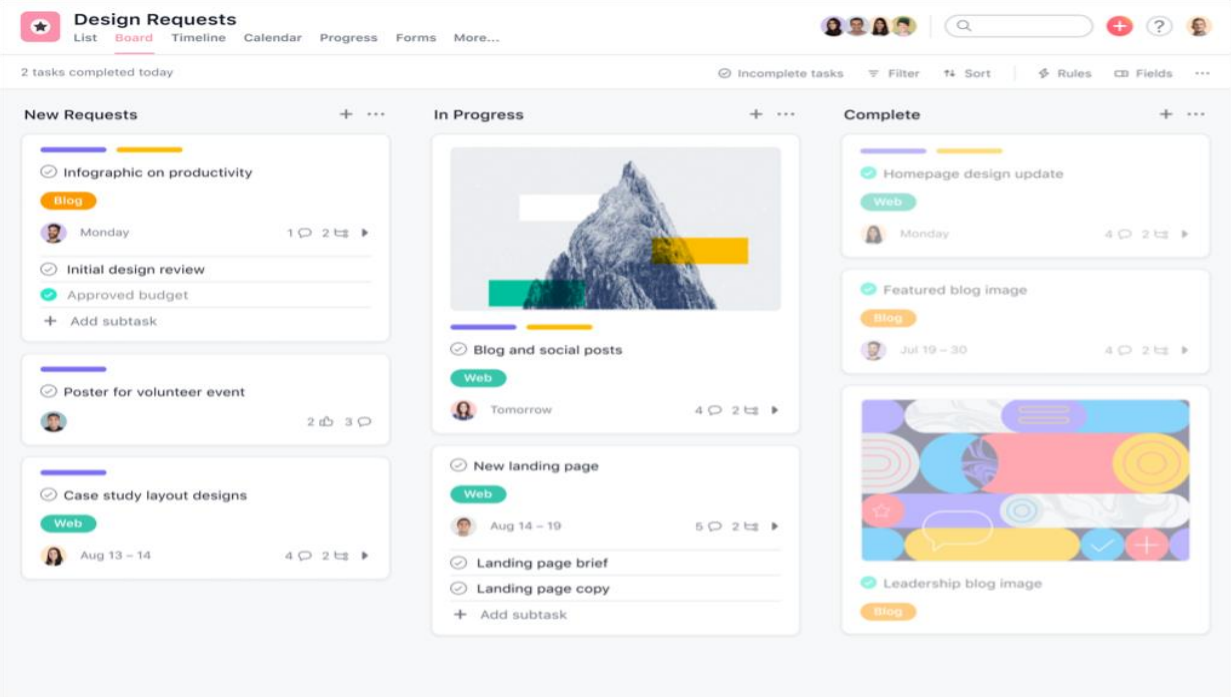
Kroz ovaj rad, nastoji se pružiti uvid u potencijal web3 tehnologije za transformaciju načina na koji se vođenje projekata odvija unutar organizacija te istaknuti mogućnosti za daljnji razvoj i primjenu ovakvih aplikacija u praksi.

2. PREGLED PODRUČJA TEME

U upravljanju projektima, pravilno odabrani alat može značajno olakšati i poboljšati produktivnost, suradnju i učinkovitost tima. Postoji veliki broj web aplikacije koje nude različite alate i funkcionalnosti za vođenje projekata. Trenutno na tržištu nema široko rasprostranjenih web3 rješenja specifično usmjerenih na vođenje projekata. Umjesto toga, najpopularnije web aplikacije koje se koriste za ovu svrhu su tradicionalni alati kao što su Asana, Trello i Jira. U ovom poglavlju, fokus je stavljen na tri prethodno navedena, danas izrazito popularna tradicionalna, alata te se pruža detaljan pregled karakteristika, prednosti i nedostataka istih.

2.1. Asana

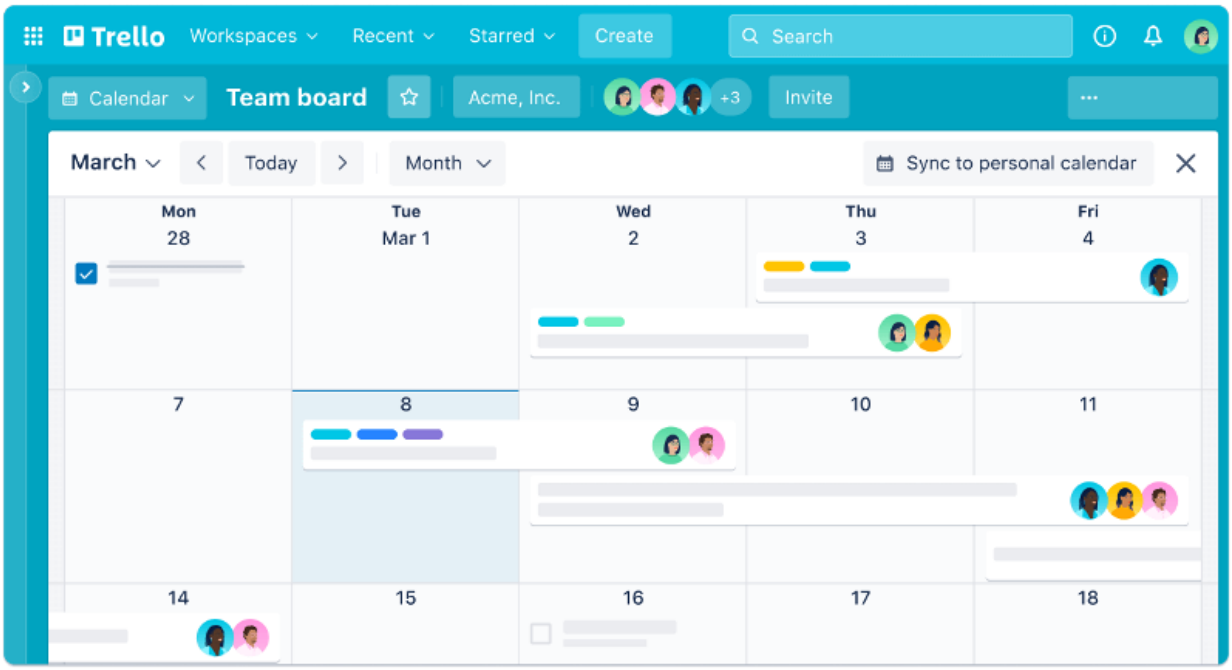
Asana je online platforma za upravljanje zadacima i projektima koja omogućava timovima organizaciju, praćenje i upravljanje vlastitim projektima. Koristeći intuitivno korisničko sučelje, Asana omogućava korisnicima da kreiraju liste zadataka, dodijele odgovornosti, postave rokove i komuniciraju unutar tima. Svaki zadatak može biti detaljno opisan, pridružen odgovornoj osobi, označen prioritetom i praćen napretkom. Asana također podržava integraciju s različitim alatima i aplikacijama, što je čini fleksibilnom i prilagodljivom za različite potrebe i industrije. Zahvaljujući svojoj jednostavnosti i učinkovitosti, Asana je postala jedan od vodećih alata za upravljanje projektima, posebno među timovima koji traže intuitivno i sveobuhvatno rješenje za organizaciju i koordinaciju svojih aktivnosti [1]. Slika 2.1. prikazuje izgled početne stranice Asana web aplikacije.



***Slika 2.1**. Početni prikaz aplikacije Asana*

2.2. Trello

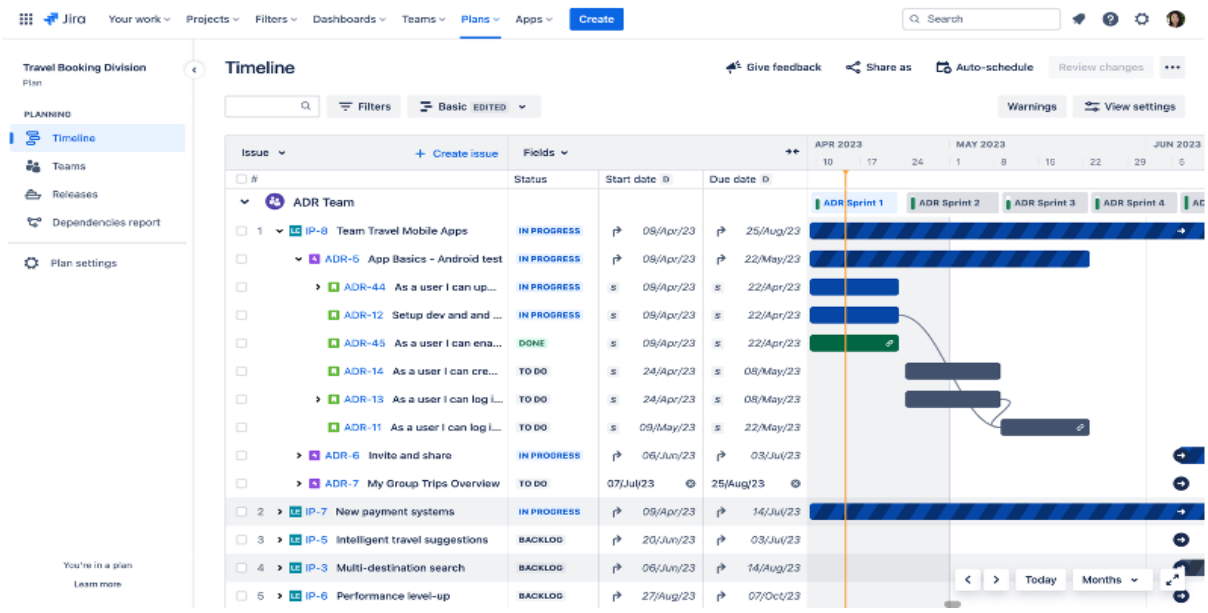
Trello je vizualna online platforma za upravljanje projektima koja koristi koncept "kanban" ploča za organizaciju i praćenje zadataka. Na Trello ploči, korisnici kreiraju kartice za svaki zadatak ili projekt koje mogu premještati između različitih kolona koje predstavljaju faze projekta (npr. "*To Do*", "*In Progress*", "*Done*"). Ove kartice mogu biti pridružene odgovornim osobama, označene prioritetom, opremljene s datotekama i komentirane kako bi se olakšala suradnja i komunikacija unutar tima. Trello je poznat po svojoj jednostavnosti, fleksibilnosti i vizualnoj prirodi, omogućavajući korisnicima da jasno vide i razumiju status i napredak svakog zadatka ili projekta na prvi pogled. Trello, također, podržava razne integracije s drugim alatima i aplikacijama, što ga čini prilagodljivim i kompatibilnim s različitim potrebama i radnim okruženjima [2]. Slika 2.2. prikazuje početni izgled Trello web aplikacije.



Slika 2.2. *Početni prikaz aplikacije Trello*

2.3. Jira

Jira je napredni online alat za upravljanje projektima, posebno popularan među razvojnim timovima. Jira pruža širok spektar funkcionalnosti za praćenje zadataka, bugova, razvojnih ciklusa i agilnih metodologija kao što su Scrum i Kanban. Koristeći Jira ploče, timovi mogu kreirati, organizirati i pratiti zadatak od početka do kraja, pridružujući im odgovorne osobe, postavljajući rokove, prioritet i status. Jira omogućava detaljno planiranje, praćenje i izvještavanje, te podržava integraciju s različitim razvojnim alatima i platformama. Zahvaljujući svojoj naprednoj funkcionalnosti i prilagodljivosti, Jira je postala nezaobilazni alat za organizacije koje traže sveobuhvatno rješenje za upravljanje projektima i razvojnim procesima, s mogućnostima za skaliranje i prilagodbu specifičnim potrebama i zahtjevima tima [3]. Slika 2.3. prikazuje početni izgled Jira web aplikacije.



Slika 2.3. *Početni prikaz aplikacije Jira*

3. TEHNOLOGIJE KORIŠTENE ZA RAZVOJ APLIKACIJE

U ovom poglavlju objašnjene su tehnologije koje su korištene za izradu aplikacije koja se prikazuje u ovom završnom radu.

3.1. HTML

HTML, skraćenica za *HyperText Markup Language*, predstavlja temeljni jezik kojim se kreiraju web stranice. Ova tehnologija omogućuje povezivanje i strukturiranje dokumenata kroz hipertekstualne veze. HTML dokumenti se sastoje od običnog teksta i posebnih naredbi, nazvanih oznakama, koje određuju kako će se dokument vizualno prikazati i kako će biti povezan s drugim resursima na internetu, poput slika, stilova i drugih HTML dokumenata.

HTML se često kombinira s CSS-om (eng. Cascading Style Sheets) kako bi se definirao izgled i stilizacija elemenata na stranici. Također, dodavanjem JavaScript-a, web stranice postaju interaktivne i dinamične, omogućavajući korisnicima bogato iskustvo pregledavanja sadržaja.

U konačnici, HTML predstavlja osnovu svake web stranice, omogućujući razvoj i dostupnost informacija širom svijeta. Bez njega, web bi izgubio svoju strukturu, smisao i funkcionalnost, čineći ga tek beznačajnim skupom podataka [4]. Slika 3.1. prikazuje logo HTML-a.

Slika 3.1. *Logo HTML - a*

3.2. CSS

CSS (eng. Cascading Style Sheets) je ključni jezik koji se koristi za stilizaciju web stranica. Dok HTML daje strukturu sadržaju, CSS pruža način da se taj sadržaj estetski oblikuje i stilizira.

Jedna od glavnih prednosti CSS-a je njegova sposobnost kaskadnog nasljeđivanja stilova. To znači da stilovi definirani na višim razinama, kao što su stilovi primijenjeni na tijelo cijele web stranice, mogu biti naslijeđeni i primijenjeni na podčinjene elemente unutar tog tijela. Na taj način, promjene na visokoj razini mogu se primijeniti na cijelu web stranicu, pružajući dosljedan dizajn.

CSS također omogućuje definiranje različitih stilova za različite medije. Na primjer, možemo definirati posebne stilove za zaslone računala i mobilne uređaje, osiguravajući da naša web stranica izgleda dobro na različitim uređajima. Kroz selektore i deklaracije, CSS pruža kontrolu nad dizajnom i izgledom web stranica. Selektori omogućuju odabir određenih elemenata na web stranici, dok deklaracije definiraju stilove koji se primjenjuju na te elemente. Ovo omogućuje izradu bogatih i estetskih korisničkih iskustava na internetu.

Napredne mogućnosti CSS-a uključuju animacije, prijelaze i fleksibilne mrežne sustave, koji omogućuju stvaranje dinamičnih i interaktivnih web stranica. Kroz kombinaciju HTML-a, CSS-a i JavaScripta, web dizajneri mogu stvoriti atraktivne i funkcionalne web stranice koje privlače i zadržavaju posjetitelje [5]. Slika 3.2. prikazuje logo CSS-a.

Slika 3.2. *Logo CSS – a*

3.3. Web3

Web3 predstavlja novu paradigmu internet tehnologije koja se temelji na konceptima decentralizacije, transparentnosti i autonomije. Dok je Web1 bio statičan i jednostavan, pružajući osnovne informacije korisnicima, te Web2 donio interaktivnost i društvene mreže, Web3 predstavlja sljedeću evoluciju interneta, gdje se naglasak stavlja na decentralizaciju podataka i aplikacija. Slika 3.3. prikazuje grafički prikaz razvoja web tehnologije.

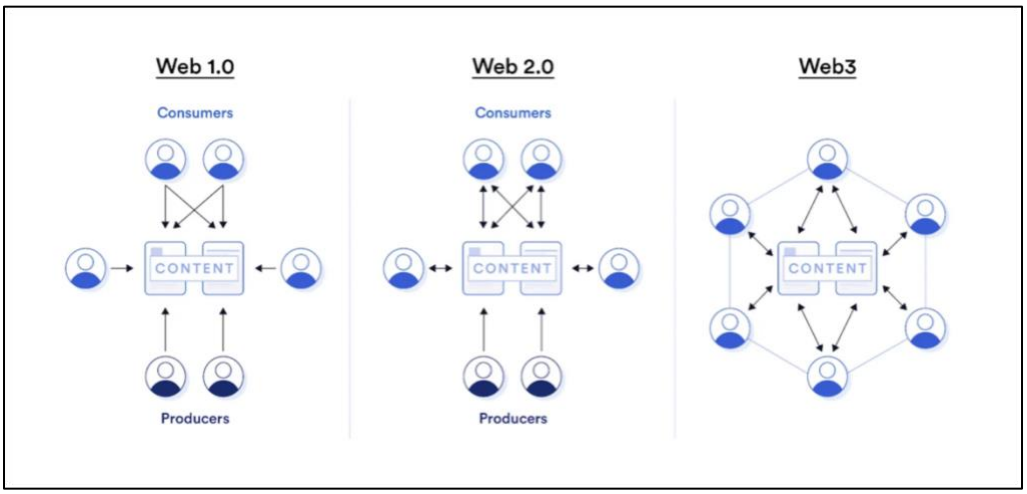
Ključna komponenta Web3 tehnologije je blockchain, distribuirana i sigurna baza podataka koja omogućuje pohranu transakcija i informacija bez potrebe za centralnim autoritetom. Blockchain tehnologija omogućuje stvaranje novih oblika digitalnih imovina, kao što su kriptovalute i NFT-ovi, te potiče razvoj decentraliziranih financijskih i aplikacijskih platformi.

Osim blockchain tehnologije, Web3 se oslanja na koncepte kao što su pametni ugovori, koji omogućuju automatizaciju i provođenje poslovnih transakcija bez posrednika, te decentralizirane aplikacije, koje su izgrađene na blockchain platformama i funkcioniraju bez centralnih servera.

Web3 tehnologija otvara vrata novim mogućnostima u područjima kao što su financije, umjetnost, igre, identitet, logistika i mnoga druga područja. Ona obećava veću privatnost i kontrolu korisnika nad vlastitim podacima, te potiče inovacije i kreativnost u digitalnom prostoru.

U kontekstu aplikacije za vođenje projekata, Web3 tehnologija može osigurati veću transparentnost i sljedivost zadataka putem blockchaina. Pametni ugovori mogu automatizirati procese i osigurati izvršavanje zadataka prema unaprijed definiranim pravilima, dok decentralizirana priroda aplikacije može poboljšati sigurnost i povjerenje među korisnicima.

Iako Web3 tehnologija donosi mnoge prednosti, suočava se i s izazovima kao što su skalabilnost, interoperabilnost i složenost implementacije [6]. Slika 3.3 prikazuje razvoj web tehnologija.



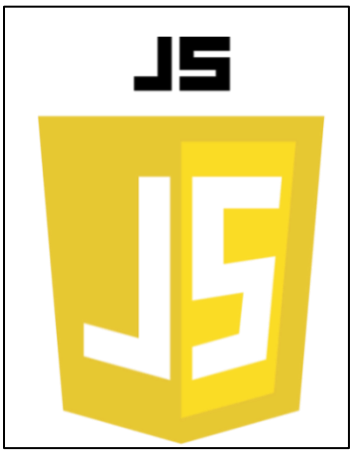
Slika 3.3. *Grafički prikaz razvoja web tehnologija*

3.4. JavaScript

JavaScript je programski jezik koji se često koristi za razvoj dinamičkih i interaktivnih web stranica. Omogućava programerima da dodaju funkcionalnosti kao što su validacija obrazaca, animacije, dinamičko ažuriranje sadržaja i interaktivni efekti. JavaScript se obično koristi zajedno s HTML-om za strukturiranje sadržaja i CSS-om za oblikovanje elemenata na web stranici.

Jedna od ključnih karakteristika JavaScripta je da se izvršava u pregledniku korisnika, što omogućava brzu reakciju na korisničke akcije bez potrebe za osvježavanjem stranice. Osim toga, JavaScript se može koristiti za razvoj složenijih web aplikacija, mobilnih aplikacija i serverskog programiranja putem okvira kao što je Node.js.

JavaScript je skriptni jezik, što znači da se izvršava u stvarnom vremenu, bez potrebe za prethodnim kompajliranjem. Podržava različite paradigme programiranja, uključujući objektno-orijentirano, funkcionalno i proceduralno programiranje. Zahvaljujući JavaScriptu, web stranice postaju interaktivnije i dinamičnije, pružajući korisnicima bogatije iskustvo [7]. Slika 3.4. prikazuje logo JavaScript programskog jezika.



Slika 3.4. *Logo JavaScript – a*

3.4.1. Node.js

Node.js je platforma temeljena na V8 JavaScript engineu koja omogućava izvršavanje JavaScript koda izvan web preglednika. Dizajniran je za razvoj skalabilnih i visoko učinkovitih serverskih aplikacija. Često se koristi za izgradnju web servera, aplikacija stvarnog vremena poput chat aplikacija, te za izradu API-ja. Njegova bogata kolekcija biblioteka i paketa, dostupna putem NPM-a (eng. Node Package Manager), dodatno pojednostavljuje razvoj i ubrzava proces implementacije [8]. Slika 3.5. prikazuje logo Node.js platforme.

Slika 3.5*. Logo Node.js - a*

3.4.2. Express.js

Express je minimalna i fleksibilna web biblioteka za Node.js koja pruža značajke za izradu web i mobilnih aplikacija. Dizajnirana je da olakša razvoj serverskih aplikacija, omogućujući jednostavno upravljanje rutama, zahtjevima i odgovorima. Express podržava middleware arhitekturu, što omogućuje proširenje funkcionalnosti aplikacije pomoću različitih modula. Popularna je zbog svoje jednostavnosti, brze implementacije i bogatog ekosustava dodataka i modula, čineći je idealnim izborom za izradu API-ja i web servera [9]. Slika 3.6. prikazuje jednostavan primjer Express.js koda.

Slika 3.6. *Prikaz Express.js koda*

3.4.3. EJS

EJS, ugrađeni JavaScript, popularan je templating jezik za generiranje HTML-a u Node.js okruženju. Omogućava dinamičko ubacivanje podataka u HTML dokumente pomoću JavaScript sintakse, olakšavajući izgradnju interaktivnih web aplikacija. EJS omogućava korištenje kontrolnih struktura poput petlji i uvjetnih naredba te podržava modularnost kroz korištenje parcijalnih predložaka. Parcijalni predlošci su dijelovi koji se mogu koristiti na više mjesta u aplikaciji, što olakšava ponovnu upotrebu koda i održavanje. Iako ima svoje prednosti, kao što su brzina razvoja i čitljivost koda, EJS može imati ograničenja u performansama u usporedbi s drugim templating jezicima. Unatoč tome, EJS ostaje popularan alat među programerima zbog svoje jednostavnosti i učinkovitosti [10]. Slika 3.7. prikazuje jednostavan primjer EJS koda.

Slika 3.7. *Prikaz EJS koda*

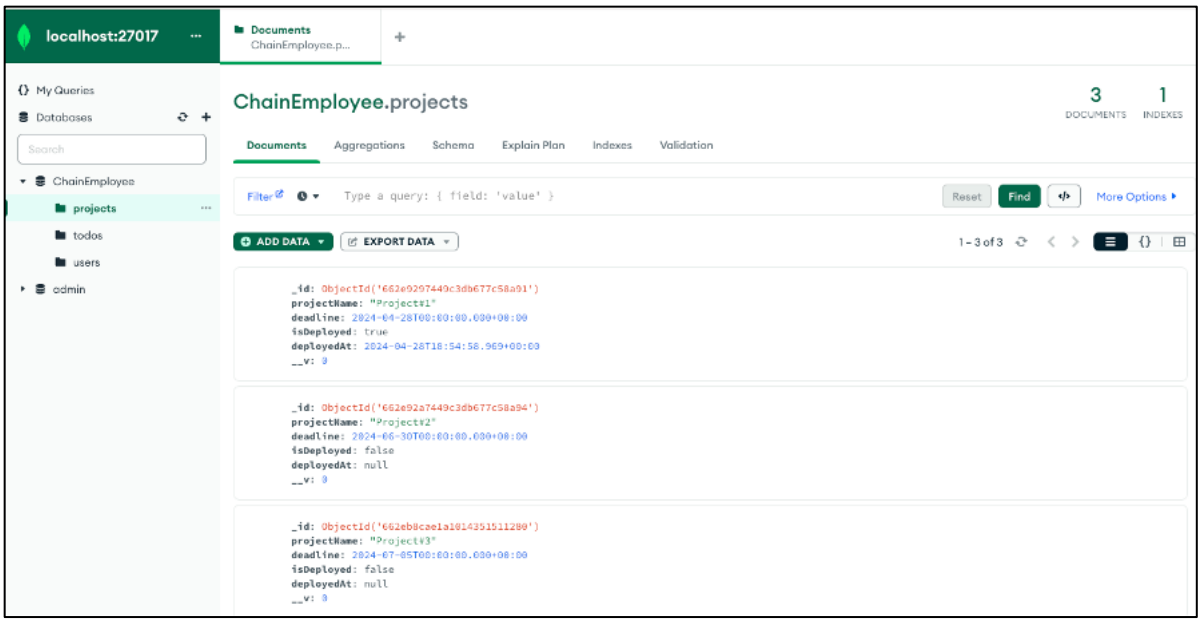
3.4.4. Ethers.js

Ethers.js je snažna JavaScript biblioteka koja omogućuje komunikaciju s Ethereum blockchainom. Sa svojim jednostavnim i intuitivnim API-jem, omogućuje programerima brzo i učinkovito izgradnju aplikacija za Ethereum ekosustav. Ethers.js pruža bogat set alata za rad s pametnim ugovorima, slanje i upravljanje Ethereum transakcijama, kao i rad s Ethereum novčanicima. Zbog svoje fleksibilnosti i performansi, Ethers.js je postao preferirani alat za razvoj decentraliziranih aplikacija i projekata koji koriste Ethereum blockchain [11]. Slika 3.8. prikazuje logo Ethers.js biblioteke.

Slika 3.8. *Logo Ethers.js*

3.5. MongoDB

MongoDB je popularna baza podataka koja se koristi za skladištenje podataka u obliku dokumenata. Dokument je struktura slična JSON formatu koja omogućuje fleksibilno čuvanje podataka bez stroge sheme. Kada se kombinira s Mongoose bibliotekom za Node.js, koja pruža alate za modeliranje podataka i olakšava komunikaciju s MongoDB-om, programeri mogu brzo i učinkovito izgraditi backend za svoje web aplikacije. Mongoose omogućuje definiranje modela podataka, validaciju unosa, definiranje veza između dokumenata te izvođenje kompleksnih upita nad bazom podataka. Korištenjem MongoDB-a i Mongoose-a u aplikaciji, programeri mogu implementirati backend sustav [12]. Slika 3.9. prikazuje MongoDB Compass grafičko sučelje za MongoDB bazu podataka.

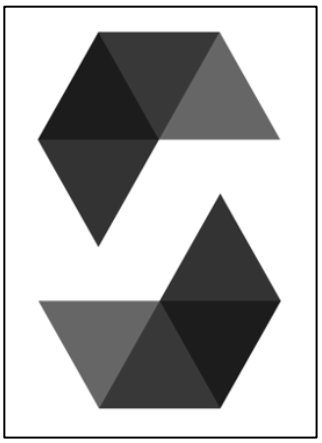


Slika 3.9. *Prikaz MongoDB Compass grafičkog sučelja za MongoDB bazu podataka*

3.6. Solidity

Solidity je programski jezik koji se koristi za razvoj pametnih ugovora na Ethereum platformi. Osnovni cilj Solidity programskog jezika je omogućiti programerima pisanje sigurnih, pouzdanih i učinkovitih pametnih ugovora. Solidity kombinira karakteristike jezika poput C++, Pythona i JavaScripta, čineći ga pristupačnim za programere različitih pozadina. Pametni ugovori napisani u Solidity programskom jeziku se kompajliraju u bytecode koji se izvršava na Ethereum Virtual Machine (EVM).

Solidity pruža bogat set alata i biblioteka za razvoj decentraliziranih aplikacija i omogućava programerima da stvaraju inovativne blockchain aplikacije na Ethereum platformi [13]. Slika 3.10 prikazuje logo Solidity programskog jezika.



Slika 3.10. *Logo Solidity - a*

3.7. HardHat

Hardhat je razvojni okvir (framework) za Ethereum pametne ugovore koji omogućava programerima jednostavno i učinkovito razvijanje, testiranje i implementaciju pametnih ugovora. Sa svojim bogatim setom alata i integracija, Hardhat olakšava automatizaciju testiranja, upravljanje okruženjima i interakciju s Ethereum mrežom [14].

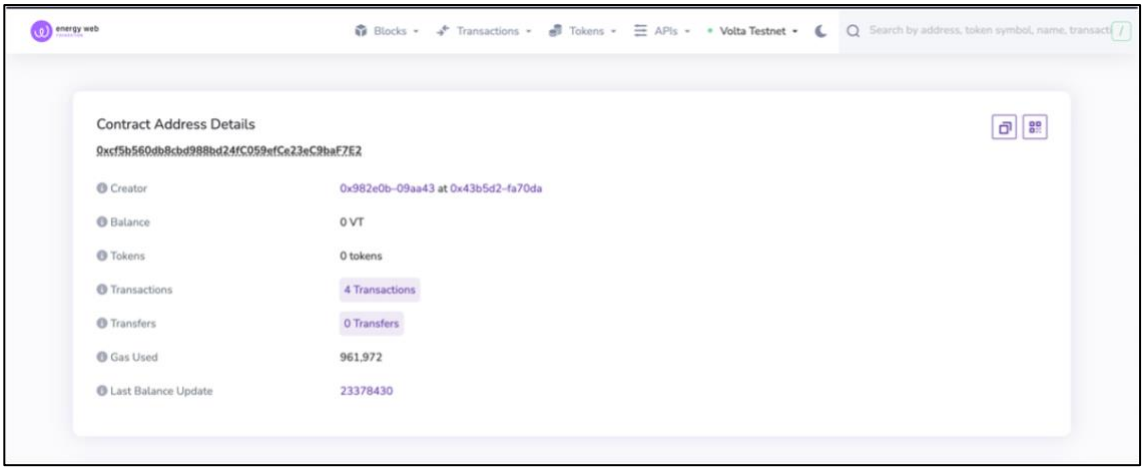
Terminal naredba "compile" omogućava kompajliranje pametnih ugovora, dok naredba "deploy" služi za njihovo postavljanje na mrežu.

3.8. Volta Testnet

Volta Testnet je Ethereum testna mreža koja omogućava programerima i korisnicima eksperimentiranje i testiranje aplikacija prije nego što ih implementiraju na glavnoj Ethereum mreži. Volta testna mreža je kompatibilan s Ethereum Virtual Machine (EVM) i podržava većinu Ethereum alata i pametnih ugovora. On pruža okruženje za testiranje performansi, funkcionalnosti i interoperabilnosti aplikacija na Ethereumu [15].

Korištenjem Volta testne mreže, korisnici mogu provjeriti funkcionalnost svojih pametnih ugovora, provesti testne transakcije i razviti sigurno i pouzdano rješenje prije nego što ga implementiraju na stvarnu Ethereum mrežu. Slika 3.11. prikazuje stranicu Volta Explorera za

pregled i istraživanje transakcija, blokova, adresa i drugih detalja povezanih s Volta testnom mrežom.



Slika 3.11. *Prikaz stranice Volta Explorera*

3.9. Metamask

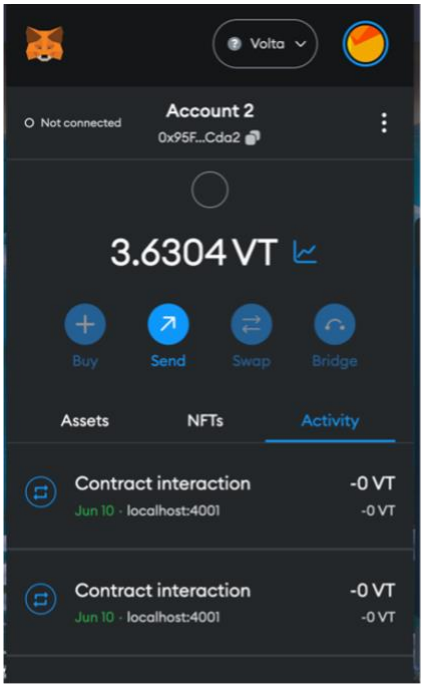
Metamask je popularan Ethereum novčanik i ekstenzija preglednika koja omogućava korisnicima jednostavno upravljanje Ethereum novčanicima i interakciju s decentraliziranim aplikacijama na Ethereum mreži.

S Metamaskom, korisnici mogu sigurno čuvati, slati i primati Ethereum (ETH) i ERC-20 tokene. Također, Metamask pruža intuitivno sučelje za upravljanje privatnim ključevima, potpisivanje transakcija i pregledanje transakcijske povijesti [16].

Kroz svoju integraciju s preglednicima, Metamask omogućava korisnicima jednostavan pristup Ethereum ekosustavu i prilagođavanje svojih financijskih aktivnosti na blockchainu.

U ovoj aplikaciji metamask služi za plaćanje transakcija postavljanja zapisa na blockchain.

Slika 3.12. prikazuje sučelje Metamask novčanika.



Slika 3.12. *Prikaz Metamask novčanika*

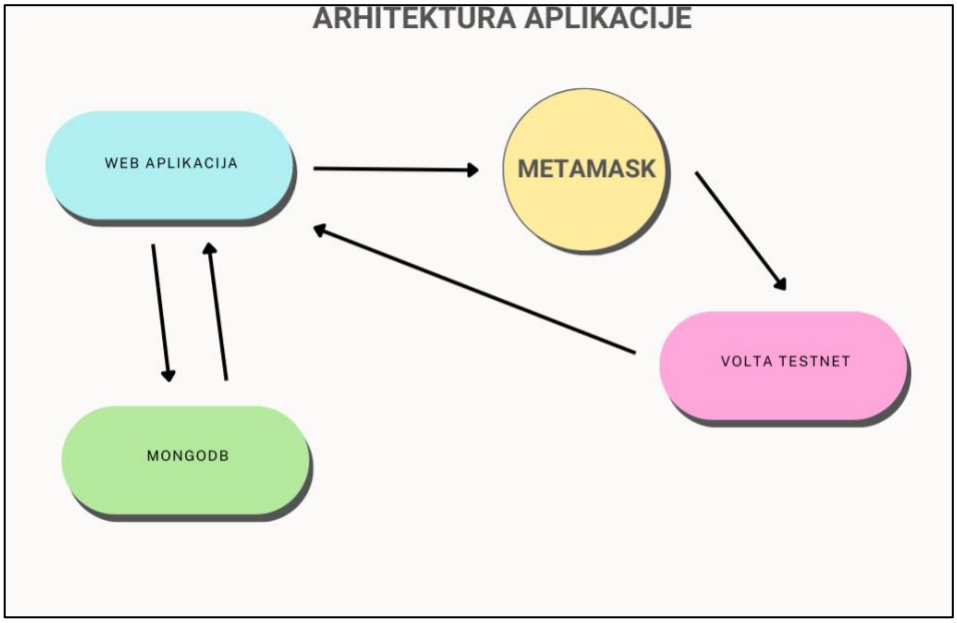
4. FUNKCIONALNOST APLIKACIJE

U ovom poglavlju opisuje se izrada aplikacije, uključujući ključne korake i tehnologije koje su korištene. Fokus je na na arhitekturi aplikacije, implementaciji funkcionalnosti, te načinu integriracije različitih komponenti poput blockchaina. Cilj je pružiti detaljan pregled procesa razvoja i objasniti kako su korištene tehnologije doprinijele ostvarivanju ciljeva projekta.

4.1. Arhitektura aplikacije

Web3 aplikacija omogućava vođenje projekata, posebno praćenje dnevnih zadataka i komunikaciju između zaposlenika. Podaci se pouzdano i transparentno pohranjuju na blockchain putem pametnog ugovora koji se nalazi na Volta testnoj mreži. Za izvršavanje transakcija i zapisivanje podataka koristimo Metamask kao digitalni novčanik.

Korisnici se mogu registrirati na aplikaciji, a podaci o korisnicima se sigurno pohranjuju u MongoDB bazi podataka. Također, na MongoDB-u se nalaze i administratori koji imaju ovlasti za zapisivanje podataka o projektima na blockchain. Ova kombinacija tehnologija omogućava visoku sigurnost, transparentnost i praktičnost u vođenju projekata. Arhitektura aplikacije je prikazana na slici 4.1.



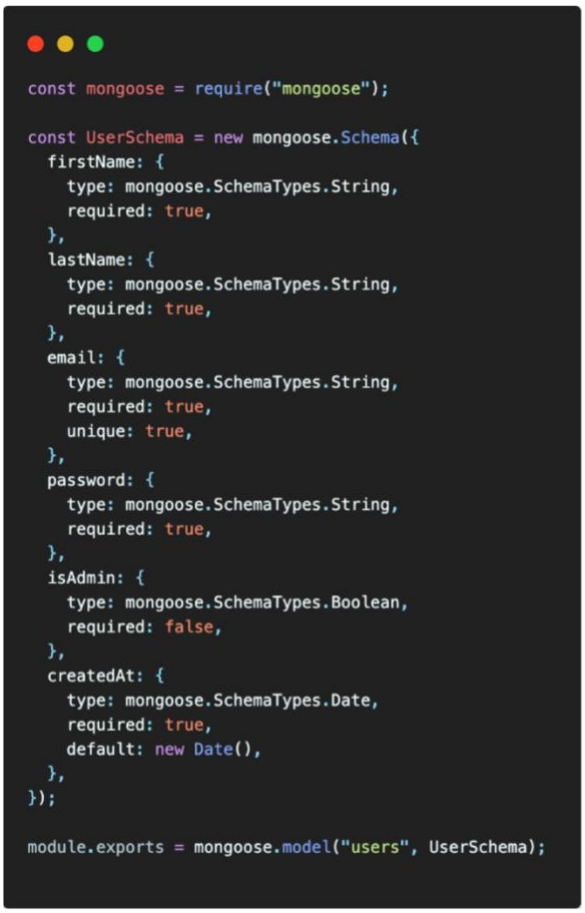
Slika 4.1. *Prikaz arhitekture aplikacije*

4.2. Upravljanje korisnicima

Registracija i prijava zaposlenika ključni su koraci za pristup aplikaciji i korištenje njenih funkcionalnosti. Ovaj proces osigurava da samo ovlašteni korisnici mogu pristupiti podacima i alatima unutar aplikacije. U nastavku su detaljno opisani procesi registracije i prijave korisnika, kao i metode za osiguranje sigurnosti lozinki.

4.2.1. Model korisnika

Definicija modela korisnika u MongoDB bazi podataka pomoću Mongoose-a omogućava strukturirano pohranjivanje korisničkih podataka. Model korisnika definira strukturu dokumenata koji se pohranjuju u bazi podataka, uključujući ime, prezime, e-mail, lozinku, rola i datum kreiranja (Slika 4.2.).



```
const mongoose = require("mongoose");

const UserSchema = new mongoose.Schema({
  firstName: {
    type: mongoose.SchemaTypes.String,
    required: true,
  },
  lastName: {
    type: mongoose.SchemaTypes.String,
    required: true,
  },
  email: {
    type: mongoose.SchemaTypes.String,
    required: true,
    unique: true,
  },
  password: {
    type: mongoose.SchemaTypes.String,
    required: true,
  },
  isAdmin: {
    type: mongoose.SchemaTypes.Boolean,
    required: false,
  },
  createdAt: {
    type: mongoose.SchemaTypes.Date,
    required: true,
    default: new Date(),
  },
});

module.exports = mongoose.model("users", UserSchema);
```

Slika 4.2. *Prikaz modela korisnika*

4.2.2. Registracija korisnika

Proces registracije zaposlenika omogućava novim korisnicima stvaranje računa u aplikaciji. Registracija je prvi korak za pristup aplikaciji i omogućuje zaposlenicima da unesu svoje podatke kako bi ih sustav mogao identificirati i autorizirati za daljnje aktivnosti. Korisnik dolaskom na početnu stranicu aplikacije može vidjeti obrazac za unos svojih podataka. Nakon unosa svojih podataka, sustav provjerava postoji li korisnik s istim emailom već u bazi. Ako korisnik već postoji, preusmjerava se na stranicu za prijavu uz poruku da je već registriran.

Implementaciju koda korisničkog sučelja za registraciju prikazana je na slici 4.3.

```
<form class="signup" action="/api/v1/auth/register" method="POST">
      <h3>Register</h3>

      <label for="firstName">First Name</label>
      <input type="text" id="firstName" name="firstName" required />
      <label for="lastName">Last Name</label>
      <input type="text" id="lastName" name="lastName" required />
      <label for="email">Email</label>
      <input type="email" id="email" name="email" required />
      <label for="password">Password</label>
      <input type="password" id="password" name="password" required />

      <button type="submit">Register</button>
    </form>
```

Slika 4.3. *Prikaz koda korisničkog sučelja za registriranje korisnika*

4.2.3. Prijava korisnika

Prijava korisnika omogućava registriranim zaposlenicima pristup aplikaciji. Prijava je ključna za autentifikaciju korisnika i osigurava da samo ovlašteni korisnici mogu pristupiti zaštićenim dijelovima aplikacije. Korisnik na stranici za prijavu unosi svoje podatke kako bi pristupio aplikaciji. Nakon unosa podataka, sustav provjerava unesene podatke prema pohranjenim podacima u bazi. Ako su podaci ispravni, korisnik se autentificira i preusmjerava na početnu stranicu aplikacije. U slučaju neuspješne prijave, korisniku se prikazuje poruka o pogrešci i omogućava mu se ponovni unos podataka. Slika 4.4 prikazuje EJS kod za prijavu korisnika.

```
<div class="m-4">
        <% if(successMessage){ %>
          <div class="alert alert-success alert-dismissible fade show">
              <strong>Success!</strong> <%= successMessage %>
              <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
          </div>
          <% } %>
        <form class="login" action="/api/v1/auth/login" method="POST">
          <h3>Log In</h3>
          <label for="email">Email</label>
          <input type="email" name="email" id="" email required />
          <label for="password">Password</label>
          <input type="password" id="password" name="password" />
          <button type="submit">Login</button>
          </form>
          <% if(errorMesssage){ %>
          <div class="error"><%= errorMesssage %></div>
        <%} %>
    </div>
```

Slika 4.4. *Prikaz koda korisničkog sučelja za prijavu korisnika*

4.2.4. Administratorske funkcije

Administratorske funkcije unutar aplikacije omogućavaju korisnicima s administratorskim ovlastima da upravljaju ključnim aspektima sustava, poput dodavanja novih projekata, upravljanja korisnicima i zapisivanja podataka na blockchain.

Ove funkcije su postignute dodavanjem polja "isAdmin" tipa boolean u model korisnika te korištenjem EJS uvjetnog renderiranja elemenata na stranici. Programski kod za uvjetno renderiranje prikazan je na slici 4.5 i prikazuje dio za dodavanje novih projekata koji su vidljivi samo administratoru.

```
<% if (user.isAdmin) { %>
            <a href="/api/v1/projects/deployed" class="nav-button"
              >Deployed Projects</a
            >
            <a href="/api/v1/employees/" class="nav-button">Employees</a>
            <a href="/api/v1/employees/create" class="nav-button"
              >Create Employee</a
            >
 <% } %>
```

Slika 4.5. *Prikaz EJS koda za uvjetno renderiranje*

4.3. Zapisivanje podataka na blockchain

Zapisivanje podataka na blockchain ključno je za osiguranje integriteta, sigurnosti i transparentnosti u aplikaciji. U kontekstu ove aplikacije, podaci o dnevnim zadacima, podaci o isporučenim projektima, te poruke između zaposlenika spremaju se na blockchain koristeći pametne ugovore. Ovaj pristup omogućava da svi unosi budu trajni, nepromjenjivi i javno provjerljivi. U sljedećim poglavljima detaljno se opisuje kako se svaki od navedenih tipova podataka zapisuje na blockchain.

4.3.1. Zapisivanje podataka o dnevnim zadacima

Zapisivanje podataka o dnevnim zadacima na blockchain omogućava visok stupanj transparentnosti i sigurnosti, jer svi unosi postaju trajni i nepromjenjivi. U ovoj aplikaciji koristi se pametni ugovor na Volta testnoj mreži za zapisivanje dnevnih zadataka.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;

contract DailyTaskContract {
    struct DailyTask {
        uint256 id;
        uint256 date;
        string projectName;
        string description;
        string employeeName;
        string employeeLastName;
    }

    mapping(uint256 => DailyTask) public dailyTasks;

    uint256 public taskCount;

    mapping(string => uint256[]) private tasksByProject;

    function addDailyTask(string memory _projectName, string memory _description, string memory _employeeName, string memory _employeeLastName) public {
        dailyTasks[taskCount] = DailyTask(taskCount, block.timestamp, _projectName, _description, _employeeName, _employeeLastName);
        tasksByProject[_projectName].push(taskCount);
        taskCount++;za
    }

    function getAllTasks() public view returns (DailyTask[] memory) {
        DailyTask[] memory tasks = new DailyTask[](taskCount);
        for (uint256 i = 0; i < taskCount; i++) {
            tasks[i] = dailyTasks[i];
        }
        return tasks;
    }

    function getTasksByEmployee(string memory _employeeName, string memory _employeeLastName) public view returns (DailyTask[] memory) {
        uint256 count = 0;
        for (uint256 i = 0; i < taskCount; i++) {
            if (keccak256(abi.encodePacked(dailyTasks[i].employeeName)) == keccak256(abi.encodePacked(_employeeName)) &&
                keccak256(abi.encodePacked(dailyTasks[i].employeeLastName)) == keccak256(abi.encodePacked(_employeeLastName))) {
                count++;
            }
        }

        DailyTask[] memory tasks = new DailyTask[](count);
        uint256 index = 0;
        for (uint256 i = 0; i < taskCount; i++) {
            if (keccak256(abi.encodePacked(dailyTasks[i].employeeName)) == keccak256(abi.encodePacked(_employeeName)) &&
                keccak256(abi.encodePacked(dailyTasks[i].employeeLastName)) == keccak256(abi.encodePacked(_employeeLastName))) {
                tasks[index] = dailyTasks[i];
                index++;
            }
        }

        return tasks;
    }

    function getTasksByProject(string memory _projectName) public view returns (DailyTask[] memory) {
        uint256[] storage taskIds = tasksByProject[_projectName];
        uint256 count = taskIds.length;

        DailyTask[] memory tasks = new DailyTask[](count);

        for (uint256 i = 0; i < count; i++) {
            tasks[i] = dailyTasks[taskIds[i]];
        }

        return tasks;
    }
}
```

Slika 4.6. *Prikaz Solidity koda za spremanje dnevnih zadataka*

Pametni ugovor je napisan u Solidity programskom jeziku i omogućava pohranu podataka o dnevnim zadacima. Svaki zadatak sadrži podatke kao što su naziv projekta, opis zadatka, ime i prezime zaposlenika. Slika 4.6. prikazuje implementaciju pametnog ugovora.

Obrazac za dodavanje dnevnih zadataka omogućava zaposlenicima unos potrebnih podataka. Obrazac je implementiran koristeći EJS (Embedded JavaScript) templating engine i prikazan na slici 4.7.

```
<% if (!user.isAdmin) { %>
        <form class="create" id="taskForm">
          <h3>Add a New Daily Task</h3>

          <label>Project Name:</label>
          <select id="project" name="projectName">
            <% projects.forEach(project => { %>
            <option value="<%= project.projectName %>">
              <%= project.projectName %>
            </option>
            <% }); %></select
          ><br />

          <label>Description:</label>
          <textarea
            id="description"
            name="description"
          ></textarea>
          <br />

          <div class="loading-spinner" id="loadingSpinner"></div>

          <button type="submit" id="submitButton">Submit</button>
        </form>
        <% } %>
```

Slika 4.7. *Prikaz EJS koda za dodavanje novih dnevnih zadataka*

U prvom retku koristi se EJS sintaksa za uvjetno prikazivanje određenog dijela HTML koda. Ova sintaksa omogućuje nam da dinamički kontroliramo prikaz elemenata ovisno o određenim uvjetima.

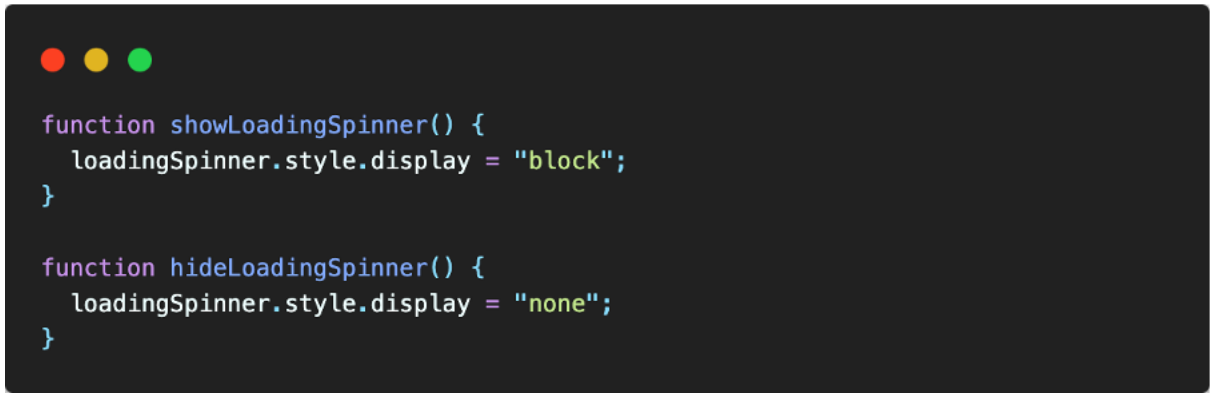
Klikom na gumb "Submit" u obrascu koji je prikazan u kodu, aktivira se skripta koja je postavljena kao listener na događaj slanja obrasca (submit event).

Skripta je podijeljena na tri dijela:

1. Dohvaćanje elementa obrasca, indikatora učitavanja i gumba za slanje (Slika 4.8)
2. Definiranje funkcija za prikazivanje i skrivanje indikatora za učitavanje (Slika 4.9)
3. Postavljanje logike slušatelja događaja na “potvrdu” obrasca (Slika 4.10)

```
const taskForm = document.getElementById("taskForm");
const loadingSpinner = document.getElementById("loadingSpinner");
const submitButton = document.getElementById("submitButton");
```

Slika 4.8. *Prikaz JavaScript koda za dohvaćanje elemenata*



```
function showLoadingSpinner() {
  loadingSpinner.style.display = "block";
}

function hideLoadingSpinner() {
  loadingSpinner.style.display = "none";
}
```

Slika 4.9. *Prikaz JavaScript funkcija prikazivanje i skrivanje indikatora za učitavanje*

```
taskForm.addEventListener("submit", async (event) => {
   event.preventDefault();
   showLoadingSpinner();

   submitButton.disabled = true;

   const projectName = document.getElementsByName("projectName")[0].value;
   const description = document.getElementsByName("description")[0].value;
   const firstName = "<%=user.firstName%>";
   const lastName = "<%=user.lastName%>";
   const contract_address = "<%= process.env.DAILY_TASK_CONTRACT_ADRESS%>";

   try {
     await ethereum.request({ method: "eth_requestAccounts" });
     const provider = new ethers.providers.Web3Provider(window.ethereum);
     const signer = provider.getSigner();
     const contractInstance = new ethers.Contract(
       contract_address,
       abi,
       signer
     );

     const transaction = await contractInstance.addDailyTask(
       projectName,
       description,
       firstName,
       lastName
     );

     await transaction.wait();

     taskForm.reset();
   } catch (error) {
     console.log(error);
   } finally {
     hideLoadingSpinner();
     submitButton.disabled = false;
     window.location.reload();
   }
 });
```

Slika 4.10. *Prikaz JavaScript koda za zapisivanje podataka na blockchain*

Ova skripta postavlja listener na submit događaj obrasca. Kada korisnik klikne na gumb "Submit", aktivira se funkcija koja se izvršava asinkrono.

U funkciji, prvo se sprječava preusmjeravanje obrasca (event.preventDefault()) i prikazuje indikator za učitavanje (showLoadingSpinner()). Zatim se dohvaćaju vrijednosti unesene u obrasac koristeći FormData. Nakon toga slijedi obrada na Ethereum mreži.

Skripta koristi ethereum.request zahtjev za pristup Ethereum računu korisnika. U praksi to znači da će se na ekranu prikazati metamask prozor i od korisnika zatražiti pristupne podatke. Nakon toga, inicijalizira se Web3 provider i signer pomoću ethers.providers.Web3Provider i

provider.getSigner(). Slijedi inicijalizacija pametnog ugovora i poziv funkcije addDailyTask na pametnom ugovoru, s proslijeđenim parametrima.

Nakon uspješnog dodavanja dnevnog zadatka, obrasac se resetira (taskForm.reset()), a spinner za učitavanje se skriva (hideLoadingSpinner()) i omogućuje se ponovno aktiviranje gumba za slanje (submitButton.disabled = false). Također, može se obaviti ponovno učitavanje stranice ako je to potrebno (window.location.reload()).

Ovaj kod pokazuje primjer kako se koristi JavaScript i Web3 biblioteka za interakciju s Ethereum mrežom, omogućavajući dodavanje zadataka putem pametnog ugovora.

U skriptu koja se izvodi klikom na "potvrdi", od velike je važnosti priložiti ABI (binarno programsko sučelje) pametnog ugovora, koji je potreban prilikom instanciranja, a nalazi se u datoteci zajedno s ugovorom u JSON formatu.

Osim toga, važno je napomenuti sigurnosne aspekate prilikom rada s Metamaskom i privatnim ključevima. Iz sigurnosnih razloga, nije poželjno pohranjivati privatne ključeve Metamaska u bazi podataka.

Privatni ključ Metamaska predstavlja osnovni mehanizam za pristup i upravljanje korisnikovim računima na Ethereum mreži. Pohranjivanje privatnih ključeva u bazi podataka predstavljalo bi potencijalnu sigurnosnu ranjivost jer bi zlonamjerni napadači mogli pokušati neovlašteno pristupiti tim ključevima.

Umjesto toga, koristi se Metamask kao sučelje za interakciju s privatnim ključevima korisnika. Metamask je razvijen kao sigurna klijentska aplikacija koja drži privatne ključeve na korisnikovom računalu ili mobilnom uređaju. Prilikom izvršavanja transakcija, korisnik će biti upitan za odobrenje i potpisivanje transakcije putem Metamaska, čime se osigurava da privatni ključ ostaje siguran na korisnikovom uređaju.

Tako, sigurnosni aspekti su očuvani jer privatni ključevi ostaju u rukama korisnika, a aplikacija koristi Metamask kao sigurno sučelje za interakciju s Ethereum mrežom. To omogućuje pružanje sigurne i pouzdane usluge bez potrebe za pohranjivanjem privatnih ključeva u bazu podataka.

Express.js, kao backend framework, omogućuje obradu zahtjeva i generiranje odgovora na serveru. Međutim, izravna interakcija s Metamaskom putem backenda nije moguća jer Metamask je klijentska aplikacija koja radi na korisnikovom pregledniku.

Da bi se postigla željena funkcionalnost, potrebno je uspostaviti međuprostor između backenda i frontenda koji će omogućiti komunikaciju s Metamaskom. Na frontendu upotrebom JavaScript i Web3 biblioteka može se ostvarili interakcija s Ethereum mrežom putem Metamaska.

Kada korisnik pristupi aplikaciji i želi izvršiti određenu radnju koja zahtijeva interakciju s Metamaskom, frontend će inicirati zahtjev za dohvaćanjem podataka s Metamaska, poput adrese korisnika i potpisivanja transakcija. Zatim će podaci biti poslani na Ethereum mrežu radi izvršavanja željene operacije.

Ovaj međuprostor omogućuje interakciju između klijentskog dijela aplikacije, Metamaska i Ethereum mreže. Express.js, kao backend framework, i dalje ima svoju važnost u obradi drugih zahtjeva i pružanju potrebne logike na serveru, ali za interakciju s Metamaskom i Ethereum mrežom, koristi se JavaScript na frontendu.

Ova arhitektura omogućuje da se integrira Metamask i Ethereum funkcionalnost u aplikaciju, pružajući korisnicima mogućnost interakcije s blockchain mrežom i potvrdu transakcija na siguran način.

4.3.2. Kreiranje projekata i postavljanje na blockchain

U ovom poglavlju objašnjava se kako se projekt kreira putem administratorske funkcije i kako se postavlja na blockchain koristeći pametni ugovor.

1. Kreiranje projekta
2. Model projekta
3. Backend za kreiranje projekta
4. Prikaz projekta i postavljanje na blockchain
5. Pametni ugovor za postavljanje projekata
6. Komunikacija s pametnim ugovorom

Na početku je potrebno kreirati projekt. Administratori imaju mogućnost kreiranja novih projekata koristeći obrasac koji je implementirana u EJS-u. Slika 4.11 prikazuje EJS kod za kreiranje projekta.

```
<% if (user.isAdmin) { %>
      <form
        class="create"
        id="taskForm"
        action="/api/v1/projects/create"
        method="POST"
      >
        <h3>Add New Project</h3>
        <label>Project Name:</label>
        <input type="text" name="projectName" /><br />
        <label>Deadline:</label>
        <input type="date" id="deadline" name="deadline" required />
        <br />

        <div class="loading-spinner" id="loadingSpinner"></div>

        <button type="submit" id="submitButton">Submit</button>
      </form>
    <% } %>
```

Slika 4.11. *Prikaz EJS koda za dodavanje novog projekta*

Obrazac omogućava administratoru unos imena projekta i rok završetka projekta. Nakon unosa podaci se šalju serveru gdje se spremaju u MongoDB bazu podataka.

Nadalje, struktura podataka koji se spremaju na bazu definirani su modelom projekta. Model projekta u MongoDB bazi podataka prikazana je na slici 4.12.

```
const mongoose = require("mongoose");

const projectSchema = new mongoose.Schema({
  projectName: {
    type: String,
    required: true,
  },
  deadline: {
    type: Date,
    required: true,
  },
  isDeployed: {
    type: Boolean,
    default: false,
  },
  deployedAt: {
    type: Date,
    default: null,
  },
});

const Project = mongoose.model("Project", projectSchema);

module.exports = Project;
```

Slika 4.12. *Prikaz modela projekta*

Funkcija za kreiranje projekta na serveru prikazana je na slici 4.13.

```
const createProject = async (req, res) => {
  try {
    const { projectName, deadline } = req.body;
    const newProject = new Project({
      projectName,
      deadline,
    });
    await newProject.save();
    res.redirect("/api/v1/projects/");
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: "Failed to create project" });
  }
};
```

Slika 4.13. *Prikaz server koda za dodavanje novog projekta*

Nakon završetka svakog projekta, administrator može postaviti projekt na blockchain i tako spremiti dnevne zadatke i zapise svakog zaposlenika za taj projekt.

Slika 4.14 prikazuje EJS kod za ispis projekta i mogućnost postavljanja projekata na blockchain.

```
<% if (projects.length !== 0) { %> <% for (let i = projects.length -
        1; i >=0; i--) { %>
        <div class="details">
          <a
            href="/api/v1/tasks/project?projectName=<%= encodeURIComponent(projects[i].projectName)  %>"
            style="text-decoration: none;"
          >
            <h4><%= projects[i].projectName %></h4>
          </a>

          <p>
            <strong>Deadline: </strong><%= new
            Date(projects[i].deadline).toLocaleDateString() %>
          </p>

          <form class="deploy-form">
            <% if (user.isAdmin) { %>
            <button class="deploy-button">Deploy</button>
            <%}%>
          </form>
        </div>
        <% } %> <% } else { %>
        <h2>No result for your search: <%=searchParameter%></h2>
  <% } %>
```

Slika 4.14. *Prikaz EJS koda za prikazivanje i postavljanje projekata na blockchain*

Postavljanje projekata na blockchain vrši se putem pametnog ugovora. Solidity kod za zapis projekta prikazan je na slici 4.15.

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;

contract ProjectRegistryContract {
    struct Project {
        string projectName;
        uint256 deploymentDate;

    }

    Project[] private projects;

    event DeployProject(string projectName);

    function deployProject(string memory projectName) public {
        Project memory newProject = Project({
            projectName: projectName,
            deploymentDate: block.timestamp
        });

        projects.push(newProject);
        emit DeployProject(projectName);
    }

    function getAllProjects() public view returns (Project[] memory) {
        return projects;
    }
}
```

Slika 4.15. *Prikaz Solidity koda za spremanje projekata na blockchain*

Za komunikaciju s pametnim ugovorom koriste se JavaScript i Web3 biblioteka, slično kao što smo opisali u prethodnom poglavlju. Korisnik koristi Metamask za autorizaciju transakcija, a JavaScript na frontendu inicira zahtjeve prema Ethereum mreži.

4.3.3. Decentralizirana chat aplikacija

U aplikacijama za vođenje projekata, chat za komunikaciju zaposlenika igra ključnu ulogu u olakšavanju suradnje i razmjene informacija. Međutim, u digitalnom okruženju gdje se svaka interakcija može promatrati kao potencijalni trag aktivnosti, sigurnost i pouzdanost postaju ključni faktori. Upravo ovdje dolazi do izražaja integracija blockchain tehnologije u chat aplikaciju.

Proces implementacije chat aplikacije koja koristi blockchain tehnologiju za pohranu podataka sastoji se od tri koraka kako slijedi:

1. Kreiranje poruke
2. Pametni ugovor za zapisivanje poruke na blockchainu
3. Komunikacija s pametnim ugovorom

U nastavku poglavlja detaljno se objašnjava proces implementacije chat aplikacije koja koristi blockchain tehnologiju za pohranu poruka.

Proces započinje kreiranjem poruke - kada korisnik posjeti stranicu timskog chata, može vidjeti sve poruke i ima mogućnost slanja poruka svojim kolegama. Na raspolaganju mu je obrasac za unos poruke, a jednostavnim klikom na dugme "Send" šalje svoju poruku. Prikaz implementacije chat aplikacije pomoću EJS-a prikazan je na slici 4.16.

```ejs
<div id="chat">
      <h2 id="chat-title">Team chat</h2>
      <div id="chat-window">
        <div id="output">
          <% if ( messages.length > 0) { %> <% messages.forEach(message => { %>
          <p>
            <strong><%= message.firstName %> <%= message.lastName %>: </strong
            ><%= message.message %>
          </p>
          <% }); %> <% } else { %>
          <p>No messages available</p>
          <% } %>
        </div>
      </div>
      <form class="create" id="messageForm">
        <input id="message" type="text" placeholder="Message" name="message" />
        <div class="loading-spinner" id="loadingSpinner"></div>
        <button id="send">Send</button>
      </form>
    </div>
```

Slika 4.16. *Prikaz EJS koda za prikazivanje i dodavanje novih poruka*

Zapisivanje poruke na blockchain se izvodi pomoću Solidity-a. Solidity kod za zapis projekta prikazan je na slici 4.17.

```solidity
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;

contract TeamChatContract {
    struct Message {
        string firstName;
        string lastName;
        string message;
    }

    Message[] private teamMessages;

    event MessageSent(string firstName, string lastName, string message);

    function sendMessage(string memory firstName, string memory lastName, string memory message) public {
        Message memory newMessage = Message({
            firstName: firstName,
            lastName: lastName,
            message: message
        });

        teamMessages.push(newMessage);
        emit MessageSent(firstName, lastName, message);
    }

    function getMessages() public view returns (Message[] memory) {
        return teamMessages;
    }
}
```

Slika 4.17. *Prikaz Solidity koda za spremanje poruka na blockchain*

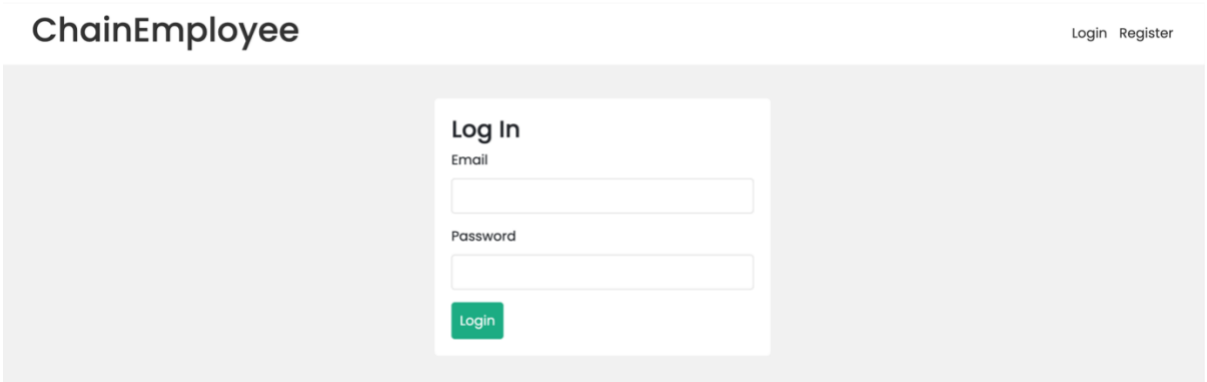
Za komunikaciju s pametnim ugovorom koriste se JavaScript i Web3 biblioteka, slično kao što smo opisali u prethodnim poglavljima. Korisnik koristi Metamask za autorizaciju transakcija, a JavaScript na frontendu inicira zahtjeve prema Ethereum mreži.

5. UPOTREBA APLIKACIJE

U ovom poglavlju detaljnos se opisuje kako koristiti aplikaciju za vođenje projekata, uključujući specifične funkcionalnosti dostupne administratorima i korisnicima. Korisnici mogu dodavati dnevne zadatke za određene projekte te pregledavati zadatke drugih zaposlenika. Također, imaju mogućnost komuniciranja putem timskog chata i praćenja svojih zadataka putem osobne ToDo liste. S druge strane, administratori imaju pristup naprednim opcijama kao što su kreiranje i upravljanje projektima te postavljanje projekata na blockchain. Podjela na administratorski i korisnički dio omogućava jasno razdvajanje uloga i odgovornosti unutar aplikacije.

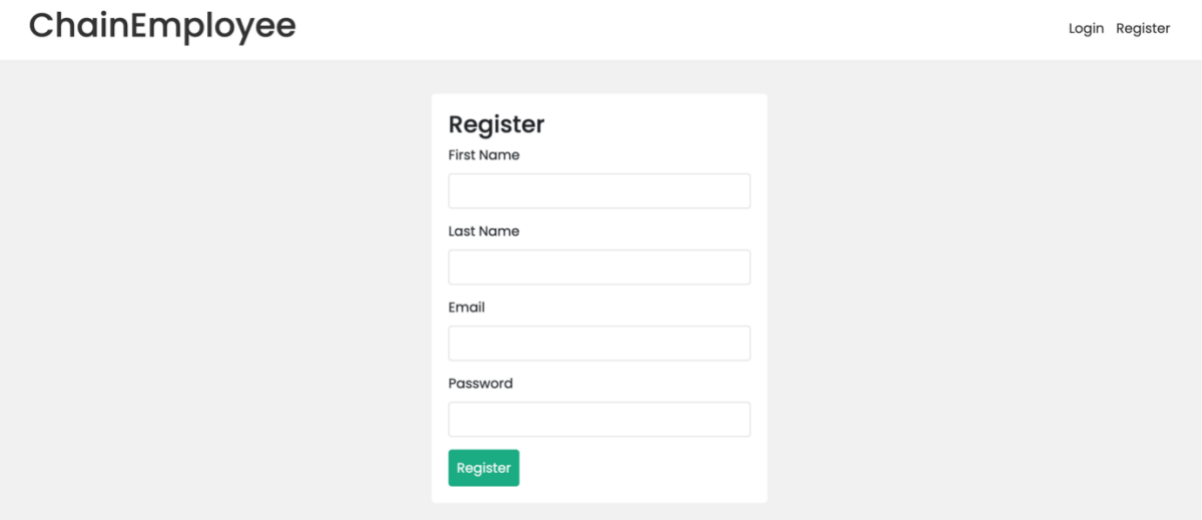
5.1. Prijava i registracija

Kada korisnik pristupi aplikaciji, prvo se prikazuje stranica za prijavu gdje unosi svoje podatke (Slika 5.1). Ako korisnik unese pogrešne podatke, prikazuje se poruka koja ga obavještava da je e-mail adresa ili lozinka netočno unesena.



Slika 5.1. *Prikaz stranice za prijavu*

Klikom na gumb "Register" u navigacijskoj traci, korisnik se preusmjerava na stranicu za registraciju. Na stranici za registraciju, korisnik treba unijeti svoje ime, prezime, e-mail adresu i lozinku (Slika 5.2). Ako su svi podaci ispravno uneseni, prikazuje se poruka o uspješnoj registraciji, a korisnik se preusmjerava na stranicu za prijavu gdje se može prijaviti i početi koristiti aplikaciju.

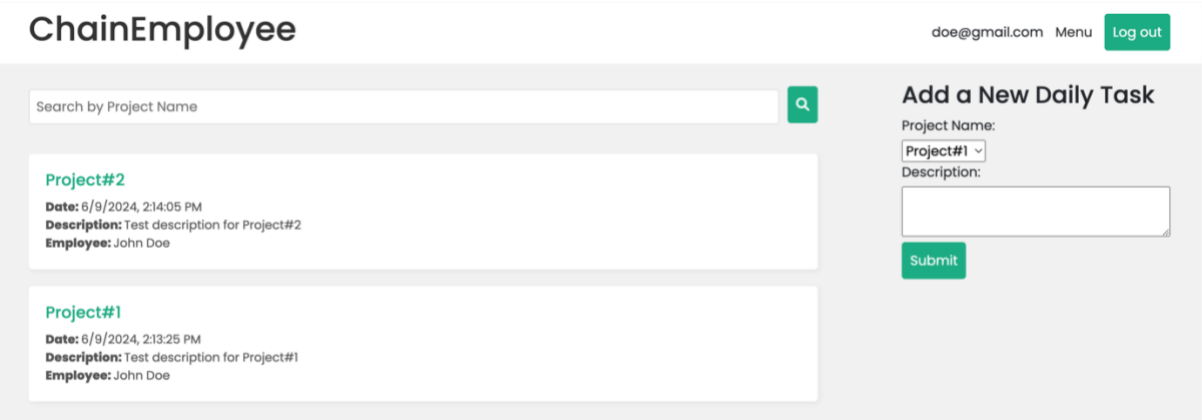


Slika 5.2. *Prikaz stranice za registraciju*

5.2. Korisnički dio

5.2.1. Početni prikaz stranice

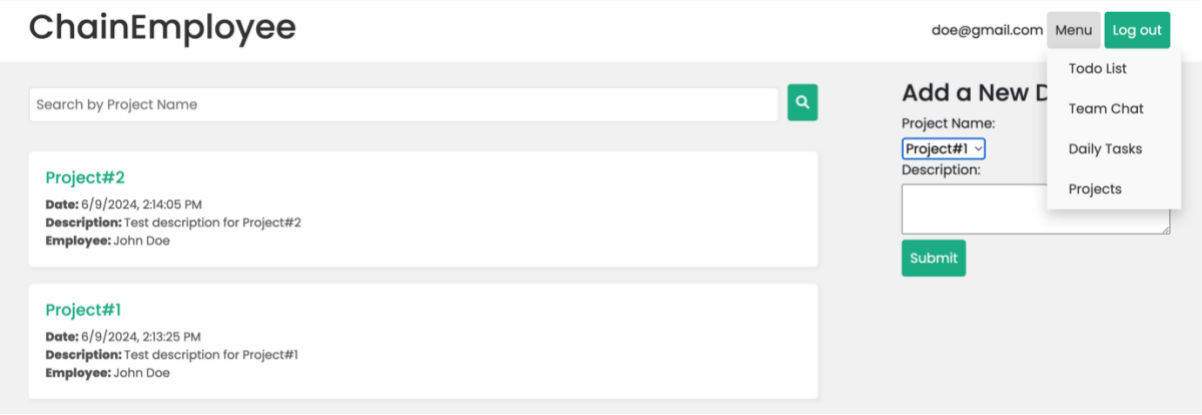
Pri dolasku na početnu stranicu, korisnik odmah vidi svoje dnevne zadatke za određeni projekt te može pregledati svoje prethodno unesene zadatke za isti projekt. Također, na desnoj strani mu je omogućen prostor za unos novih dnevnih zadataka za taj projekt prikazano na slici 5.3.



Slika 5.3. *Prikaz početne stranice*

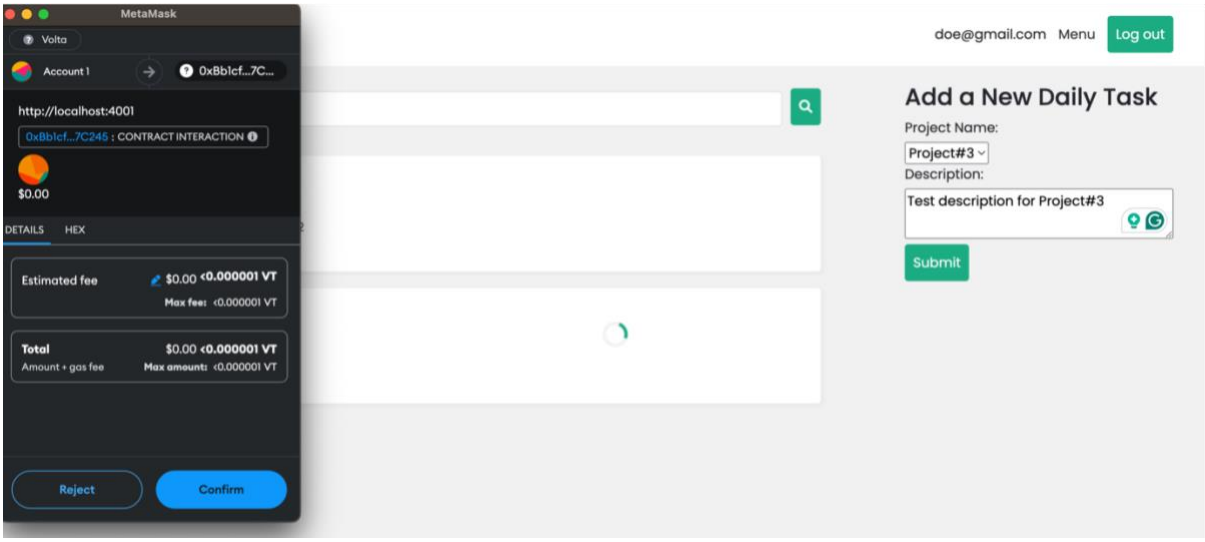
U navigacijskoj traci, korisniku je prikazano njegovo ime, a odmah pored toga se nalazi tipka "Menu"(Slika 5.4). Klikom na ovu tipku otvara se padajući izbornik s drugim opcijama koje nudi

aplikacija, kao što su linkovi na druge dijelove aplikacije. Na samom kraju navigacijske trake nalazi se tipka "Logout" koja korisniku omogućuje izlazak iz aplikacije.



Slika 5.4. *Prikaz padajućeg izbornika*

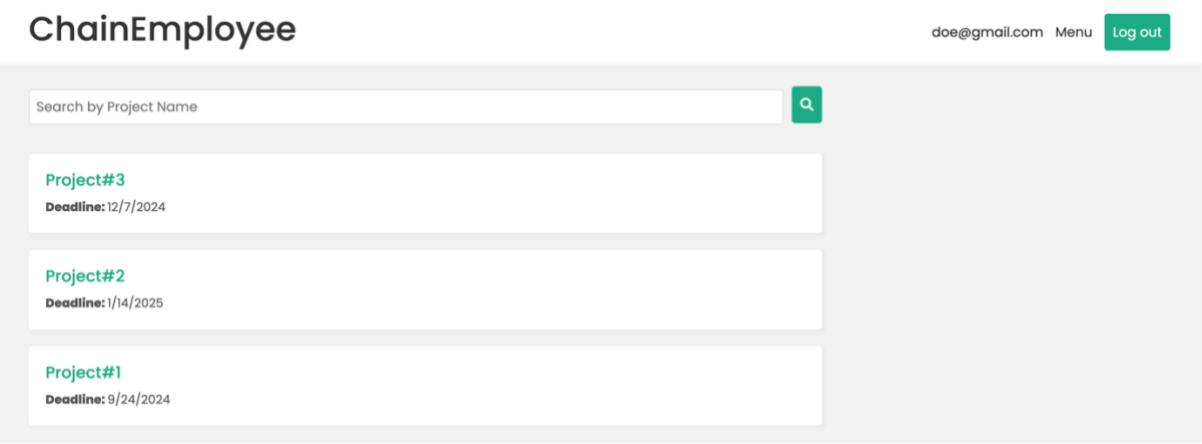
Korisnik može unijeti dnevne zadatke za odabrani projekt tako što prvo u padajućem izborniku odabere trenutno aktivan projekt. Nakon toga, unosi opis dnevnog zadatka u predviđeni prostor te klikne na gumb "Submit". Klikom na ovaj gumb, korisniku se otvara Metamask novčanik radi potvrde transakcije. Nakon potvrde klikom na tipku "Confirm" u Metamask novčaniku, izvršava se plaćanje određene transakcijske naknade, a zatim se podaci o dnevnom zadatku zapisuju na blockchain. Slika 5.5. prikazuje proces dodavanja dnevnog zadatka.



Slika 5.5. *Prikaz dodavanja dnevnog zadatka*

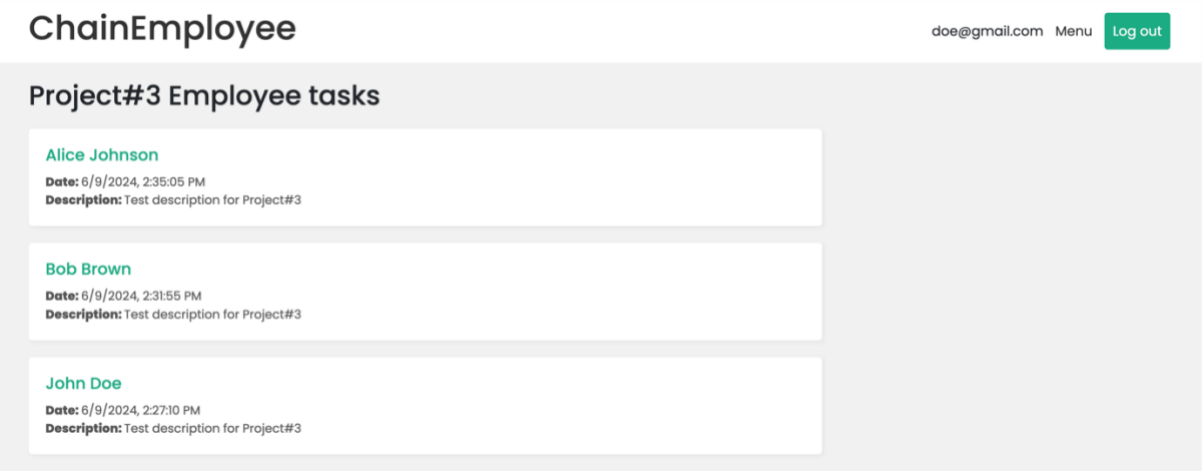
5.2.2. Prikaz trenutnih projekata

Kada korisnik odabere opciju "Projects" iz padajućeg menija, vodi ga se na stranicu trenutnih aktivnih projekata, prikazanu na slici 5.6. Korisniku je omogućena opcija pretraživanja određenog projekta unosom imena projekta u predviđeno polje za pretraživanje. Nakon unosa imena projekta, korisnik može kliknuti na gumb s ikonom povećala kako bi započeo pretraživanje.



Slika 5.6. *Prikaz dostupnih projekata*

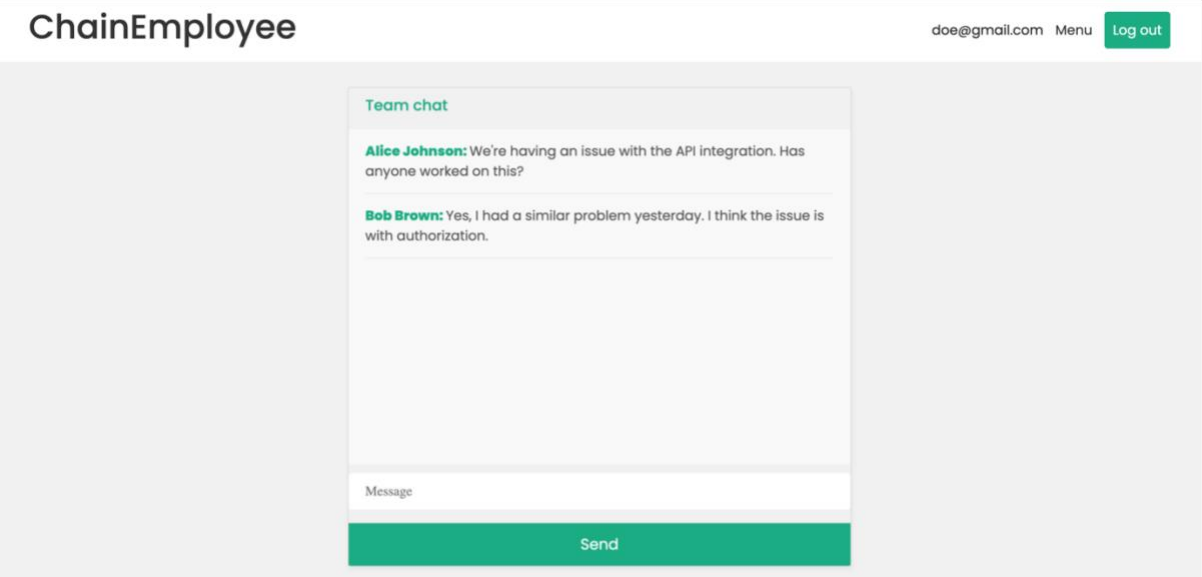
Klikom na određeni projekt, korisnika se vodi na stranicu gdje su ispisani svi dnevni zadaci od svakog zaposlenika za taj projekt prikazano na slici 5.7. Na taj način korisnik može pratiti trenutno stanje projekta i aktivnosti svojih kolega.



Slika 5.7. *Prikaz dnevnih zadataka zaposlenika za odabrani projekt*

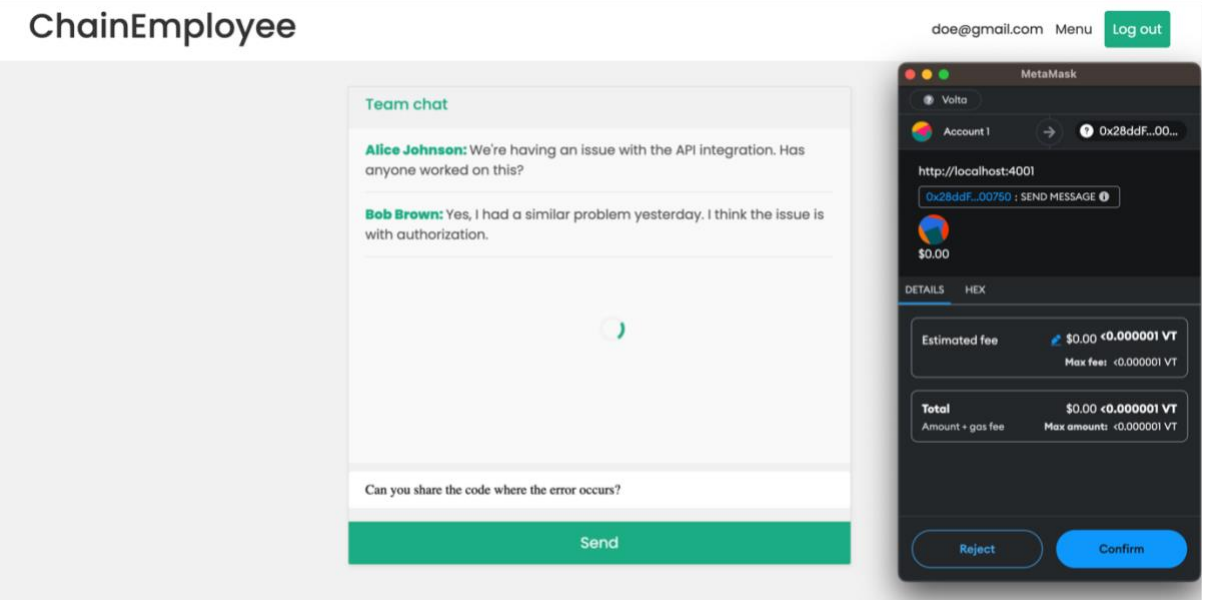
5.2.3. Korištenje timskog chata

Kada korisnik odabere opciju "Team Chat" iz padajućeg izbornika na navigacijskoj traci, otvara se stranica s timskim chatom. Na toj stranici, korisnik može pregledati poruke zaposlenika i ima opciju za unos nove poruke, što je prikazano na slici 5.8.



Slika 5.8. *Prikaz chata aplikacije*

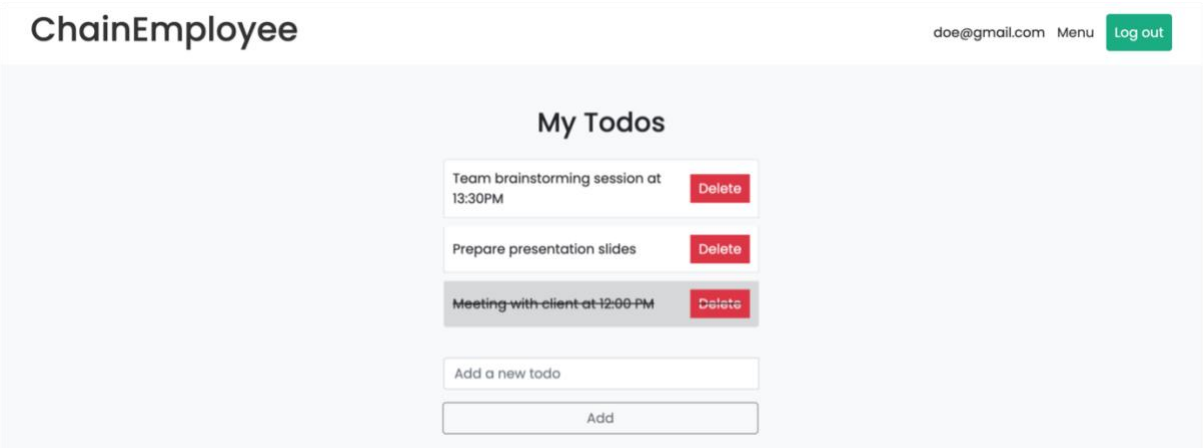
Kada korisnik unese poruku u predviđeno polje, može je poslati klikom na gumb "Send". Klikom na ovaj gumb, otvara se Metamask novčanik, a korisnik može potvrditi transakciju klikom na tipku "Confirm" (Slika 5.9). Nakon što se plati naknada, poruka se sprema na blockchain.



Slika 5.9. *Prikaz zapisivanja poruke na blockchain*

5.2.4. Korištenje Todo liste

Kada korisnik odabere opciju "Todo List" iz padajućeg izbornika, bit će preusmjeren na stranicu za vođenje osobne todo liste. Na ovoj stranici, korisnik može dodavati, uređivati i brisati svoje osobne zadatke, koji su vidljivi isključivo njemu. Osim toga, korisnik može postavljati podsjetnike za važne zadatke kako bi lakše upravljao svojim obvezama. Sve ove funkcionalnosti omogućavaju korisniku da učinkovito organizira svoj radni dan i ostane fokusiran na prioritete. Prikaz osobne todo liste može se vidjeti na slici 5.10.

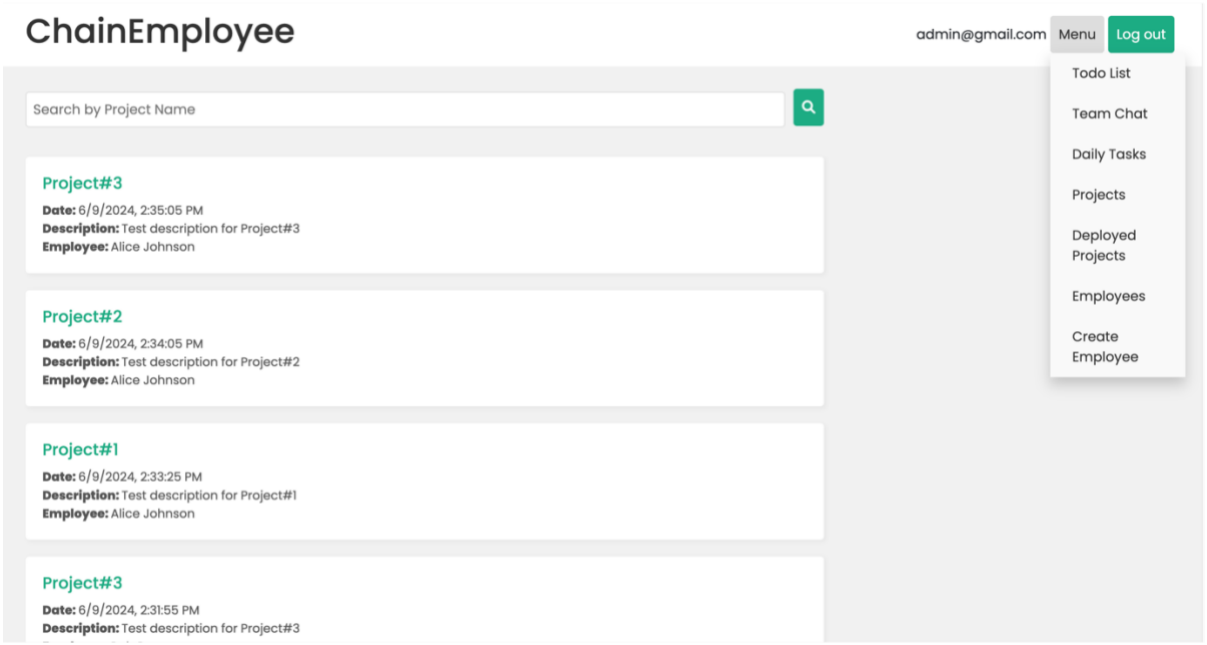


Slika 5.10. *Prikaz Todo liste*

5.3. Administratorski dio

5.3.1. Početni prikaz stranice

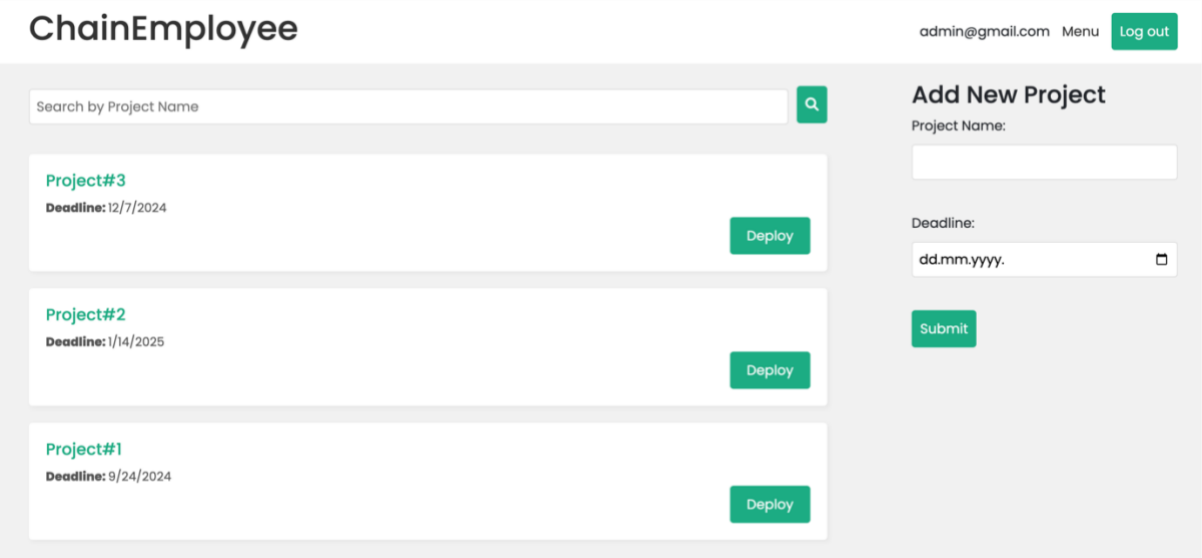
Dolaskom na početnu stranicu administrator ima popis svih dnevnih zadataka od svih zaposlenika te tako ima uvid tko je što radio taj dan. Na stranici ima opciju pretrage projekta po imenu. Isto tako u navigacijskoj traci ima padajući meni sa svim opcijama koje ima običan korisnik koje su opisane u prethodnim poglavljima te isto tako ima dodatne opcije koje su vidljive samo administratoru te koje će biti opisane u narednim poglavljima. Slika 5.11 prikazuje početnu stranicu koju vidi administrator.



Slika 5.11. *Prikaz početne stranice koju vidi administrator*

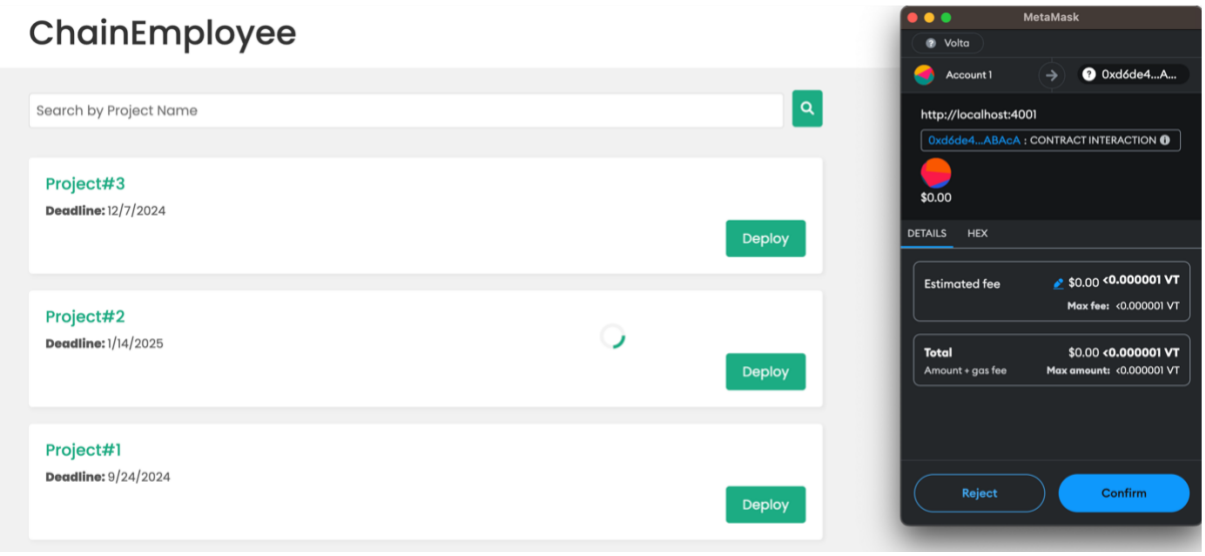
5.3.2. Stranica za dodavanje i postavljanje projekata na blockchain

Odabirom opcije "Projects" u padajućem meniju, administrator se preusmjerava na stranicu s popisom svih projekata. Klikom na određeni projekt, administratoru se prikazuju svi dnevni zadaci koje su kreirali zaposlenici. Na desnoj strani stranice nalazi se obrasac za dodavanje novih projekata. Ako administrator želi unijeti novi projekt, mora unijeti ime projekta i rok do kojeg taj projekt treba biti isporučen. Klikom na gumb "Submit", korisnik kreira novi projekt. Slika 5.12 prikazuje izgled stranice za dodavanje i postavljanje projekata na blockchain.



Slika 5.12. *Prikaz stranice za dodavanje novih projekata*

Kada se određeni projekt završi, administrator može klikom na gumb "Deploy" postaviti projekt i sve njegove zapise od svakog zaposlenika na blockchain. Na taj način, administrator stvara trajni zapis koji se može pregledati u bilo kojem trenutku, osiguravajući da svi podaci ostanu dostupni i nepromjenjivi. Kada administrator klikne na gumb "Deploy", otvara se Metamask novčanik te se plaća određena naknada za zapis na blockchain. Klikom na gumb "Confirm" u novčaniku, podaci se spremaju na blockchain. Slika 5.13 prikazuje proces zapisivanja podataka na blockchain.



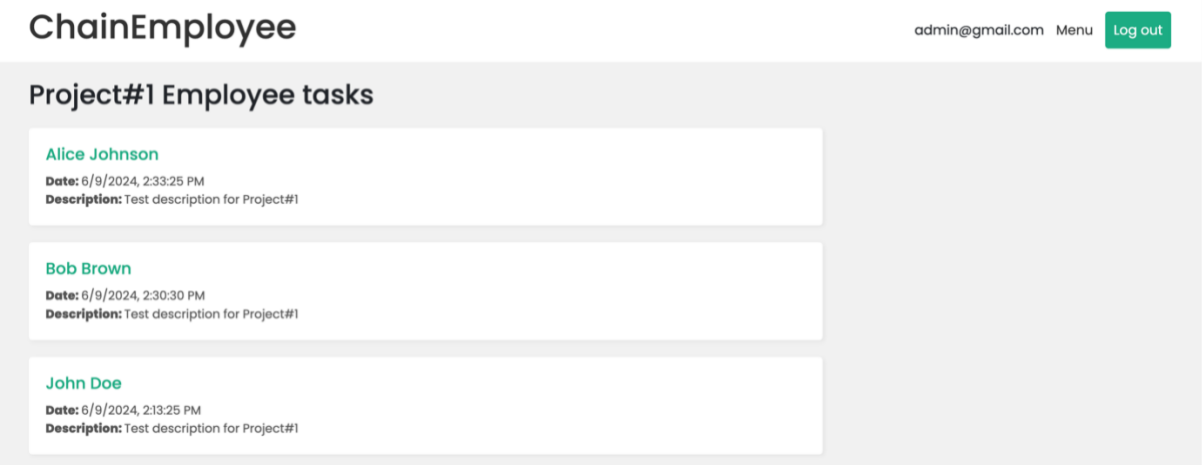
Slika 5.13. *Prikaz zapisivanja projekta na blockchain*

Kada se transakcija uspješno izvrši, administrator se preusmjerava na stranicu s postavljenim projektima na blockchainu. Na toj stranici administrator može vidjeti sve projekte koji su

postavljeni na blockchain i pretraživati projekte po imenu. Slika 5.14 prikazuje stranicu s projektima postavljenim na blockchain.

Slika 5.14. *Prikaz stranice s projektima postavljenim na blockchain*

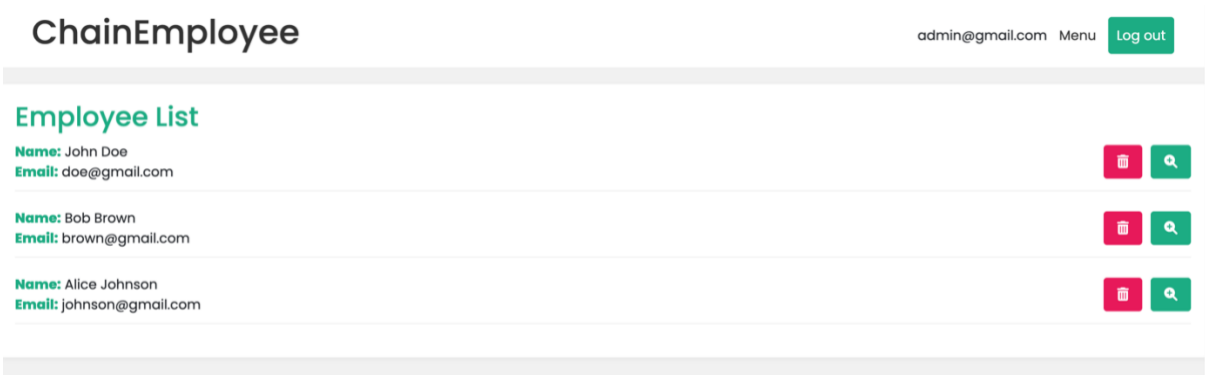
Klikom na odabrani projekt, administratoru se prikazuju svi zapisi, odnosno dnevni zadaci svakog zaposlenika za odabrani projekt. Na taj način, administrator se može vraćati u prošlost i pregledavati zapise. Slika 5.15 prikazuje stranicu na kojoj su prikazani svi dnevni zadaci svakog zaposlenika za odabrani projekt.



Slika 5.15. *Prikaz svih dnevnih zadataka za odabrani projekt*

5.3.3. Stranica za upravljanje zaposlenicima

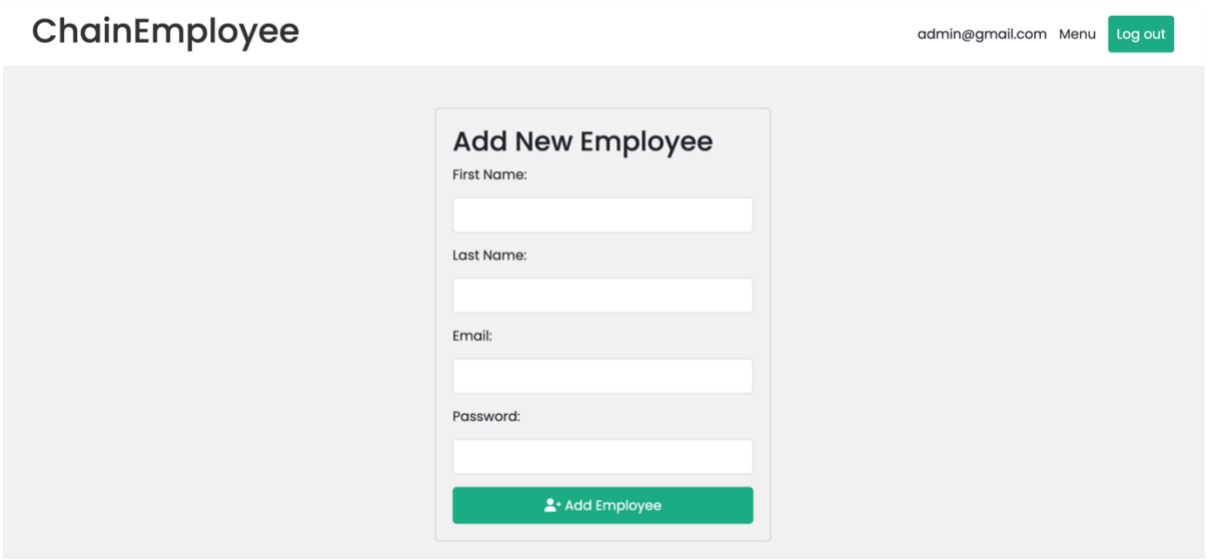
Pomoću opcije 'Employees' u padajućem izborniku, administrator se preusmjerava na stranicu s popisom svih zaposlenika. Na ovoj stranici, klikom na gumb s ikonom kante za smeće, administrator može obrisati odabranog zaposlenika. Također, klikom na gumb povećala, administratoru se prikazuju svi dnevni zadaci za tog odabranog zaposlenika. Slika 5.16 prikazuje stranicu za upravljanje zaposlenicima.



Slika 5.16. *Prikaz stranice za upravljanje zaposlenicima*

5.3.4. Stranica za dodavanje zaposlenika

Pomoću opcije 'Create Employee' u padajućem izborniku, administrator ima mogućnost ručno dodavanja novog zaposlenika. Administrator u obrasac unosi ime, prezime, e-mail i lozinku novog zaposlenika. Nakon unosa potrebnih podataka, klikom na gumb 'Add Employee', stvara se novi zaposlenik u sustavu. Slika 5.17 prikazuje stranicu za kreiranje novog zaposlenika.



Slika 5.17. *Prikaz stranice za kreiranje novih zaposlenika*

6. ZAKLJUČAK

U ovom završnom radu analizirana je i razvijena web3 aplikacija za vođenje projekata i praćenje dnevnih zadataka unutar tvrtke, s naglaskom na sigurnost i transparentnost putem blockchain tehnologije. Kroz ovaj rad istražene su prednosti i nedostaci primjene web3 tehnologija u poslovnim aplikacijama, kao i praktične implementacije koje omogućuju bolju sljedivost i odgovornost zaposlenika u radu na projektima.

Glavna prednost ove web3 aplikacije jest sigurnost i nepromjenjivost zapisanih podataka. Korištenje blockchain tehnologije omogućuje trajno pohranjivanje dnevnih zadataka i poruka između zaposlenika, čime se osigurava visoka razina povjerenja i transparentnosti. Svi zapisi postaju nepromjenjivi i dostupni za pregled u svakom trenutku, što pomaže u praćenju odgovornosti i povećava vjerodostojnost unesenih podataka.

Međutim, primjena web3 tehnologije dolazi s određenim izazovima. Jedan od ključnih nedostataka je sporost u zapisivanju podataka na blockchain zbog vremena potrebnog za verifikaciju i potvrdu transakcija. Ovaj problem može utjecati na korisničko iskustvo, posebno kada je potreban brzi unos i obrada podataka. Nadalje, neizbježni transakcijski troškovi (gas fees) predstavljaju još jedan izazov. Iako je Volta testna mreža omogućila besplatno testiranje funkcionalnosti, stvarna primjena u produkcijskom okruženju zahtijevala bi plaćanje naknada, što bi moglo opteretiti zaposlenike ili tvrtku. U budućnosti, ovo bi moglo postati problem ukoliko bi svaki zaposlenik morao plaćati određeni trošak kako bi zapisao podatke na blockchain, što može dovesti do nepotrebnih financijskih prepreka.

Kako bi se ovi nedostaci ublažili, moguće je razmotriti hibridne pristupe, gdje se manje kritični podaci pohranjuju u tradicionalne baze podataka, a samo ključni zapisi koji zahtijevaju maksimalnu sigurnost i transparentnost postavljaju na blockchain.

Zaključno, seminar je omogućio detaljan uvid u potencijal blockchain tehnologije i web3 aplikacija u poslovnom okruženju. Iako web3 tehnologija pruža iznimne prednosti u smislu sigurnosti i transparentnosti, njena puna implementacija dolazi s određenim izazovima, uključujući sporost i financijske troškove. Ova aplikacija predstavlja primjer kako bi se web3 tehnologija mogla koristiti u praćenju projekata i zadataka unutar tvrtke, ali i otvara prostor za daljnja istraživanja i poboljšanja u ovom području.

Web3 tehnologija, iako u ranoj fazi usvajanja, ima ogroman potencijal u raznim industrijama, a ovaj rad prikazuje samo jedan od mogućih načina njene primjene. U budućnosti, s napretkom tehnologije i razvojem inovativnih rješenja, moguće je očekivati da će ovakve aplikacije postati standard u osiguravanju sigurnosti i transparentnosti poslovnih podataka.

LITERATURA

[1] „Asana“ [mrežno]. Dostupno na: https://asana.com/ [pristupljeno 24. svibnja 2024.]

[2] “Trello” [mrežno]. Dostupno na: https://trello.com/ [pristupljeno 24. svibnja 2024.]

[3] “Jira” [mrežno]. Dostupno na: https://www.atlassian.com/software/jira [pristupljeno 24. svibnja 2024.]

[4] “HTML.com” [mrežno]. Dostupno na: https://html.com/ [pristupljeno 20. kolovoza 2024.]

[5] “W3Schools” [mrežno]. Dostupno na: https://www.w3schools.com/ [pristupljeno 20. kolovoza 2024.]

[6] “Ethereum.org” [mrežno]. Dostupno na: https://ethereum.org/en/ [pristupljeno 30. svibnja 2024.]

[7] “MDN web docs” [mrežno]. Dostupno na: https://developer.mozilla.org/en-US/ [pristupljeno 30. svibnja 2024.]

[8] “Node.js” [mrežno]. Dostupno na: https://nodejs.org/en [pristupljeno 30. svibnja 2024.]

[9] “Express.js” [mrežno]. Dostupno na: https://expressjs.com/ [pristupljeno 30. svibnja 2024.]

[10] “EJS - Embedded JavaScript templates” [mrežno]. Dostupno na: https://ejs.co/ [pristupljeno 30. svibnja 2024.]

[11] “Ethers.js” [mrežno]. Dostupno na: https://docs.ethers.org/v5/ [pristupljeno 30. svibnja 2024]

[12] ”MongoDB” [mrežno]. Dostupno na: https://www.mongodb.com/ [pristupljeno 30. svibnja 2024]

[13] “Solidity Programming Language” [mrežno]. Dostupno na: https://soliditylang.org/ [pristupljeno 30. svibnja 2024.]

[14] “Hardhat” [mrežno]. Dostupno na: https://hardhat.org/ [pristupljeno 31. svibnja 2024]

[15] “Volta Testnet” [mrežno]. Dostupno na: https://volta-explorer.energyweb.org/ [pristupljeno 31. svibnja 2024.]

[16] “MetaMask” [mrežno]. Dostupno na: https://metamask.io/ [pristupljeno 31. svibnja 2024.]

SAŽETAK

Završni rad opisuje izradu web3 aplikacije za vođenje projekata unutar tvrtke koja pruža sigurno pohranjivanje dnevnih zadataka i poruka zaposlenika na blockchainu. Rad istražuje primjenu blockchain tehnologije i njene prednosti, kao što su sljedivost i transparentnost podataka. Korištenjem Node.js platforme, Express.js okvira te Solidity pametnih ugovora, aplikacija omogućuje administraciju projekata i praćenje zadataka na Volta testnoj mreži. Opisane su ključne funkcionalnosti aplikacije, uključujući mogućnost postavljanja projekata na blockchain te pohranjivanje komunikacije između zaposlenika pomoću Metamask sučelja. Rad također ističe sigurnosne aspekte pohranjivanja podataka na blockchain te potencijalne nedostatke kao što su sporost zapisivanja i troškovi transakcija. Kroz rad je demonstrirana implementacija web3 tehnologija te prikazana njihova primjena u stvarnom poslovnom okruženju.

Ključne riječi: Blockchain, Express.js, Metamask, Node.js, Solidity, Web3

ABSTRACT

Web3 application for project managment

The final thesis describes the development of a web3 application for managing projects within a company, which provides secure storage of daily tasks and employee messages on the blockchain. The paper explores the application of blockchain technology and its advantages, such as traceability and data transparency. By utilizing the Node.js platform, the Express.js framework, and Solidity smart contracts, the application enables project administration and task tracking on the Volta test network. The key functionalities of the application are described, including the ability to deploy projects on the blockchain and store employee communication using the Metamask interface. The thesis also highlights the security aspects of storing data on the blockchain and potential drawbacks such as the slow speed of data entry and transaction costs. The work demonstrates the implementation of web3 technologies and showcases their application in a real business environment.

Keywords: Blockchain, Express.js, Metamask, Node.js, Solidity, Web3

PRILOZI

P.1. Izvorni kod aplikacije

Dostupno na: https://github.com/KarloBuhinjak/ChainEmployee