Sustav za praćenje rada na projektu

Zoković, Lovro

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1020

SUSTAV ZA PRAĆENJE RADA NA PROJEKTU

Lovro Zoković

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1020

SUSTAV ZA PRAĆENJE RADA NA PROJEKTU

Lovro Zoković

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

ZAVRŠNI ZADATAK br. 1020

Pristupnik:	**Lovro Zoković (0036515497)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Vedran Mornar

Zadatak: **Sustav za praćenje rada na projektu**

Opis zadatka:

Izraditi sustav za praćenje rada na projektu. Korisnici sustava prijavljuju se kao radnik, voditelj i administrator sustava. Voditelj ima mogućnosti definiranja projekata, liste zadataka za svakog radnika i praćenje statusa izvršavanja zadataka. Također, voditelj može dodjeljivati zaposlenike projektu, pregledavati satnicu za svakog radnika te odobravati, korigirati ili odbijati mjesečni obračun za svakog radnika. Radnik dnevno unosi svoju satnicu s opisom obavljenih poslova za dodijeljeni zadatak. Voditelj i radnik unose i napomene vezane za poslove koje treba obaviti. Administrator sustava unosi korisnike sustava, uređuje postojeće korisnike te upravlja ulogama. Koristiti razvojno okruženje .NET i bazu podataka SQL Server.

Rok za predaju rada: 9. lipnja 2023.

Uvod

U suvremenom poslovnom okruženju, voditeljima projekata je u interesu pratiti rad djelatnika na projektu kako bi postigli bolju produktivnosti i uspješno upravljali svojim resursima.

Još uvijek je učestala pojava da voditelj projekta djelatnicima zada zadatke usmeno, mailom ili nekom aplikacijom za razmjenu poruka te tako i dobije povratnu informaciju od svojih djelatnika. Voditelj često ne zna što mu točno djelatnici rade i koliko su vremena potrošili na zadatke. Postoje alati koji omogućavaju vođenje projekata ali su većinom skupi i ne sadrže sve specifične potrebne funkcionalnosti.

Ovaj završni rad omogućuje detaljan prikaz radnih navika i učinkovitosti tima na projektu. Napravljena je Web aplikacija koja omogućava voditelju da kreira projekt, definira njemu pripadne zadatke, dodaje djelatnike na projekt, te im dodijeli zadatke s projekta. Djelatnici, umjesto da radnu evidenciju vode preko Excela, maila ili poruke, putem aplikacije odabiru na kojem zadatku su radili, te unose broj sati potrošenih na istom. Osim toga mogu ostaviti i komentar za voditelja i pogledati njegovu povratnu informaciju. Voditelj projekta pomoću aplikacije može za svakog djelatnika pratiti kako napreduje te s njim razmjenjivati poruke vezane za izvršavanje zadatka. Tako voditelj može bolje raspolagati resursima i brže predvidjeti potencijalne probleme. Na kraju mjeseca voditelj kontrolira unesene sate od strane djelatnika, mijenja ih po potrebi i zaključava ih, čime onemogućava daljnje ažuriranje podataka od strane djelatnika.

Aplikacija trenutno ne raspolaže s puno funkcionalnosti i postoji mnogo smjerova u kojima se može razvijati. Njena je prednost što na jednom mjestu čuva podatke do kojih je u bilo kojem trenutku jednostavno doći što je voditeljima projekata bitno prilikom izrade izvještaja i općenito praćenja rada na projektu.

1. Funkcionalni zahtjevi sustava

Funkcionalni zahtjevi sustava opisuju očekivano ponašanje sustava, tj. što korisnici mogu očekivati od sustava i kako će se sustav ponašati u različitim situacijama. Ovise o ulozi prijavljenog korisnika. Ova aplikacija je specifična po tome što inicijalno ima unesenog jednog korisnika s administrativnim dozvolama i on unosi sve ostale korisnike. Korisnici sustava se ne mogu sami registrirati jer moraju biti samo iz skupa osoba koje su unaprijed poznate po svojoj ulozi na nekom postojećem ili budućem projektu.

Svaki korisnik sustava inicijalno ima ulogu Djelatnik, a može još imati i Administrator i/ili Voditelj projekta. Korisnik može istovremeno imati i više uloga tako da npr. može istovremeno biti Djelatnik i Administrator ili Voditelj projekta i Administrator. Svaka uloga nosi različite funkcionalnosti sustava. Bitno je naglasiti da uloge Administrator sustava i Voditelja sustava uključuju i sve ovlasti uloge Djelatnik. U nastavku će funkcionalni zahtjevi biti navedeni po ulogama

1.1. Uloga Administrator

Korisnik s ovom ulogom ima mogućnost:

1. Kreiranje novih korisnika i pridruživanje uloga.
2. Ažuriranje podataka o postojećim korisnicima.
3. Unos i održavanje podataka o statusima zadataka.

1.2. Uloga Voditelj projekta

Korisnik s ovom ulogom ima mogućnost:

1. Kreiranje projekata i održavanje podataka o istom. Unosom projekta, korisnik automatski postaje djelatnik na projektu s administratorskim dozvolama na tom projektu. Administrator na projektu može dodavati zadatke na projekt, korisnike na zadatke, čitati sva dopisivanja na projektu ali ne može kontrolirati sate djelatnika niti unositi nove projekte (osim ako ima postavljenu ulogu Voditelj projekta).
2. Pridruživanje djelatnika projektima. Administrator projekta ima mogućnost da i drugim djelatnicima da ovlasti administrator na projektu.

3. Kreiranje i održavanje podataka o zadatcima na projektu.
4. Pridruživanje djelatnika s projekta na definirane zadatke.
5. Pregled i ažuriranje unesenih aktivnosti i sati djelatnika na projektu
6. Zaključavanje podataka o unesenim satima za zadani projekt, djelatnika i mjesec. Nakon što je mjesec zaključan djelatnik ne može mijenjati dodavati ili mijenjati unesene podatke.

1.3. Uloga Djelatnik

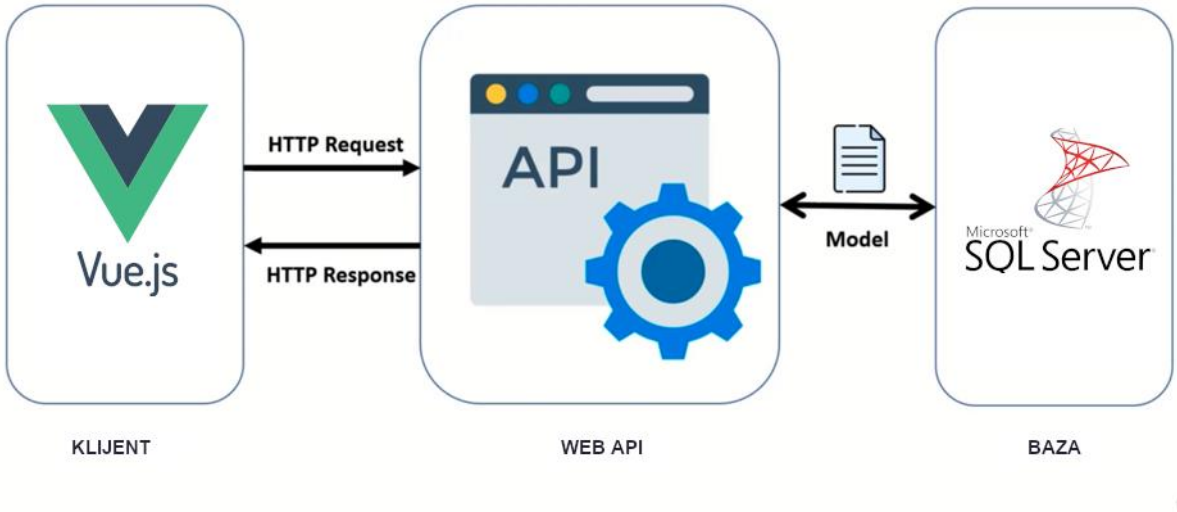
Korisnik s ovom ulogom ima mogućnost:

1. Pregled podataka o zadatcima na koje je djelatnik pridružen.
2. Slanje poruka drugim djelatnicima na projektu.
3. Brzi dohvat pristiglih nepročitanih poruka.
4. Unos aktivnosti i sati za svaki dan po zadatku na projektu.
5. Prijava u sustav.
6. Odjava iz sustava.
7. Promjena lozinke.

2. Arhitektura i dizajn sustava

Arhitektura sustava (Slika 2.1) temeljena je na arhitekturi klijent-poslužitelj gdje je klijent implementiran kao Web aplikacija, a poslužitelj je oblikovan REST arhitekturnim stilom.

Dodatno proširenje čini još podatkovna razina (baza) koja služi za trajnu pohranu podataka.



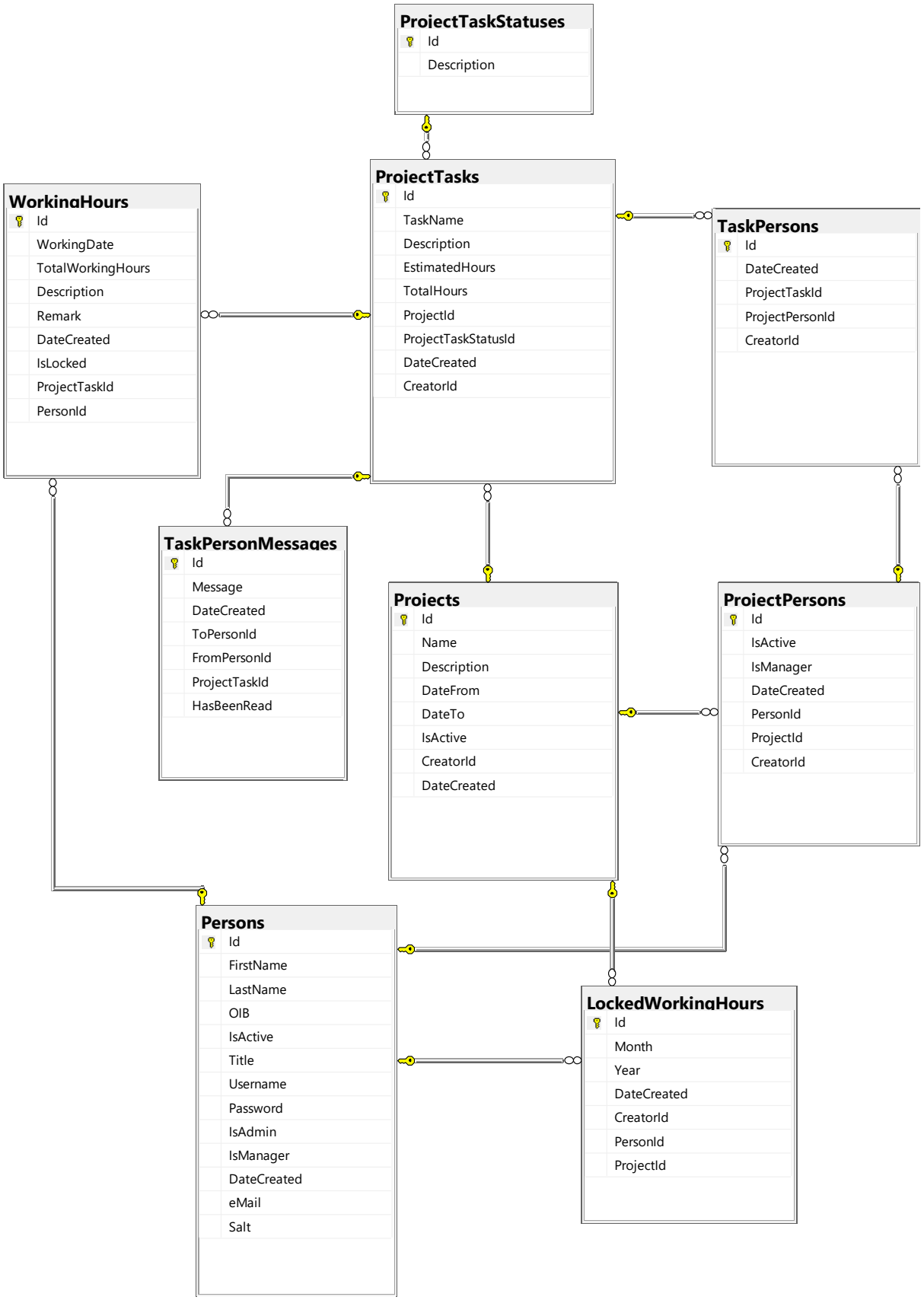
Slika 2.1 - Arhitektura sustava

2.1. Podatkovna razina

Podatkovnu razinu čini baza podataka koja služi za trajno pohranjivanje podataka, a u ovom radu korišten je Microsoft SQL Server Express.

U nastavku su pobrojane sve relacije koje se koriste u ovom sustavu kao i njihova kratak opis i svrha te popis atributa. Primarni ključ svake relacije je atribut s imenom Id koji je u bazi definiran kao samo inkrementirajući (eng. identity).

Slika 2.2 prikazuje relacijski model baze podataka.



Slika 2.2 - Relacijski model baze podataka

2.1.1. Person

Relacija Person sadrži podatke o korisnicima sustava. Korisnik se u bazu ne može sam dodati, već ga dodaje korisnik s administratorskim ulogom. Korisnik u sustavu može imati ulogu administratora (atribut IsAdmin), voditelja projekta (atribut IsManager) i ako nije ništa od toga ima samo ulogu Djelatnik. U tablici se pri samom početku rada sustava mora nalaziti jedan korisnik s administratorskom ulogom zato da bi mogao dodavati druge korisnike. Bitno je naglasiti da se vrijednost za lozinku korisnika čuva u bazi kriptirano, a da bi sustav mogao autorizirati korisnika spremljen je i *salt* koji se koristi prilikom kriptiranja lozinke tj. u trenutku pohranjivanja podatka u bazu. Algoritam koji je korišten za kriptiranje lozinke je SHA256. Tako lozinka korisnika nije čitljiva ni administratoru baze.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa
Id	int	ne	Jedinstveni identifikator zapisa u bazi
FirstName	varchar(100)	ne	Ime korisnika
LastName	varchar(100)	ne	Prezime korisnika
OIB	varchar(11)	da	Osobni identifikacijski broj korisnika
IsActive	bit	ne	0 – korisnik je neaktivan 1 – korisnik je aktivan
Title	varchar(100)	da	Zvanje korisnika
Username	varchar(50)	ne	Korisničko ime koje se koristi prilikom prijave u aplikaciju
Password	varchar(500)	ne	Kriptirana lozinka korisnika
IsAdmin	bit	ne	0 – korisnik nema ulogu Administrator 1 – korisnik ima ulogu Administrator
IsManager	bit	ne	0 – korisnik nema ulogu Voditelj projekata 1 – korisnik ima ulogu Voditelj projekata
DateCreated	datetime2(7)	ne	Datum unosa podatka u tablicu
eMail	varchar(50)	ne	eMail korisnika
salt	varchar(500)	ne	Nasumično generirani dodatak (*salt*) s kojim se kriptira lozinka korisnika

Tablica 2.1 - Popis atributa relacije Person

2.1.2. Projects

Relacija Projects sadrži podatke o projektima. Podatke o projektu može unijeti samo korisnik s dozvolom Voditelj projekta. Uz opisane atribute, projekt se sastoji od svojih zadataka koji su opisani u relaciji ProjectTasks i korisnika koji rade na tom projektu koji su definirani u relaciji ProjectPersons.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa
Id	int	ne	Jedinstveni identifikator zapisa u relaciji
Name	varchar(200)	ne	Naziv projekta
Description	varchar(1000)	da	Opis projekta
DateFrom	datetime2(7)	da	Datum od kojeg kreće izrada projekta
DateTo	datetime2(7)	da	Datum završetka izrade projekta
IsActive	bit	ne	0 – projekt je završio ili nije još počeo 1 – projekt je u procesu izrade
CreatorId	int	ne	Jedinstveni identifikator korisnika koji je kreirao projekt
DateCreated	datetime2(7)	ne	Datum unosa podatka u tablicu

Tablica 2.2 - Popis atributa relacije Projects

2.1.3. ProjectPersons

Relacija ProjectPersons služi kao veza između relacija Project i Person i s njom se opisuje pripadnost korisnika projektu, odnosno da je korisnik član na projektu.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa
Id	int	ne	Jedinstveni identifikator zapisa u relaciji
IsActive	bit	ne	0 - korisnik nije aktivan na projektu 1 – korisnik je aktivan na projektu
IsManager	bit	ne	0 – korisnik nema ulogu administratora na projektu 1 – korisnik ima ulogu administratora na projektu
DateCreated	datetime2(7)	ne	Datum unosa podatka u tablicu
PersonId	int	ne	Jedinstveni identifikator korisnika
ProjectId	int	ne	Jedinstveni identifikator projekta

CreatorId	int	ne	Jedinstveni identifikator korisnika koji je kreirao zapis

Tablica 2.3 - Popis atributa relacije ProjectPersons

2.1.4. ProjectTask

Relacija ProjectTasks sadrži opise zadataka koje neki projekt sadržava. Zadatke može unijeti bilo koji korisnik koji za taj projekt u relaciji ProjectPerson ima postavljeno da je administrator projekta (IsManager postavljeno na vrijednost True). Uz opis zadatka potrebno je unijeti i procjenu koliko sati će trebati za izvršenje tog zadatka. Također se uz zadatak stavlja i status kako bi lakše pratio izvršavanje zadataka.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa
Id	int	ne	Jedinstveni identifikator zapisa u relaciji
TaskName	varchar(100)	ne	Naziv zadatka
Description	varchar(500)	ne	Opis zadatka
EstematedHours	int	ne	Procijenjen broj sati koliko treba za izvršavanje zadatka
TotalHours	int	da	Broj sati koliko se stvarno utrošilo na zadatku. Unosi ga voditelj projekta nakon što je zadatak obavljen. Nema ugrađene automatike.
DateCreated	datetime2(7)	ne	Datum unosa podatka u tablicu
ProjectId	int	ne	Jedinstveni identifikator projekta za koji je zadatak definiran
ProjectTaskStatusId	int	ne	Jedinstveni identifikator statusa zadatka
CreatorId	int	ne	Jedinstveni identifikator korisnika koji je napravio zapis

Tablica 2.4 - Popis atributa relacije ProjectTask

2.1.5. TaskPersons

Relacija TaskPersons služi kao veza između relacija ProjectTasks i Persons i s njom se opisuje pripadnost korisnika zadatku unutar projekta. Korisniku je moguće dodijeliti zadatak samo na projektu na kojem je unesen kao član projekta.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa

Id	int	ne	Jedinstveni identifikator zapisa u relaciji
ProjectTaskId	bit	ne	Jedinstveni identifikator zadatka iz tablice ProjectTasks koji se pridružuje korisniku
ProjectPersonId	bit	ne	Jedinstveni identifikator za tog korisnika u relaciji ProjectPersons
DateCreated	datetime2(7)	ne	Datum unosa podatka u tablicu
CreatorId	int	ne	Jedinstveni identifikator korisnika koji je kreirao zapis

Tablica 2.5 - Popis atributa relacije TaskPersons

2.1.6. TaskPersonMessages

Relacija TaskPersonMessages sadrži podatke o porukama koje šalju korisnici. Unutar aplikacije korisnik može poslati poruku samo unutar projektnog tima. Ideja je da se šalju poruke koje imaju veze s radom na zadatku kako bi u bazi ostale sačuvane. Administrator projekta i član na projektu mogu kasnije poruke pregledavati i filtrirati kroz aplikaciju.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa
Id	int	ne	Jedinstveni identifikator zapisa u relaciji
Message	varchar(500)	ne	Tekst poruke
DateCreated	datetime2(7)	ne	Datum unosa podatka u tablicu
ToPersonId	int	ne	Jedinstveni identifikator korisnika kome se poruka šalje
FromPersonId	int	ne	Jedinstveni identifikator korisnika koji je kreirao poruku
ProjectTaskId	int	ne	Jedinstveni identifikator zadatka
HasBeenRead	bit	ne	0 - poruka nije pročitana 1 - poruka je pročitana

Tablica 2.6 - Popis atributa relacije TaskPersonMessages

2.1.7. WorkingHours

Relacija WorkingHours opisuje jedan radni dan korisnika; na kojem je zadatku radio, koliko sati te kratak opis što je radio. Voditelj projekta ima mogućnost pregledavati sate za sve djelatnike na njegovom projektu, umanjiti ili uvećati broj zapisanih sati, napisati svoj

komentar te na kraju mjeseca zaključati te sate i s time onemogućiti daljnji unos ili promjenu od strane korisnika.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa
Id	int	ne	Jedinstveni identifikator zapisa u relaciji
WorkingDate	datetime2(7)	ne	Datum za koji se unose podaci o radu
TotalWorkingHours	float	ne	Broj sati rada
Description	varchar(500)	ne	Proizvoljni komentar korisnika
Remark	varchar(500)	ne	Proizvoljni komentar voditelja projekta na kojem je zadatak
DateCreated	datetime2(7)	ne	Datum unosa podatka u tablicu
IsLocked	bit	ne	0 – zapis nije zaključan i time je omogućena njegova promjena 1 – zapis je zaključan i time je onemogućena njegova promjena od strane korisnika koji ga je kreirao
ProjectTaskId	int	ne	Identifikator zadatka na kojem je korisnik radio
PersonId	int	ne	Identifikator korisnika koji unosi svoje sate

Tablica 2.7 - Popis atributa relacije WorkingHours

2.1.8. LockedWorkingHours

Relacija LockedWorkingHours pohranjuje podatak da je Voditelj projekta zaključao zapis za jedan mjesec za korisnika na projektu. Nakon što je mjesec zaključan korisnik ne može više dodavati ili mijenjati unesena podatke za taj mjesec. Akcija zaključavanja mjeseca automatski zaključa i sve dane u tom mjesecu. Ako voditelj otključa mjesec djelatniku se ponovo omogućava unos ili izmjena podataka. Voditelj može zaključati samo podatke za djelatnike koji su na njegovom projektu.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa
Id	int	ne	Jedinstveni identifikator zapisa u relaciji
Month	int	ne	Mjesec koji se zaključava
Year	int	ne	Godina koja se zaključava

DateCreated	datetime2(7)	ne	Datum kreiranja zapisa u tablici
CreatorId	int	ne	Jedinstveni identifikator korisnika koji je kreirao zapis
PersonId	varchar(500)	ne	Jedinstveni identifikator korisnika čiji se mjesec zaključava
ProjectId	datetime2(7)	ne	Jedinstveni identifikator projekta korisnika čiji se sati zaključavaju

Tablica 2.8 - Popis atributa relacije LockedWorkingHours

2.1.9. ProjectTaskStatuses

Relacija ProjectTaskStatuses sadrži opise statusa koji se mogu postaviti za zadatak. Opise može unijeti samo korisnik s ulogom Administrator.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa
Id	int	ne	Jedinstveni identifikator zapisa u relaciji
Description	varchar(200)	ne	Opis statusa

Tablica 2.9 - Popis atributa relacije ProjectTaskStatuses

2.1.10. __EFMigrationsHistory

U bazi se nalazi još i tablica koja nije prikazana na slici (Slika 2.2), a koju kreira EntityFrameworkCore biblioteka prilikom svake akcije ažuriranja relacijske sheme kroz programski kod Web servisa.

Naziv atributa	Tip podataka	Opcionalan	Opis atributa
MigrationId	int	ne	Jedinstveni identifikator zapisa u relaciji, sastoji se od vremena u kojem je zapis kreiran i naziva za ažuriranje relacijske sheme koju je programer naveo u konzoli za pokretanje migracije.
ProductVersion	varchar(100)	ne	Verzija EntityFrameworkCore biblioteke koja je korištena prilikom ažuriranja

Tablica 2.10 - Popis atributa relacije _EFMigrationsHistory

2.2. WEB API

Back-end sustava implementiran je korištenjem RESTfull web tehnologije.

REST (eng. Representational State Transfer) API je programsko sučelje, odnosno tip aplikacije koji omogućava komunikaciju ili interakciju između softverskih komponenti. REST API se često koristi za pružanje sučelja web stranicama i klijentskim aplikacijama kako bi im omogućio pristup podacima.

REST API implementira klase kontrolere koji sadrže metode preko kojih ide komunikacija s klijentskom aplikacijom. Implementirane metode većinom dohvaćaju podatke iz baze i šalju ih klijentu ili pohranjuju u bazu podatke koje je klijentska aplikacija poslala.

Karakteristike WEB API kao back-end sustava su sljedeće:

1. Protokoli i metode komunikacije:

   Web API koristi HTTP i HTTPS kao protokole za komunikaciju između klijenta i servera. Klijenti (npr. web preglednici, mobilne aplikacije) koriste HTTP metode za slanje zahtjeva prema serveru i dobivanje odgovora.

   Sljedeće HTTP metode se najčešće koriste u REST arhitekturi:

   GET - dohvat resursa

   PUT - ažuriranje postojećeg resursa

   DELETE - uklanjanje (brisanje) resursa

   POST - kreiranje novog resursa

2. Formatiranje podataka:

   Najčešći format za prijenos podataka u Web API je JSON (eng. JavaScript Object Notation) zbog svoje jednostavnosti i lakoće korištenja u različitim programskim jezicima. XML se također koristi, ali je manje popularan zbog svoje kompleksnosti.

3. Rutiranje poziva (eng. routing):

   Web API koristi rutiranje (usmjeravanje) kako bi mapirao URL-ove (eng. Uniform Resource Locators) na odgovarajuće metode. Na primer, URL "/persons/get-all-persons" je mapiran na GET metodu koja vraća sve korisnike, dok "/persons/get-all-persons/1" je mapiran na GET metodu koja vraća informacije o određenom korisniku.

4. Autentifikacija i autorizacija:

Web API često zahtijevaju autentifikaciju kako bi se osiguralo da samo korisnici s odgovarajućim dozvolama mogu pristup određenim metoda. Popularne metode autentifikacije uključuju API ključeve, OAuth i JSON Web Token (JWT).

5. Verzioniranje API-ja:

   Kako se Web API metode razvijaju i ažuriraju tijekom vremena, koristi se mehanizam verzioniranja. To omogućava klijentima da nastave koristiti stariju verziju API-ja dok se prilagođavaju novim promjenama.

6. Testiranje:

   Za testiranje API metoda postoje gotovi alati kako bi se osigurala funkcionalnost i ispravnost API-ja prije nego što se uvede u produkciju.

7. Skalabilnost i performanse:

   Web API se često koriste u aplikacijama s visokom opterećenosti i postoje različite tehnike kojima se osigurava skalabilnost i dobre performanse. Neke od tehnika su keširanje (eng. caching), optimizacija baze podataka, horizontalno skaliranje i upravljanje opterećenjem (eng. load balancing).

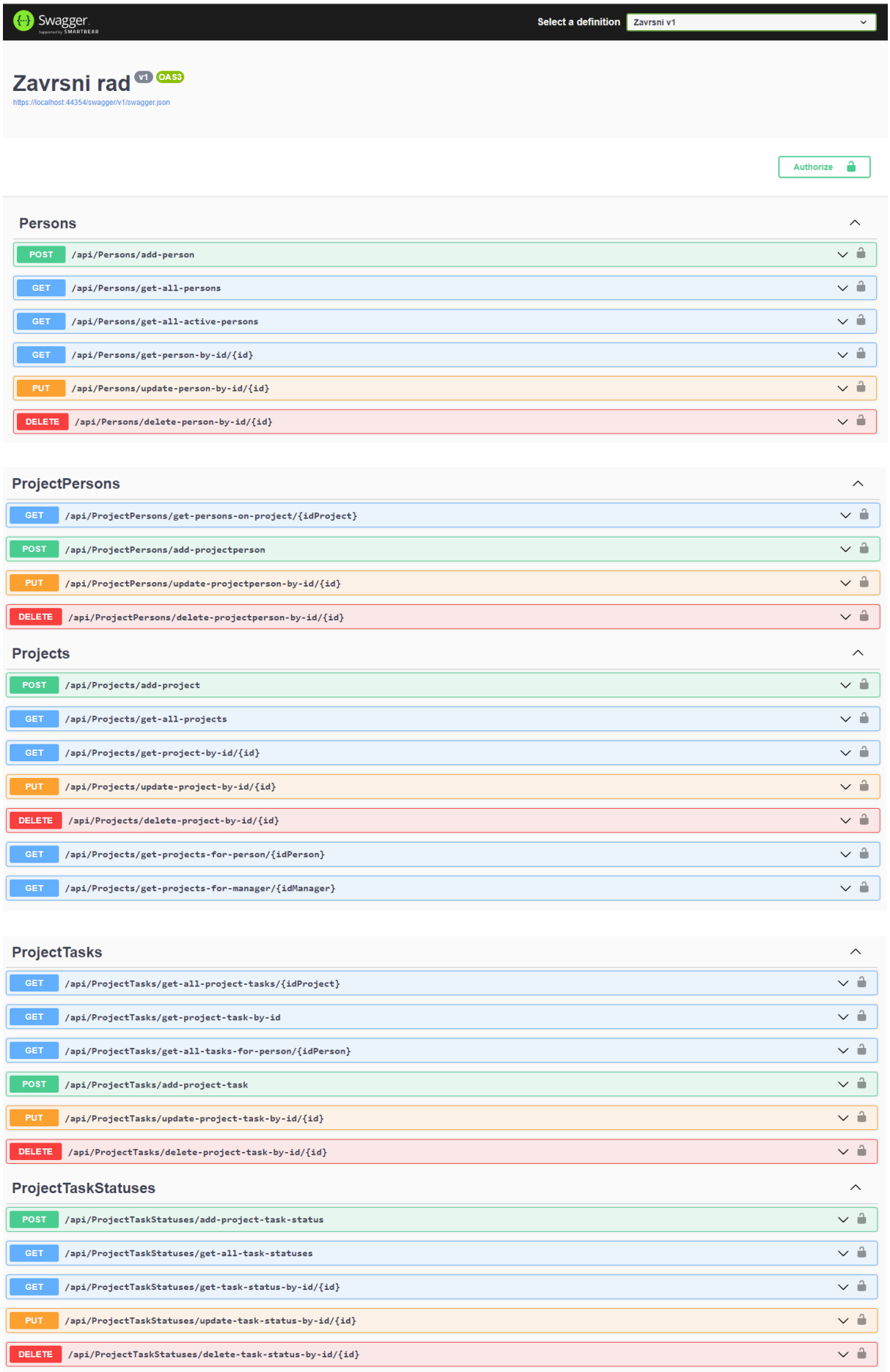
2.2.1. Implementirane metode

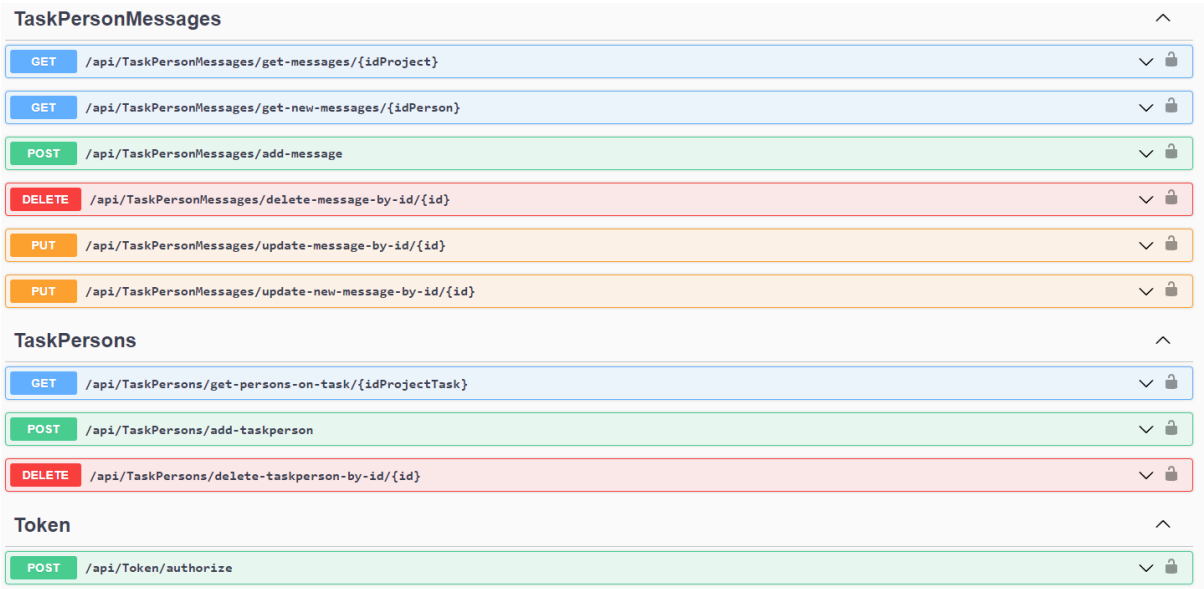
Budući da je Back-end napravljen poštujući REST načela, sama implementacija Back-enda, s gledišta Front-enda, nije bitna, jedino su bitni end-pointovi (metode) koji postoje te se implementacija može napraviti različitim alatima i tehnologijama.

U izradi ovog završnog za izradu WEB API metoda korišten je Visual Studio 2022, programski jezik C# i odabran tip projekta ASP.NET Core Web API.

U projekt je dodan Swashbuckle.AspNetCore paket koji implementira Swagger UI [1] specifikaciju i tako pruža web korisničko sučelje s prikazom svih metoda te mogućnost testiranje svake.

Implementirano Web sučelje, korištenjem Swagger UI-a prikazano je na slici (Slika 2.3).


Swagger
Select a definition
Zavrsni v1
Zavrsni rad
v1
OAS3
https://localhost:44354/swagger/v1/swagger.json
Authorize
Persons
POST /api/Persons/add-person
GET /api/Persons/get-all-persons
GET /api/Persons/get-all-active-persons
GET /api/Persons/get-person-by-id/{id}
PUT /api/Persons/update-person-by-id/{id}
DELETE /api/Persons/delete-person-by-id/{id}
ProjectPersons
GET /api/ProjectPersons/get-persons-on-project/{idProject}
POST /api/ProjectPersons/add-projectperson
PUT /api/ProjectPersons/update-projectperson-by-id/{id}
DELETE /api/ProjectPersons/delete-projectperson-by-id/{id}
Projects
POST /api/Projects/add-project
GET /api/Projects/get-all-projects
GET /api/Projects/get-project-by-id/{id}
PUT /api/Projects/update-project-by-id/{id}
DELETE /api/Projects/delete-project-by-id/{id}
GET /api/Projects/get-projects-for-person/{idPerson}
GET /api/Projects/get-projects-for-manager/{idManager}
ProjectTasks
GET /api/ProjectTasks/get-all-project-tasks/{idProject}
GET /api/ProjectTasks/get-project-task-by-id
GET /api/ProjectTasks/get-all-tasks-for-person/{idPerson}
POST /api/ProjectTasks/add-project-task
PUT /api/ProjectTasks/update-project-task-by-id/{id}
DELETE /api/ProjectTasks/delete-project-task-by-id/{id}
ProjectTaskStatuses
POST /api/ProjectTaskStatuses/add-project-task-status
GET /api/ProjectTaskStatuses/get-all-task-statuses
GET /api/ProjectTaskStatuses/get-task-status-by-id/{id}
PUT /api/ProjectTaskStatuses/update-task-status-by-id/{id}
DELETE /api/ProjectTaskStatuses/delete-task-status-by-id/{id}



Slika 2.3 - Swagger web sučelje

Kratak opis svake metode dan je u tablici (Tablica 2.1).

Persons		
POST	/api/Persons/add-person	Kreira i pohranjuje u tablicu Persons novog korisnika
GET	/api/Persons/get-all-persons	Vraća podatke za sve korisnike
GET	/api/Persons/get-all-active-persons	Vraća podatke za sve aktivne korisnike
GET	/api/Persons/get-person-by-id/{id}	Vraća podatke o korisniku čiji je jedinstveni identifikator jednak parametru {id}
PUT	/api/Persons/update-person-by-id/{id}	Ažurira podatke o korisniku čiji jedinstveni identifikator je jednak parametru {id} s pristiglim podacima
DELETE	/api/Persons/delete-person-by-id/{id}	Briše podatke o korisniku čiji jedinstveni identifikator je jednak parametru {id}
ProjectPersons		
GET	/api/ProjectPersons/get-persons-on-project/{idProject}	Vraća podatke za sve korisnike na projektu čiji jedinstveni identifikator je jednak parametru {id}
POST	/api/ProjectPersons/add-projectperson	Kreira i pohranjuje u tablicu ProjectPerson podatke da je korisnik na projektu
PUT	/api/ProjectPersons/update-projectperson-by-id/{id}	Ažurira podatke o korisniku na projektu čiji jedinstveni identifikator je jednak parametru {id}

DELETE	/api/ProjectPersons/delete-projectperson-by-id/{id}	Uklanja korisnika kao djelatnika na projektu iz tablice ProjectPersons čiji jedinstveni identifikator je jednak parametru {id}
Projects		
POST	/api/Projects/add-project	Kreira i pohranjuje u tablicu Projects novi projekt
GET	/api/Projects/get-all-projects	Vraća podatke o svim projektima
GET	/api/Projects/get-project-by-id/{id}	Vraća podatke o projektu čiji jedinstveni identifikator je jednak parametru {id}
PUT	/api/Projects/update-project-by-id/{id}	Ažurira podatke o projektu čiji jedinstveni identifikator je jednak parametru {id} s poslanim podacima
DELETE	/api/Projects/delete-project-by-id/{id}	Briše podatke o projektu čiji jedinstveni identifikator je jednak parametru {id}
GET	/api/Projects/get-projects-for-person/{idPerson}	Vraća podatke o svim projektima na kojima se nalazi korisnik čiji jedinstveni identifikator je jednak parametru {idPerson}
GET	/api/Projects/get-projects-for-manager/{idManager}	Vraća podatke o svim projektima na kojima je korisnik čiji jedinstveni identifikator je jednak parametru {idManager} u ulozi administratora sustava
ProjectTasks		
GET	/api/ProjectTasks/get-all-project-tasks/{idProject}	Vraća podatke o svim zadacima na projektu čiji jedinstveni identifikator je jednak parametru {idProject}
GET	/api/ProjectTasks/get-project-task-by-id/{id}	Vraća podatke o zadatku čiji jedinstveni identifikator je jednak parametru {id}
GET	/api/ProjectTasks/get-all-tasks-for-person/{idPerson}	Vraća sve zadatke koji su pridruženi korisniku čiji jedinstveni identifikator je jednak parametru {idPerson}
POST	/api/ProjectTasks/add-project-task	Kreira i pohranjuje u bazu u tablicu ProjectTasks novi zadatak.
PUT	/api/ProjectTasks/update-project-task-by-id/{id}	Ažurira podatke o zadatku čiji jedinstveni identifikator je jednak parametru {id}

DELETE	/api/ProjectTasks/delete-project-task-by-id/{id}	Briše podatke o zadatku čiji jedinstveni identifikator je jednak parametru {id}
ProjectTaskStatuses		
POST	/api/ProjectTaskStatuses/add-project-task-status	Kreira i pohranjuje u tablicu ProjectTaskStatuses novu vrstu statusa zadatka
GET	/api/ProjectTaskStatuses/get-all-task-statuses	Vraća sve statuse koje zadatak može imati
GET	/api/ProjectTaskStatuses/get-task-status-by-id/{id}	Vraća podatke o statusu čiji je jedinstveni identifikator je jednak parametru {id}
PUT	/api/ProjectTaskStatuses/update-task-status-by-id/{id}	Ažurira podatke o statusu čiji jedinstveni identifikator je jednak parametru {id} s poslanim podacima
DELETE	/api/ProjectTaskStatuses/delete-task-status-by-id/{id}	Briše podatke o statusu čiji jedinstveni identifikator je jednak parametru {id}
TaskPersonMessages		
GET	/api/TaskPersonMessages/get-messages	Za osobu koja je administrator projekta vraća sve poruke između ljudi na projektu. Za korisnika koja ima samo ulogu Djelatnik vraća poruke koje je ona poslala ili primila.
GET	/api/TaskPersonMessages/get-new-messages/{idPerson}	Vraća sve nepročitane poruke za korisnika čiji jedinstveni identifikator je jednak parametru {idPerson}
POST	/api/TaskPersonMessages/add-message	Kreira i dodaje u tablicu TaskPersonMessages novu poruku.
DELETE	/api/TaskPersonMessages/delete-message-by-id/{id}	Briše poruku čiji jedinstveni identifikator je jednak parametru {id}
PUT	/api/TaskPersonMessages/update-message-by-id/{id}	Ažurira poruku čiji jedinstveni identifikator je jednak parametru {id} s poslanim podacima
PUT	/api/TaskPersonMessages/update-new-message-by-id/{id}	Ažurira u tablici da je poruka čiji jedinstveni identifikator je jednak parametru {id} pročitana
TaskPersons		
GET	/api/TaskPersons/get-persons-on-task/{idProjectTask}	Vraća sve korisnika na zadatku iz tablice TaskPersons s jedinstvenim identifikatorom jednakim parametru { idProjectTask}

POST	/api/TaskPersons/add-taskperson	Kreira i pohranjuje u tablicu TaskPersons zapis da je korisniku pridružen zadatak
DELETE	/api/TaskPersons/delete-taskperson-by-id/{id}	Briše osobu na zadatku čiji jedinstveni identifikator je jednak parametru {id}
Token		
POST	/api/Token/authorize	Provjerava je li u bazi postoji korisnik s poslanim korisničkim imenom i lozinkom i ako da vraća JWT token
WorkingHours		
POST	/api/WorkingHours/add-workinghour	Kreira zapis o radu u jednom danu za zadanog djelatnika u tablicu WorkingHours
POST	/api/WorkingHours/get-all-wh-for-month	Vraća podatke o radu korisnika za svaki dan u zadanom mjesecu
POST	/api/WorkingHours/get-wh-for-person-project	Vraća podatke o radu korisnika za svaki dan u zadanom mjesecu i zadanom projektu
PUT	/api/WorkingHours/update-wh-by-id/{id}	Ažurira podatak o radu u zadanom danu za zadanog korisnika u tablici WorkingHours čiji jedinstveni identifikator je jednak parametru {id}
DELETE	/api/WorkingHours/delete-wh-by-id/{id}	Briše podatak o radu u zadanom danu za zadanog korisnika iz tablice WorkingHours čiji jedinstveni identifikator je jednak parametru {id}
POST	/api/WorkingHours/add-locked-month	Kreira zapis u tablici LockedWorkingHour za zadani mjesec, djelatnika i projekt pa ga djelatnik više ne može mijenjati
POST	/api/WorkingHours/is-locked-month	Vraća podatak jesu li podaci o djelatniku za projekt i mjesec zaključani
POST	/api/WorkingHours/delete-locked-month	Briše podatak iz tablice LockedWorkingHour za zadani mjesec, djelatnika i projekt. Tako se omogućava promjena zapisa od strane djelatnika.

Tablica 2.11 - Popis svih metoda WEB API-ja

2.2.2. Model podataka

Sva komunikacija prema bazi podataka implementirana je korištenjem Entity Framework Core [2] biblioteka koje su uključene u projekt.

Korišten je pristup koji se naziva "Code First", a označava da se u programskom kodu implementiraju klase i ovisnosti među njima iz kojih EntityFramework Core može generirati shemu baze podataka.

"Code first" je pristup gdje se model podataka stvara u programskom kodu i koristan je kada se želi imati potpunu kontrolu nad shemom baze podataka i onemogućiti njeno uređivanje izvan programskog koda.

Suprotno pristupu "Code first" je pristup "Database first" gdje se najprije stvara shema baze podataka u nekom od grafičkih alata (npr. SQL Server Management Studio), a zatim koristi Entity Framework Core ili neka druga biblioteka za mapiranje klasa na relacije (npr. Dapper) za generiranje klasa koje joj odgovaraju. Ovo je pogodno ako već postoji baza podataka koju se želi koristiti s aplikacijom, ili ako se preferira dizajniranje modela podataka koristeći neki grafički alat.

Osim klasa koje predstavljaju poslovne entitete (od njih će nastati relacije u bazi podataka), da bi Entity Framework mogao generirati shemu baze podataka potrebno je implementirati i klasu koja predstavlja kontekst baze podataka. Ova klasa se stvara izvedbom iz klase Microsoft.EntityFrameworkCore.DbContext koja je dio Entity Framework Core biblioteke.

U okviru ovog završnog rada ta klasa je implementirana na sljedeći način:

```
public class AppDbContext:DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext>
options):base(options)
        {
        }
        protected override void OnModelCreating(ModelBuilder
modelBuilder)
        {
            modelBuilder.Entity<TaskPerson>()
                .HasOne(a => a.ProjectTask)
```

```
            .WithMany(b => b.TaskPersons)
            .HasForeignKey(c => c.ProjectTaskId)
            .OnDelete(DeleteBehavior.NoAction);

        modelBuilder.Entity<TaskPerson>()
            .HasOne(a => a.ProjectPerson)
            .WithMany(b => b.TaskPersons)
            .HasForeignKey(c => c.ProjectPersonId)
            .OnDelete(DeleteBehavior.NoAction);

        modelBuilder.Entity<ProjectPerson>()
            .HasOne(a => a.Project)
            .WithMany(b => b.ProjectPersons)
            .HasForeignKey(c => c.ProjectId)
            .OnDelete(DeleteBehavior.NoAction);

        modelBuilder.Entity<ProjectPerson>()
            .HasOne(a => a.Person)
            .WithMany(b => b.ProjectPersons)
            .HasForeignKey(c => c.PersonId)
            .OnDelete(DeleteBehavior.NoAction);
    }

    public DbSet<Project> Projects { get; set; }
    public DbSet<Person> Persons { get; set; }
    public DbSet<ProjectTask> ProjectTasks { get; set; }
    public DbSet<ProjectTaskStatus> ProjectTaskStatuses { get; set; }
    public DbSet<TaskPerson> TaskPersons { get; set; }
    public DbSet<TaskPersonMessage> TaskPersonMessages { get; set; }
    public DbSet<WorkingHour> WorkingHours { get; set; }
```

```
        public DbSet<ProjectPerson> ProjectPersons { get; set; }
        public DbSet<LockedWorkingHour> LockedWorkingHours { get; set; }
    }
```

Kod 2.1. – Klasa: Kontekst baze podataka

Osim kolekcija (DbSet) poslovnih entiteta iz kojih će nastati relacije, u ovom primjeru iz završnog rada opisane i veze N:N koje postoje, a to su veza između relacija ProjectTasks i ProjectPersons te relacija Projects i Persons.

Podatak za spajanje na bazu podataka (eng. connection string) unesen je u datoteku appsettings.json:

```
"ConnectionStrings": {
    "DefaultConnectionString":
      "Data Source=localhost\\SQLEXPRESS;Initial Catalog=ProjectDB;
      Integrated Security=True;Pooling=False;Encrypt=False"
},
```

i njega ASP.NET Core putem Dependency Injection (DI) mehanizmima ubacuje preko parametara konstruktora: `DbContextOptions<AppDbContext> options` u klasu koja predstavlja kontekst baze podataka.

U konzoli za rad s paketima unutar Visual Studija naredbom:

```
Add-Migration prozvoljan_naziv_migracije
```

Generira se kod za stvaranje sheme baze podataka temeljene na modelu koji je specificiran u klasi `AppDbContext`.

Da bi iz generiranog koda nastala relacijska shema potrebno je pokrenuti naredbu:

```
Update-Database
```

izvršavanje koje Entity Framework Core stvara bazu podataka.

2.2.3. Autentifikacija korisnika

Svi end-pointovi implementiranog Back-end rješenja u okviru završnog rada zaštićeni su od neovlaštenog pristupa. Jedini koji nije zaštićen je "/token/authorize" koji mora biti otvoren kako bi se korisnik probao prijaviti u sustav slanjem korisničkog imena i lozinke. REST API

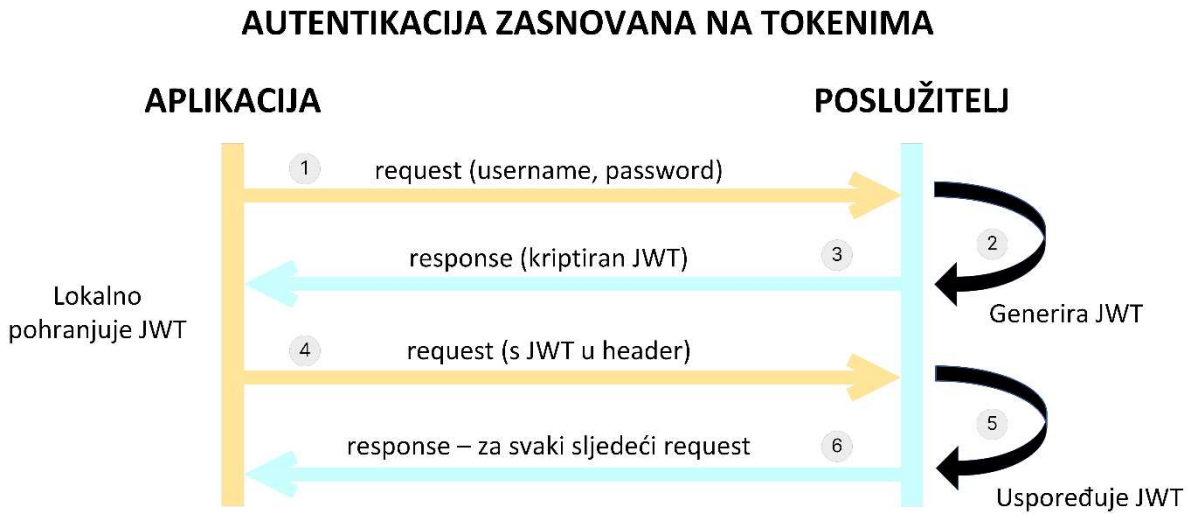
rješenje implementirano u okviru ovog završnog rada koristi mehanizam JSON Web Token (JWT) za autentifikaciju korisnika.

JSON Web Token [3] je otvoren standard (RFC 7519) koji definira kompaktni i samodostatan način za sigurni prijenos informacija između različitih strana kao JSON objekt. Ove informacije mogu se provjeriti i pouzdane su jer su digitalno potpisane. JWT-ovi se mogu potpisati koristeći tajni ključ (uz HMAC algoritam što se koristi u završnom radu) ili par javnog/privatnog ključa koristeći RSA ili ECDSA [4] algoritme.

Nakon unosa korisničkih podataka unutar klijentske aplikacije na stranici za prijavu ona šalje te podatke na end-point "/token/authorize". Metoda REST API koja se poziva provjerava u bazi postoji li korisnik s takvim podacima, te generira JWT token koji je potpisan korištenjem tajnog ključa i vraća ga klijentskoj aplikaciji.

Kada korisnik kroz aplikaciju pristupa zaštićenim end-pointovima klijentska aplikacija više ne šalje prema WEB API korisničke podatke već u svaki poziv metode ugrađuje JWT, u zaglavlju Authorization koristeći Bearer shemu. Zaštićene end-pointovi provjeravaju valjanost JWT-a i ako je valjan, korisnik će biti dopušten. Budući da su JWT-ovi samodostatni, sve potrebne informacije su sadržane u njima, smanjujući potrebu za stalnim povratom i slanjem zahtjeva bazi podataka.

Slika 2.4 prikazuje način autentifikacije zasnovane na JWT tokenima.



Slika 2.4 - Autentifikacija JWT tokenima

Da bi se u REST API projekt uključila autentifikacija JWT-om potrebno je u projekt uključiti paket: Microsoft.AspNetCore.Authentication.JwtBearer.

2.3. Klijentska aplikacija

U okviru završnog zadatka klijentska aplikacija implementirana je kao Web aplikacija.

Može se reći da su danas Web aplikacije najzastupljeniji način razvoja aplikacija iz više razloga:

1. Pristupačnost:

Web aplikacijama se može pristupiti s računala, pametnih telefona, tableta i drugih uređaja bez potrebe za preuzimanjem i instaliranjem dodatnog softvera. Dovoljan je pristup internetu i instalirani Web preglednik.

2. Platforma nezavisnost:

Korisnici mogu pristupiti web aplikacijama bez obzira koji operativni sustav koriste poput (Windows, macOS-a, Linux-a itd.)

3. Lako održavanje:

Ažuriranje Web aplikacije se obavlja samo na serverskoj strani, a korisnici automatski dobivaju najnoviju verziju aplikacije prilikom pristupa. To je posebno korisno kod većih aplikacija s velikim brojem korisnika.

4. Jednostavna distribucija:

Nema potrebe za distribucijom i instalacijom aplikacije na svakom korisničkom uređaju jer se Web aplikacija nalazi se na serveru, a korisnici pristupaju putem web preglednika.

5. Skalabilnost:

Web aplikacije se lako skaliraju, bez utjecaja na korisničko iskustvo, dodavanjem više resursa na serverskoj strani.

6. Interaktivnost:

Razvoj interaktivnih korisničkih sučelja omogućen je tehnologijama poput JavaScript-a, HTML-a i CSS-a.

7. Povezanost i integracija:

Web aplikacije omogućavaju jednostavnu integraciju s drugim web uslugama i API. Mogu se povezati s različitim sustavima i izvorima podataka kako bi korisnicima pružile sveobuhvatno iskustvo.

2.3.1. Tehnologije i alati

Klijentska aplikacija implementirana je unutar Visual Studio Code programskog okruženja, a za implementaciju korištene su sljedeće tehnologije i alati:

- **DevExpress** [5] – skup kontrola za jednostavniju i bržu izradu WEB stranica. Kao podloga za Web aplikaciju poslužila je DevExpress aplikacija skinuta s linka: https://js.devexpress.com/Documentation/Guide/Vue_Components/Application_Template/. DevExpress kontrole je slobodno koristiti za nekomercijalne projekte, a za komercijalne potrebno je kupiti licencu. Najkorisnija kontrola iz skupa DevExpress kontrola je sigurno DxDataGrid kontrola koja omogućava tablični prikaz podataka iz baze ali i relativno jednostavnu nadogradnju za mogućnost potpune manipulacije podacima tj. ugradnja CRUD operacija **C**reate**R**ead**U**pdate**D**elete (kreiranje, čitanje, ažuriranje i brisanje) za podatke u bazi.
- **Vue** [6] **-** open-source JavaScript okruženje za izradu korisničkih sučelja i jednostranih aplikacija SPA (eng. single-page application) koji se temelji na modelu–prikazu–modelu prikaza (eng. Model-View-ViewModel). Kreirao ga je Evan You, a održavaju ga on i ostali članovi aktivnog jezgre tima. Jednostrana aplikacija je web aplikacija ili web stranica koja komunicira s korisnikom tako da dinamički mijenja trenutnu web stranicu novim podacima s poslužitelja, umjesto uobičajenog načina učitavanja cijelih novih stranica u web pregledniku. Cilj je postići brže prijelaze koji čine da web stranica djeluje više poput *native* aplikacije. U jednostranim aplikacijama se nikada ne događa osvježavanje stranice; umjesto toga, sav potreban HTML, JavaScript i CSS kod se ili preuzima preglednikom s jednim učitavanjem stranice, ili se odgovarajući resursi dinamički učitavaju i dodaju na stranicu prema potrebi, obično kao odgovor na akcije korisnika. Neki od besplatnih popularnih JavaScript okvira i biblioteka koji su usvojili principe jednostranih aplikacija su AngularJS, Ember.js, Knockout.js, Meteor.js, React, Svelte i Vue. Neke od najpoznatijih firmi koje koriste Vue za svoje Web stranice su Apple za (SwiftUI Tutorials), Google (za Google Careers), Zoom, GitLab, Microsoft (Microsoft Edge Landing Page)

- **Vuex** [7] - biblioteka za upravljanje stanjem u Vue aplikacijama. Kada se razvijaju kompleksne aplikacije, stanje podataka može postati teško pratiti i održavati. Vuex rješava ove izazove pružajući centralizirano skladište (eng. *store*) koje sadrži sve podatke aplikacije. Glavna ideja iza Vuex-a je da se stanje aplikacije pohrani na jednom mjestu, koje je dostupno svim komponentama. Komponente mogu čitati stanje iz skladišta i izvršavati mutacije za promjenu tog stanja. Mutacije su strogo kontrolirane i osiguravaju da se stanje može mijenjati samo na predvidljiv način. Korištenje Vuex-a donosi brojne prednosti. Omogućava bolju organizaciju i praćenje stanja aplikacije, poboljšava skalabilnost i olakšava dijeljenje podataka između komponenti. Također olakšava testiranje jer se logika upravljanja stanjem nalazi na jednom mjestu. Najnovija verzija upravljanja stanjem u Vue aplikacijama je Pinia [8] ali kako je većina primjera koji se mogu naći na internetu napravljena pomoću Vuex biblioteke ona je korištena u implementaciji završnog rada.
- **Axios** [9] - JavaScript biblioteka koja se koristi za slanje HTTP zahtjeva iz node.js okruženja ili XMLHttpRequest-ova iz preglednika. Omogućava izvršavanje HTTP zahtjeva prema serveru i manipuliranje podacima koji su vraćeni kao odgovor. Axios pruža intuitivno sučelje te podržava različite vrste zahtjeva poput GET, POST, PUT, DELETE itd. Axios također omogućava automatsku pretvorbu podataka iz JSON formata u JavaScript objekte ili iz XML formata u JavaScript objekte, ovisno o konfiguraciji.

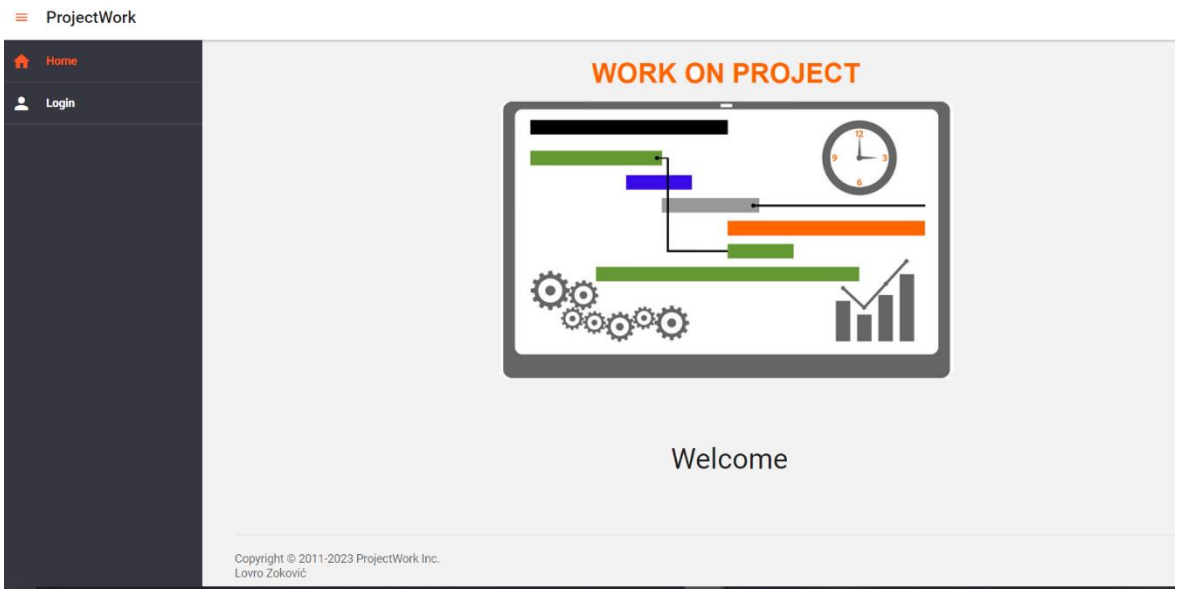
3. Korištenje klijentske aplikacije

Slijedi opis korištenja aplikacije prema ulogama korisnika. Izbornik klijentske aplikacije, a time i stranice koje se korisniku otvaraju ovisi o ulazi korisnika. Svi prijavljeni korisnici imaju ulogu Djelatnik i s njom vide određeni skup stranica. Ako korisnik ima ulogu Administrator i s njom mu se izbornik širi, kao i ako ima ulogu Voditelja projekta.

3.1. Stranice koje vidi Djelatnik

3.1.1. Početna stranica aplikacije

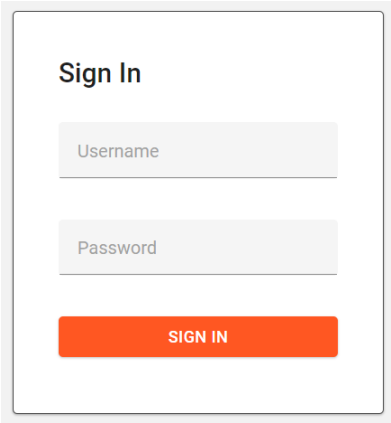
Početna stranica aplikacije (Slika 3.1). sadrži samo poveznicu Login za stranicu za prijavu korisnika. Korisniku stranice dok se ne prijavi zapravo nije omogućena nikakva funkcionalnost aplikacije.



Slika 3.1 - Početna stranica aplikacije

3.1.2. Stranica za prijavu

Stranica (Slika 3.2) omogućava unos korisničkog imena i lozinke.



Slika 3.2 - Stranica za prijavu

Ako korisnik postoji u sustavu na početnoj stranici pojavit će mu se:

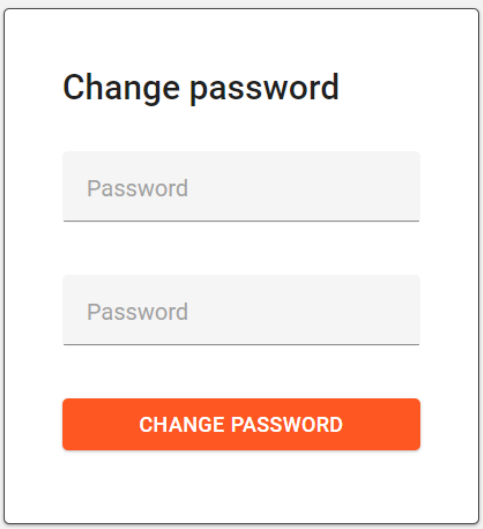
1. Izbornik s poveznicama
2. Poveznice za odjavu i promjenu lozinka

kao što je vidljivo na slici Slika 3.3.

Slika 3.3 - Početna stranica prijavljenog korisnika

3.1.3. Stranica za promjenu lozinke

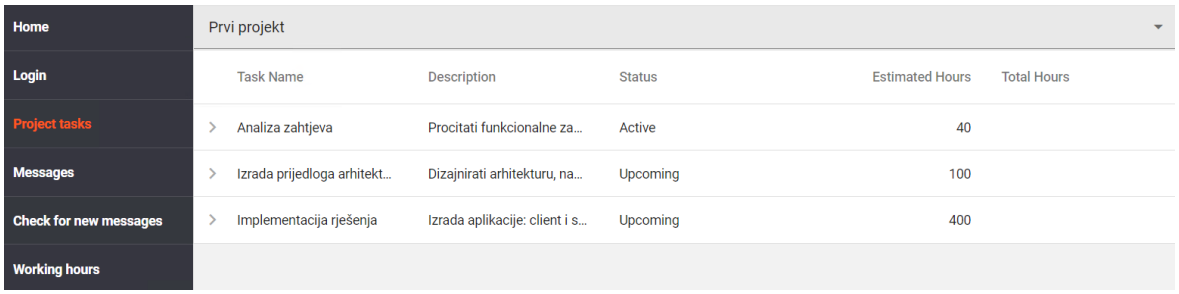
Stranica (Slika 3.4) služi da si korisnik može promijeniti lozinku ako ju je zaboravio. Da bi on to mogao prvo mora javiti (ili otići do njega osobno) administratoru sustava da mu postavi lozinku na neku *dummy* vrijednost kako bi se uopće mogao logirati u sustav. Čim administrator to napravi korisnik se može logirati s tom *dummy* lozinkom i odmah promijeniti je u samo njemu poznatu preko poveznice Change Password. Ovdje postoji sigurnosna rupa jer u jednom kratkom vremenu administrator sustava zna lozinku korisnika pa bi se taj dio u komercijalnoj aplikaciji mogao doraditi slanjem *dummy* lozinke preko maila.



Slika 3.4 - Stranica za promjenu lozinke

3.1.4. Pregled zadataka i korisnika na projektu

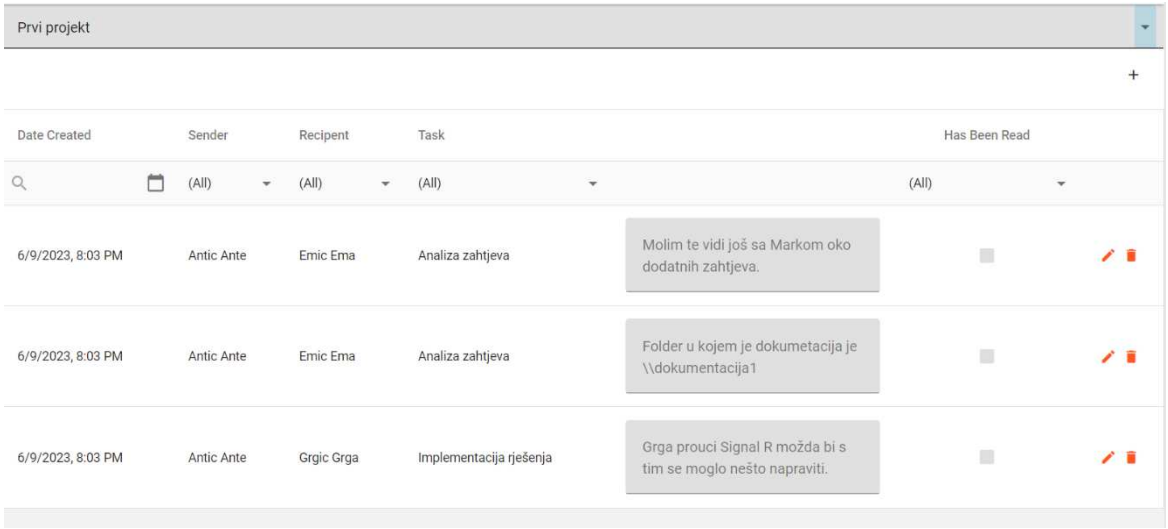
Stranica (Slika 3.5) za pregled definiranih zadataka po projektima i njima pridruženi korisnici otvara se na poveznicu Project tasks. Korisnik ima mogućnost odabira projekta kojem je pridružen i pregleda osnovnih informacija o zadacima na projektu. Korisnik nema dozvolu mijenjati ili dodavati podatke.



Slika 3.5 - Stranica za pregled zadataka

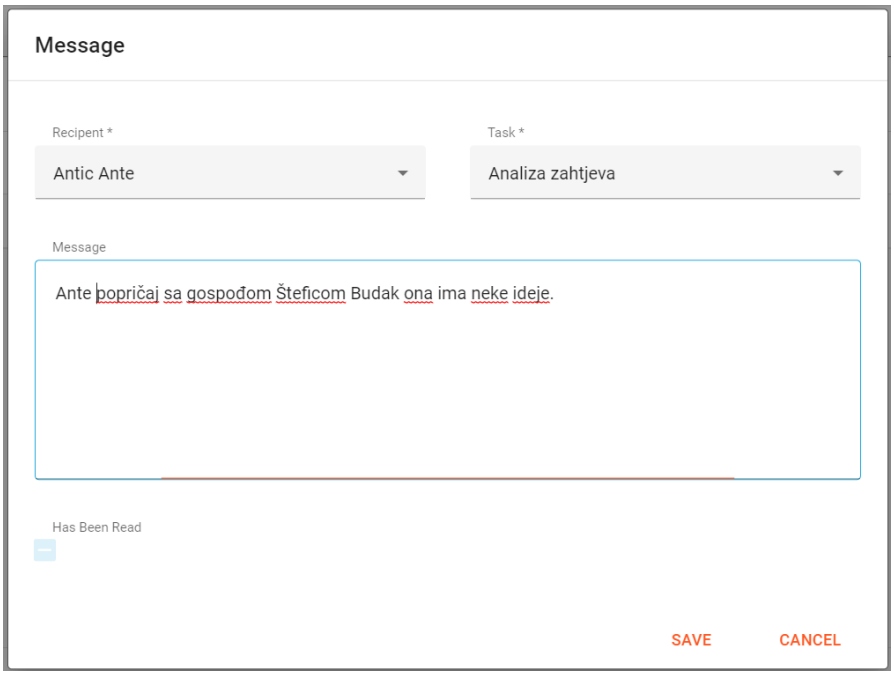
3.1.5. Kreiranje i pregled poruka

Stranica (Slika 3.6) za kreiranje i pregled poruka otvara se na poveznicu Messages. Odbirom jednog od projekta na kojima korisnik radi može vidjeti sve poruke su mu poslane ili koje je poslao korisnicima na tom projektu, sortirati ih i filtrirati, čak i mijenjati i pobrisati ako su to poruke koje je sam kreirao, a još nisu označene kao pročitane.



Slika 3.6 - Stranica za poruke

Nova poruka se unosi klikom na znak + u desnom uglu pri čemu se otvara skočna (eng. Pop-up) forma (Slika 3.7) u kojoj odabire korisnika kojem šalje poruku, zadatak na koji će se poruka odnositi tekst poruke.

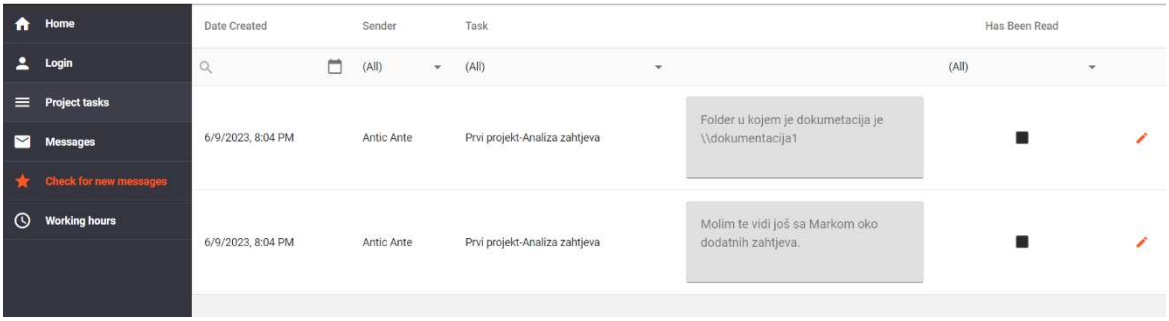


Slika 3.7 - Forma za kreiranje poruke

Prilikom kreiranja poruke korisnik sam ne može mijenjati vrijednost polja je li poruka pročitana (polje Has Been Read), već će njega moći promijeniti primatelj poruke da potvrdi pošiljatelju poruke da ju je pročitao.

3.1.6. Provjera novih poruka

Korisnik može na brzinu saznati ima li novih poruka preko poveznice Check for new messages koja otvara stranicu (Slika 3.8) s nepročitanom porukama

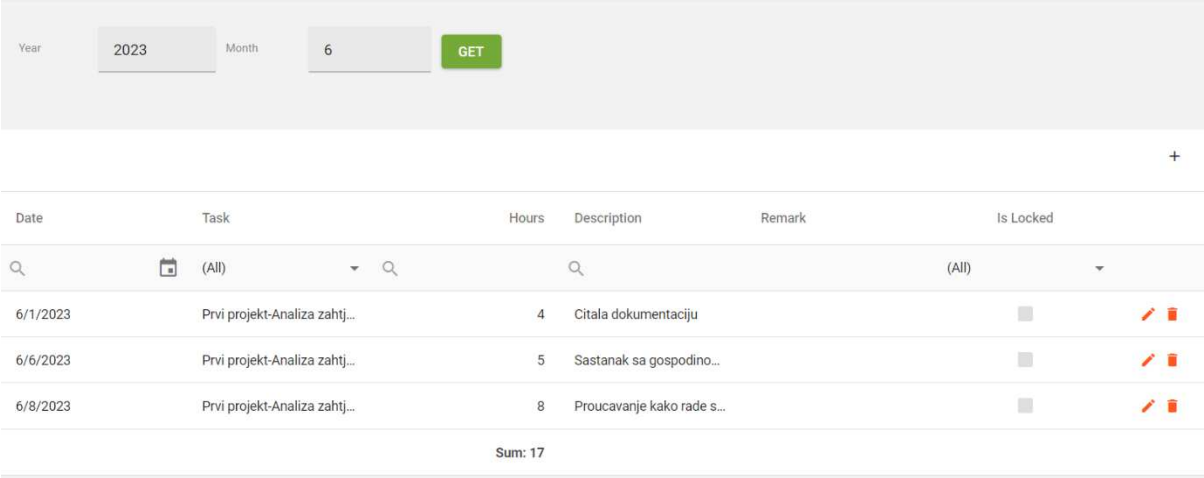


Slika 3.8 - Stranica za nepročitane poruke

Pregledom poruka može potvrditi njihov primitak (polje Has Been Read) i nakon toga ih više neće moći vidjeti unutar ove stranice ali ih uvijek može vidjeti na stranici za kreiranje i pregled poruka (poveznica Messages).

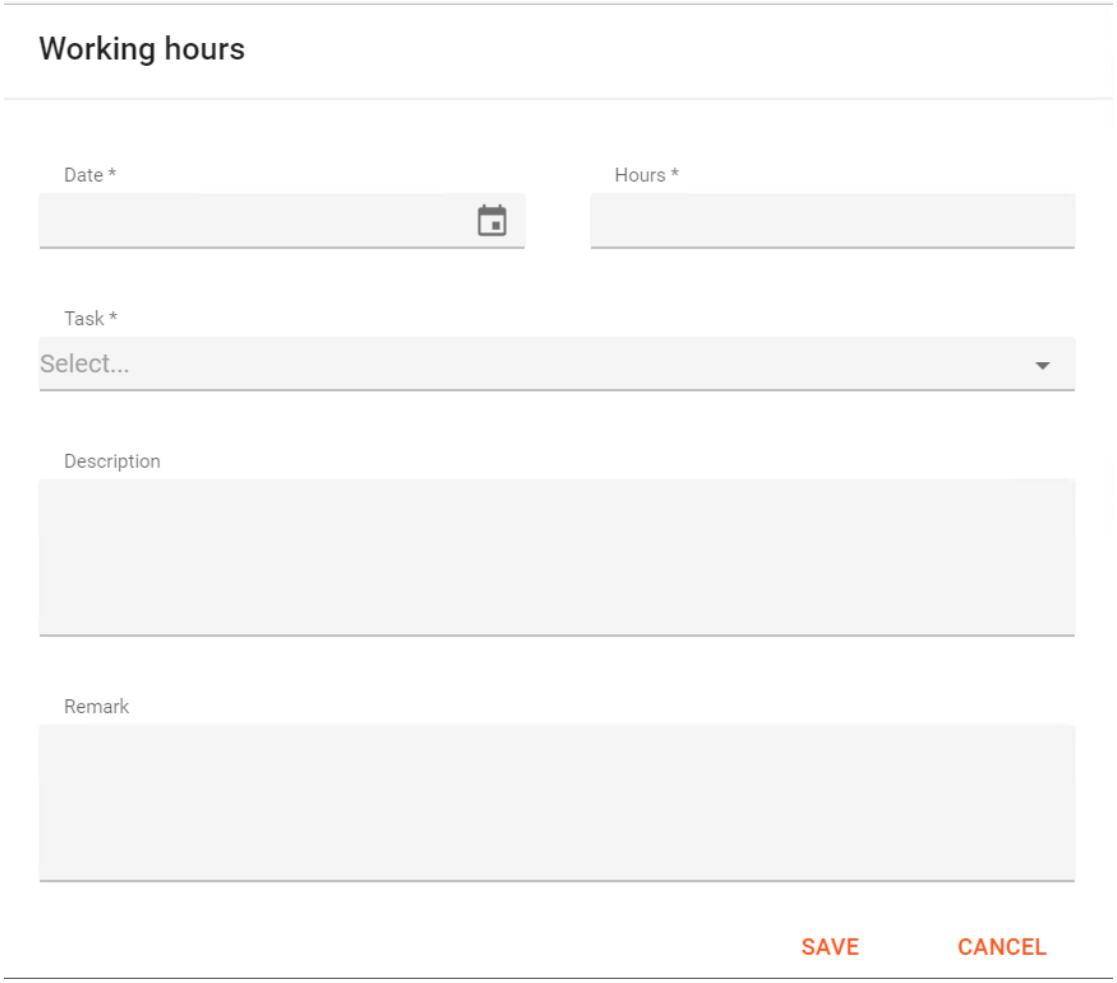
3.1.7. Unos sati rada po danima

Stranica u kojoj korisnik može unijeti aktivnosti i utrošene sate (Slika 3.9) otvara se na poveznicu Working hours.



Slika 3.9 - Stranica za unos rada

Da bi dohvatio ili unosio podatke korisnik mora unijeti godinu i mjesec i pritisnuti gumb GET. Nakon toga dohvaćaju se i prikazuju podaci za odabrani i mjesec i korisnik ima mogućnost dodavanja novih podataka (na znak +) nakon čega mu se otvara popup forma za unos (Slika 3.10).

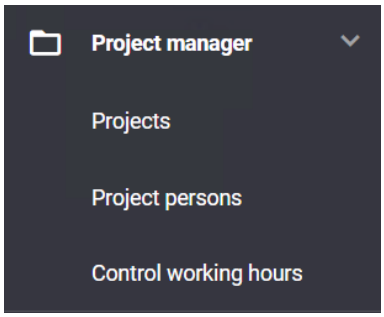


Slika 3.10 - Forma za unos radnih sati

Korisnik ima i mogućnost promjene postojećih podataka ili brisanja odabirom odgovarajuće ikone uz svaki redak. Korisnik može dodavati, brisati i mijenjati podatke za jedan mjesec na jednom projektu sve dok voditelj projekta ne zaključa podatke za taj mjesec.

3.2. Stranice koje vidi Voditelj projekta

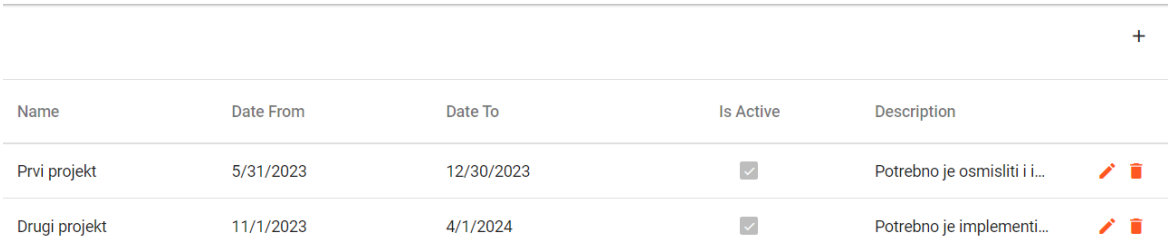
Kada se prijavi korisnik koji ima ulogu Voditelja projekata izbornik za ulogu Djelatnika se proširuje s dodatnim poveznicama (Slika 3.11) koje mu pružaju mogućnost kreiranja projekata, stavljanja korisnika na projekte te kontrolu sati korisnika na njegovim projektima.



Slika 3.11 - Izbornik Voditelja projekta

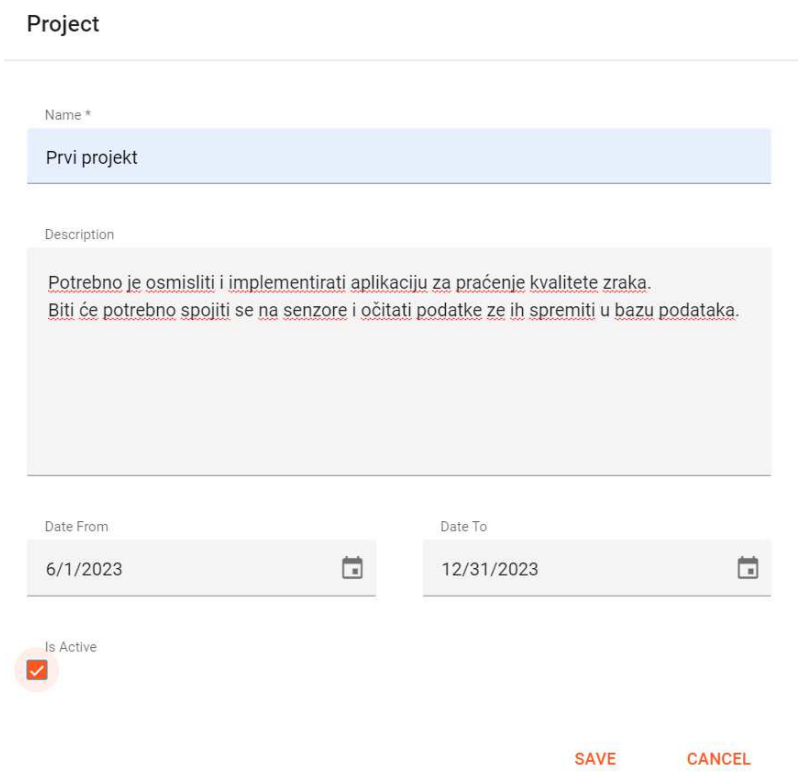
3.2.1. Kreiranje projekata

Stranica za pregled i kreiranje projekata (Slika 3.12) otvara se na poveznicu Projects



Slika 3.12 - Stranica za projekte

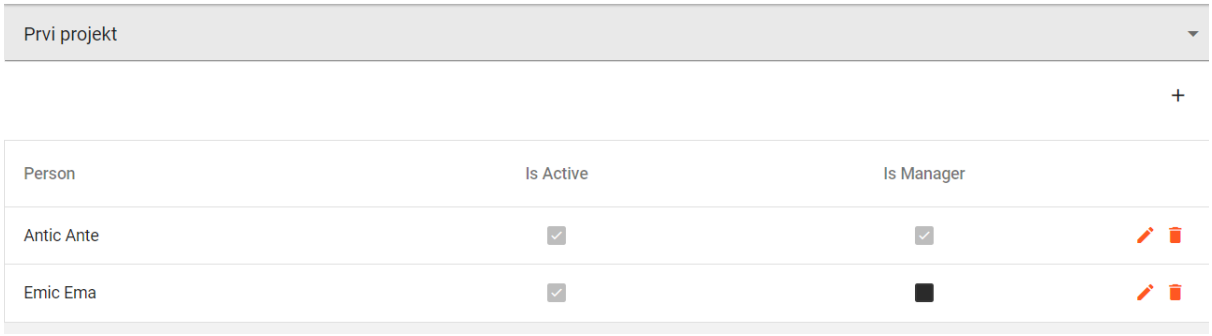
Korisniku se prikazuju svi projekti koje je sam kreirao i one projekte za koje je dobio dozvole od nekog drugog voditelja. Prilikom kreiranja novog projekta on sam postaje i djelatnik na tom projektu s dozvolom administriranja podataka. Novi projekt se kreira pritiskom na gumb + koji otvara popup formu (Slika 3.13) koju treba ispuniti s informacijama o novom projektu.



Slika 3.13 - Forma za kreiranje projekta

3.2.2. Dodavanje korisnika na projekt

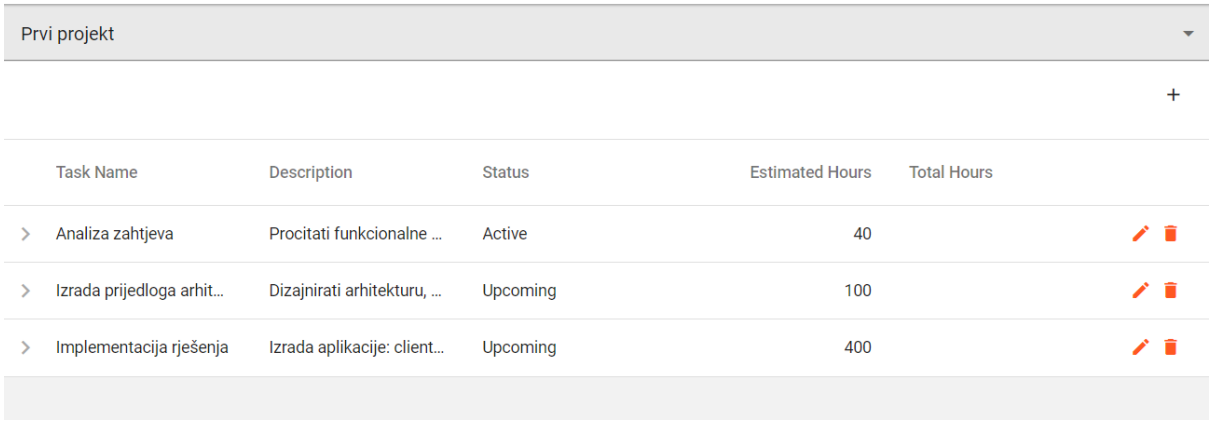
Stranica za dodavanje korisnika na projekt (Slika 3.14) se otvara na poveznicu Project persons i omogućuje voditelju projekta da na projekt osim sebe stavi i druge korisnike sustava. On može dodanom korisniku staviti da je ima administratorske dozvole za projekt koje se odnose samo na taj projekt (polje Is Manager). To nije istovjetno ulozi Voditelja projekata koje podrazumijeva mogućnost kreiranja novih projekata i kontrole sati djelatnika što administrator na projektu nema. Administrator na projektu može dodavati nove zadatke na projekt, stavljati djelatnike na zadatke i pregledati sve poruke između korisnika na projektu.



Slika 3.14 - Stranica za dodavanje korisnika u projekt

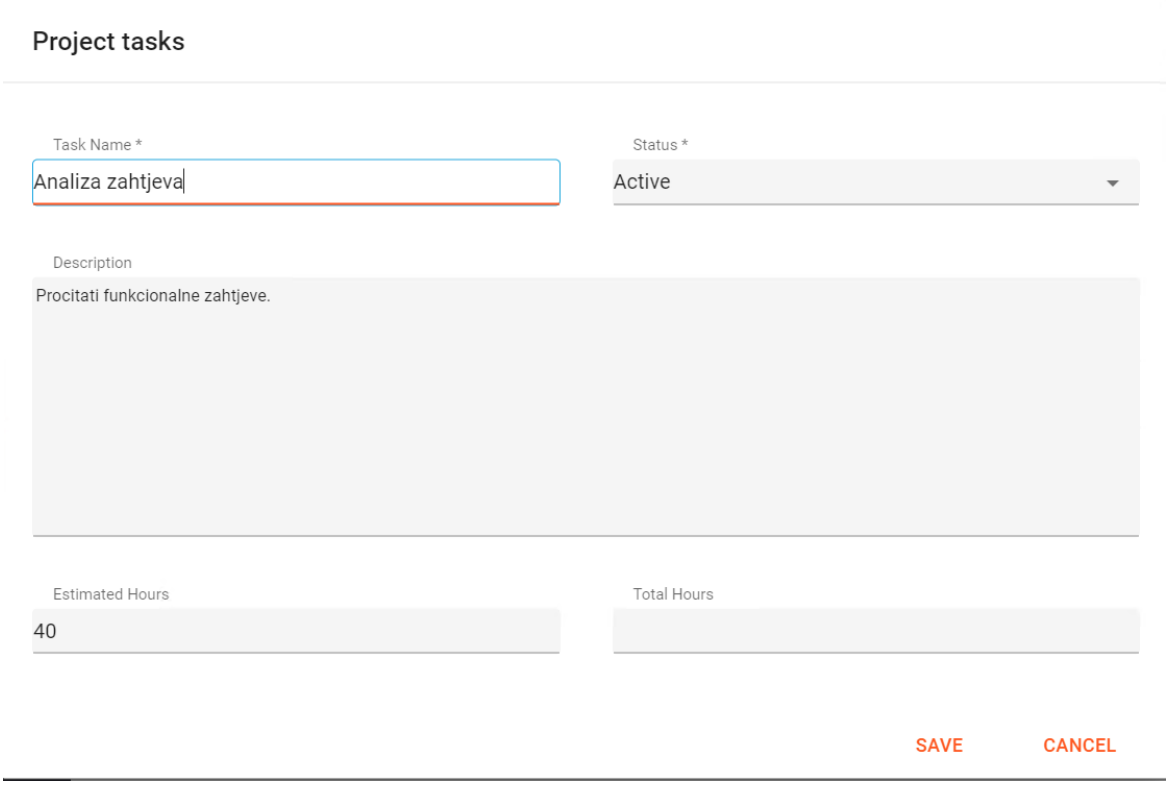
3.2.3. Definiranje projektnih zadataka i pridruživanje korisnika

Ova opcija nije nadodana voditelju već se nalazi u poveznicama izbornika za korisnika s ulogom Djelatnik (poveznica Projects tasks, otvara stranicu (Slika 3.15)) pa je Voditelj može koristiti ali za razliku od običnog korisnika on odabirom te opcije može i dodavati zadatke i pridruživati djelatnike na zadatke, a ne samo pregledati podatke. Međutim to isto može i korisnik u ulozi djelatnika ako mu je Voditelj projekta dao dozvolu administriranja projekta.



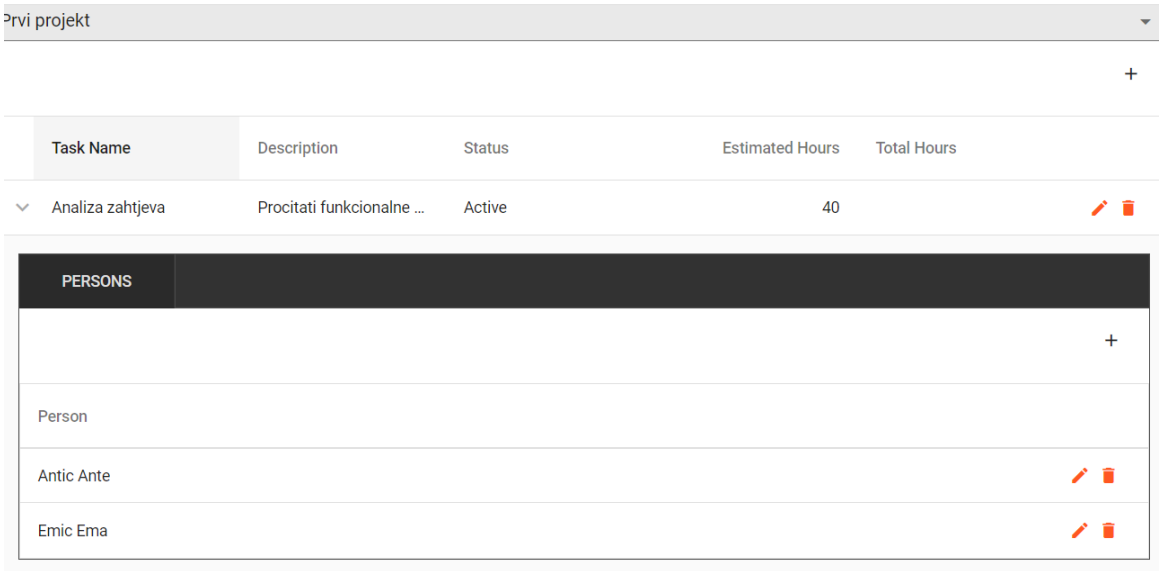
Slika 3.15 - Stranica zadataka na projektu

Pritiskom na + otvara se popup forma (Slika 3.16) za dodavanje novog zadatka na odabrani projekt.



Slika 3.16 - Forma za dodavanje novog zadatka u projektu

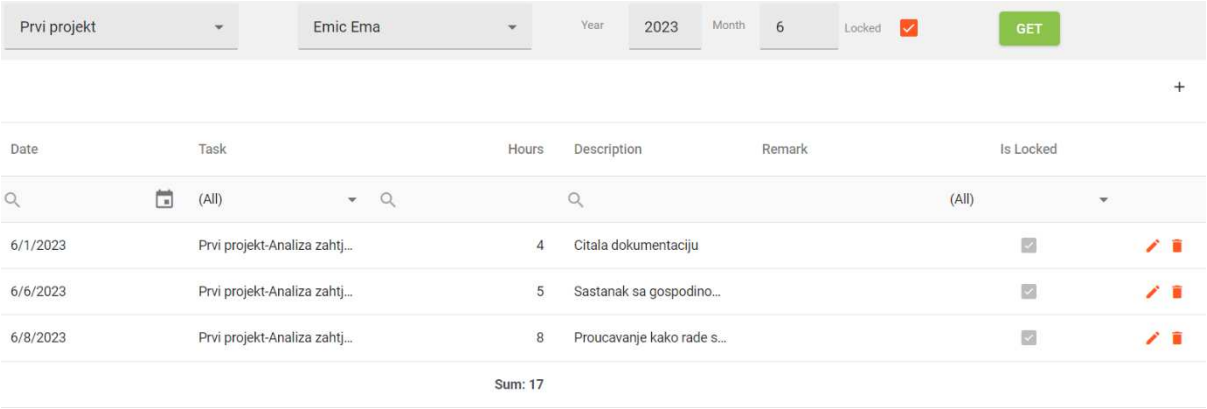
Nakon što je definiran zadatak njemu je moguće pridružiti djelatnike koji su članovi projekta čiji je to zadatak. (Slika 3.17). Djelatnici na projektu unose sate rada po zadatku na projektu zato je potrebno napraviti pridruživanje djelatnika i na zadatak a ne samo na projekt.



Slika 3.17 - Stranica za pridruživanje djelatnika zadatku

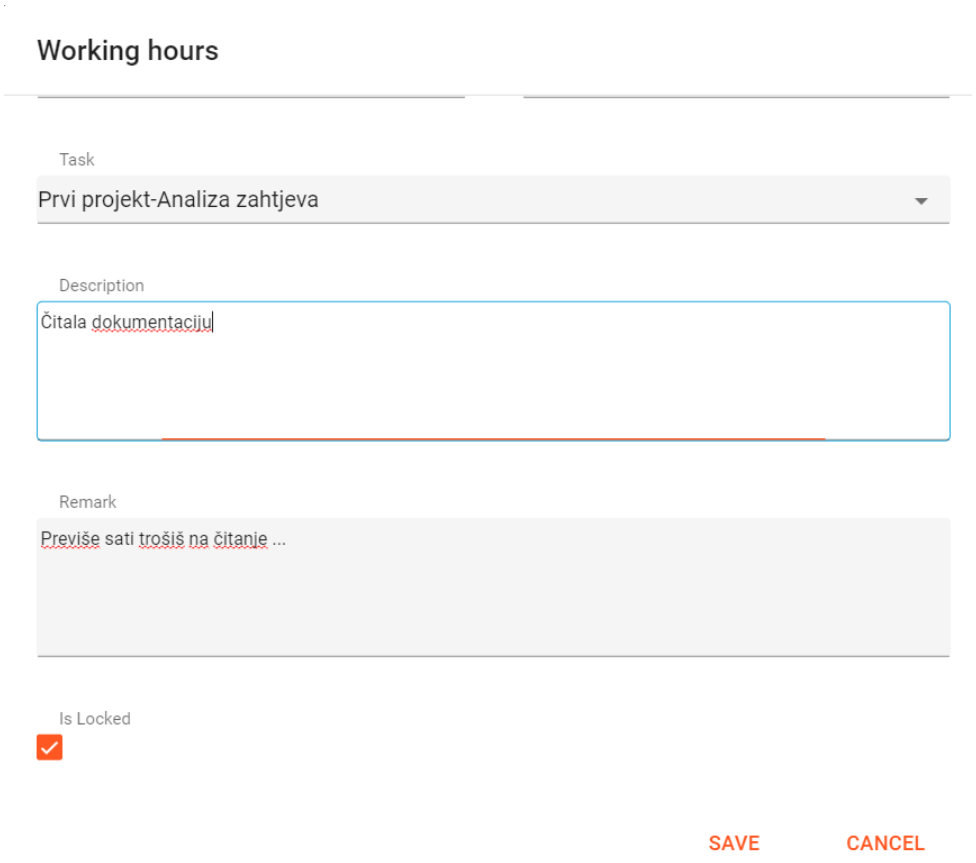
3.2.4. Kontrola sati

Stranica (Slika 3.18) za kontrolu unesenih sati od strane djelatnika otvara na poveznicu Control working hours. Voditelju se na odabir nude projekti na kojima ima dozvole administriranja, nakon toga odabire djelatnika na tom projektu, te godinu i mjesec. Pritiskom na gumb GET pokazat će mu se svi podaci koji zadovoljavaju postavljene kriterije.



Slika 3.18 - Stranica za kontrolu unesenih sati

Voditelj ima mogućnost ažurirati podatke (Slika 3.19) i napisati objašnjenje za učinjenu promjene (polje Remark) i ako želi zaključati dan (polje Is Locked). Zaključani dan djelatnik više ne može mijenjati.

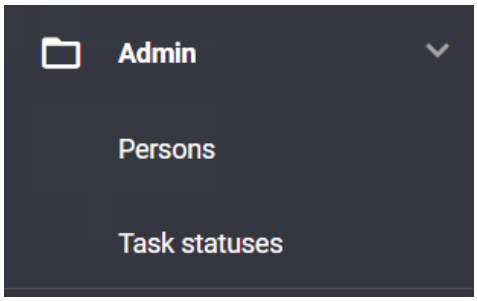


Slika 3.19 - Forma za ažuriranje unesenih sati

Voditelj ima mogućnost i zaključati cijeli mjesec za djelatnika na projektu, polje Locked na stranici (Slika 3.18) s dohvaćenim podacima. Time se zaključavaju svi dani u mjesecu i djelatnik više ne može unositi i mijenjati podatke za sate unesene za taj mjesec na tom projektu.

3.3. Stranice koje vidi Administrator

Prijavom korisnika koji ima ulogu Administratora sustava izbornik aplikacije se proširuje (Slika 3.20) za poveznicom Persons koja otvara stranicu za dodavanje novih osoba te poveznicom Task statuses koja otvara stranicu za definiranje statusa zadataka.

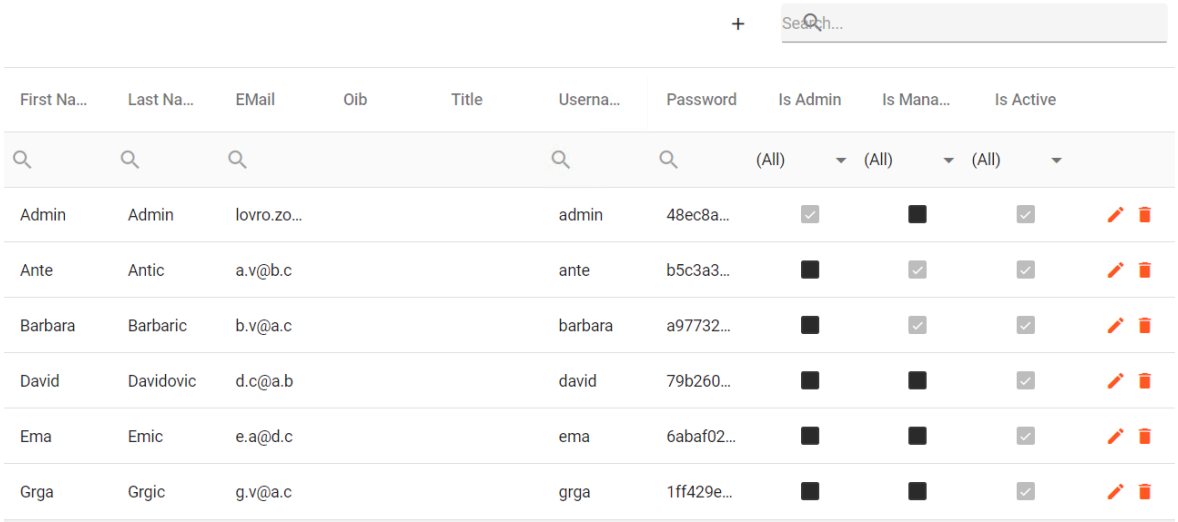


Slika 3.20 - Izbornik Administratora

3.3.1. Dodavanje korisnika

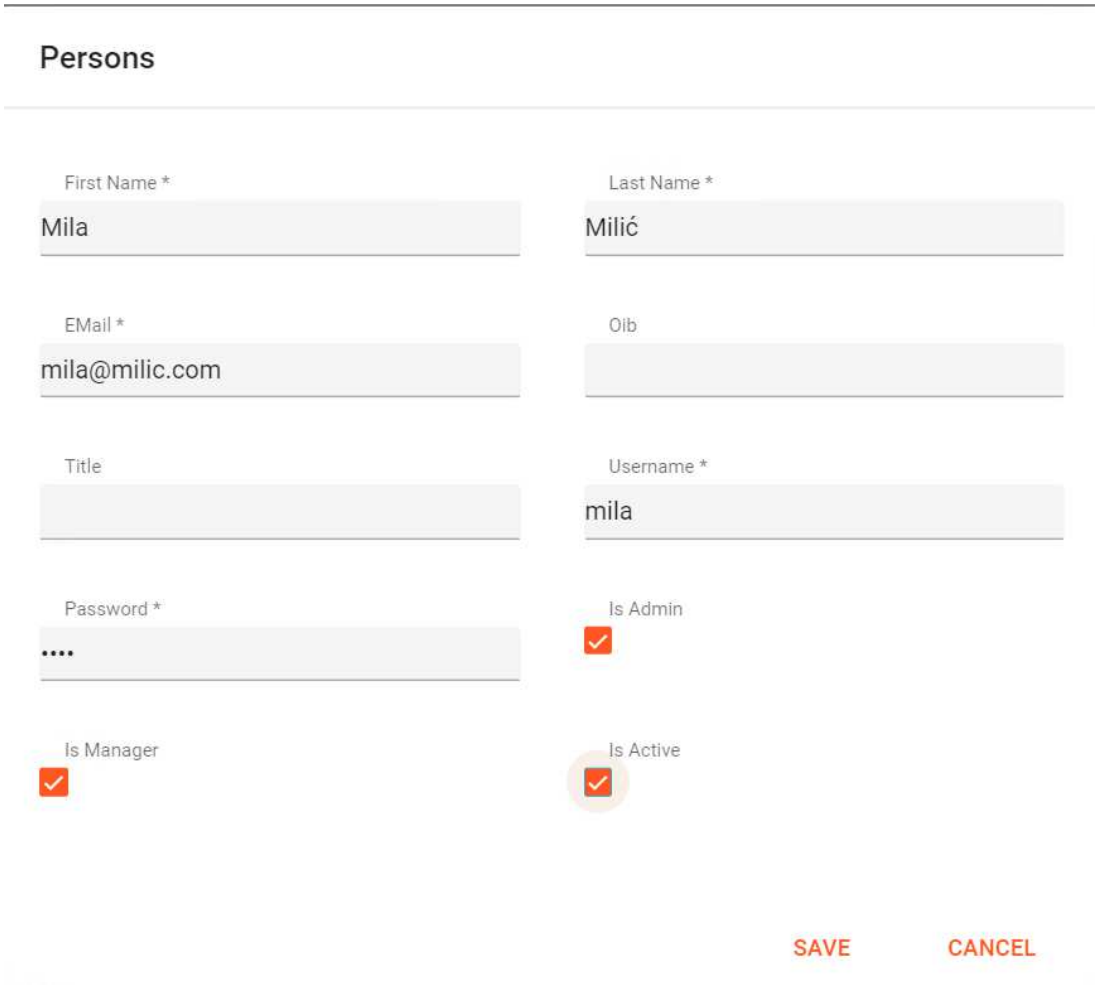
Stranica za dodavanje novih korisnika (Slika 3.21) sustava otvara se na poveznicu Persons.

Administratoru se prikazuju svi do tad uneseni korisnici, a omogućen mu je unos novih i ažuriranje podataka postojeći korisnicima.



Slika 3.21 - Stranica za dodavanje novog korisnika

Odabirom opcije za dodavanje novog korisnika (+) otvara se popup forma (Slika 3.22) u koju administrator unosi podatke i označavanjem polja Is Admin i Is Manager korisniku zapravo dodjeljuje ulogu Administratora i/ili Voditelja projekta. Ako niti jedna od te dvije opcije nije odabrana novi korisnik ima ulogu Djelatnik.



Slika 3.22 - Forma za dodavanje novog korisnika

4. Upute za pokretanje aplikacije

Trenutno ne postoji binarna izvršna datoteka koja bi korisnicima omogućila jednostavnu pokretanje i korištenje, nego je potrebno ručno pokrenuti Back-end aplikaciju (RESTfull servis) i Front-end aplikaciju (klijentska aplikacija), a pripadna baza podataka se kreira pomoću SQL skripte. U nastavku su napisane upute kako se aplikacija pokreće u razvojnom okruženju.

Preduvjeti za pokretanje Back-end aplikacije:

- instaliran DBMS Microsoft SQL Server i pokrenuta skripta za kreiranje baze koja se nalazi na Gitlabu
- .NET 6.0 Core framework
- alat za razvoj aplikacija Visual Studio 2022
- programski kod preuzet sa Gitlaba

Nakon što su osigurani preduvjeti programski kod je potrebno otvoriti u Visual Studio okruženju i pokrenuti. Datoteka launchSetting.json sadrži postavke URL aplikacije, a datoteka appsettings.json sadrži podatke za spajanje na bazu podataka. Tek nakon što je pokrenuta Back-end aplikacija moguće je pokrenuti klijentsku aplikaciju.

Preduvjeti za pokretanje Front-end aplikacije:

- instaliran Node.js (verzija 18.15.0 ili novija) i npm (verzija 9.5 ili novija)
- instaliran moderan web preglednik (npr. Chrome, Firefox, Edge, Safari, itd.)
- alat za razvoj aplikacija Visual Studio Code
- programski kod preuzet sa Gitlaba

Za pokretanje aplikacije potrebno je napraviti sljedeće:

- u Visual Studio Codu otvoriti direktorij s programskim kodom
- u terminal prozoru Visual Studio coda pokrenuti naredbu „npm install“
- u terminal prozoru Visual Studio coda pokrenuti naredbu „npm run serve“

Nakon toga ispisuje se URL na kojem je pokrenuta aplikacija, npr: http://localhost:8081/.

Zaključak

Uvođenje sustava za praćenje rada djelatnika na projektu može pozitivno utjecati na organizaciju samog projekta. Ako se to dobro izvede, može pomoći pri učinkovitosti, transparentnosti te produktivnosti na projektima. Cilj ovog završnog rada bio je napraviti programsku implementaciju koja to omogućava. Web aplikacija pomaže djelatnicima na projektu da učinkovito vode evidenciju radnog vremena i da jednostavno dolaze do informacije koji je njihov zadatak na dodijeljenom projektu. S druge strane voditelj projekta ima mogućnost definirati projekt i zadatke na projektu, pridružiti djelatnike projektima, zadavati im zadatke te kontrolirati i odobravati njihovu evidenciju radnog vremena. Implementirano rješenje ima i mogućnost razmjene poruka između djelatnika i voditelja na projektu koje se može iskoristiti i za prijavu pogrešaka. U okviru završnog zadatka implementirana je web aplikacija korištenjem Vue okruženja. Aplikacija pruža intuitivno sučelje kako djelatniku tako i voditelju te im tako olakšava posao unosa ili kontrole podataka. Podrška web aplikaciji je pozadinski servis s kojim aplikacija komunicira tako da mu šalje podatke ili od njega dobiva podatke. Pozadinski servis implementiran je kao RESTfull web servis preko kojega se komunicira s bazom podataka.

Sustav je otvoren za nadogradnje, a neke bi mogle biti:

- Dodavanja izvještaja
- Grafička vizualizacija podataka
- Unaprjeđenje prijave pogrešaka (praćenje bugova)
- Pohranjivanje dokumenata vezanih za projekt

Prilikom izrade programskog rješenja korištene su moderne tehnologije razvoja web aplikacija (Vue, RESTfull servis i dr.) koje su zahtijevale da proširim znanje stečeno na položenim predmetima.

Literatura

[1] Službena stranica Swagger UI-a. Poveznica: https://swagger.io/tools/swagger-ui; Pristupljeno 23. travnja 2023.

[2] Službena dokumentacija za Entity Framework Core. Poveznica: https://learn.microsoft.com/en-us/ef/core/; Pristupljeno 23. travnja 2023.

[3] Službena dokumentacija JWT-a. Poveznica: https://jwt.io/introduction; Pristupljeno 23. travnja 2023.

[4] Jay Thakkar, ECDSA vs RSA: Everything You Need to Know. Članak na internetu. Poveznica: https://sectigostore.com/blog/ecdsa-vs-rsa-everything-you-need-to-know/; Pristupljeno 23. travnja 2023.

[5] Službena stranica DevExpressa. Poveznica: https://www.devexpress.com/; Pristupljeno 23. travnja 2023.

[6] Službena stranica Vue.js-a. Poveznica: https://vuejs.org/; Pristupljeno 23. travnja 2023.

[7] Službena stranica Vuex-a. Poveznica: https://vuex.vuejs.org/; Pristupljeno 23. travnja 2023.

[8] Službena stranica biblioteke Pinia. Poveznica: https://pinia.vuejs.org/; Pristupljeno 23. travnja 2023.

[9] Službena stanica Axios-a. Poveznica: https://axios-http.com/; Pristupljeno 23. travnja 2023.

Sažetak

Naslov: Sustav za praćenje rada na projektu

Sažetak: U okviru ovog rada napravljena je web aplikacija i pozadinski servis za praćenje rada na projektu. Sustav razlikuje tri vrste korisnika: Administrator, Voditelj projekta i Djelatnik. Administrator je zadužen za unos korisnika sustava i održavanje tablice statusa zadataka. Voditelj projekata je zadužen za definiranje projekta, definiranje zadataka na projektu, stavljanje djelatnika na projekt, dodjeljivanje zadataka djelatnicima te kontroliranje i ažuriranje evidencije radnog vremena djelatnika na projektu. Djelatnik na projektu ima mogućnost evidencije radnog vremena po projektu i zadatku te komunikaciju s drugim djelatnicima na projektu. Programsko rješenje se sastoji od tri dijela: klijentske aplikacije, pozadinskih servis i baze podataka. Klijentska aplikacija implementirana je korištenjem Vue.js okruženja. Pozadinski servis je implementiran u C# jeziku kao RESTfull servis. Za bazu podataka korišten je Microsoft SQL Server Express.

Ključne riječi: web aplikacija, WEB API, VUE.js, DevExpress; C#, Evidencija radnog vremena, Microsoft SQL Server, Entity Framework Core.

Summary

Title: Project monitoring system

Summary: As part of this work, a web application and a backend service for project monitoring were developed. The system distinguishes three types of users: Administrator, Project Manager, and Employee. The Administrator is responsible for user input into the system and maintaining the task status table. The Project Manager is responsible for defining the project, defining tasks within the project, assigning employees to the project, assigning tasks to employees, and monitoring and updating the employee's working hours record on the project. The Employee on the project has the ability to record working hours per project and task, as well as communicate with other employees on the project. The software solution consists of three parts: a client application, a backend service, and a database. The client application is implemented using the Vue.js framework. The backend service is implemented in C# as a RESTful service. Microsoft SQL Server Express is used as the database.

Keywords: web application , WEB API, VUE.js, DevExpress; C#, Recording of working hours, Microsoft SQL Server, Entity Framework Core