

Web aplikacija za upravljanje ljudskim resursima i internim poslovnim procesima

Fehir, Karla

Master's thesis / Diplomski rad

2025

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:200:273876>

Rights / Prava: [In copyright](#)/[Zaštićeno autorskim pravom.](#)

Download date / Datum preuzimanja: **2025-03-14**

Repository / Repozitorij:

[Faculty of Electrical Engineering, Computer Science and Information Technology Osijek](#)



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

Sveučilišni studij

**Web aplikacija za upravljanje ljudskim resursima i internim
poslovnim procesima**

Diplomski rad

Karla Fehir

Osijek, 2024.

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Karla Fehir
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1282R, 07.10.2022.
JMBAG:	0165081529
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 diplomskog rada:	Web aplikacija za upravljanje ljudskim resursima i internim poslovnim procesima
Znanstvena grana diplomskog rada:	Informacijski sustavi (zn. polje računarstvo)
Zadatak diplomskog rada:	U diplomskom radu potrebno je opisati i analizirati sustav za upravljanje ljudskim resursima i internim poslovnim procesima. Treba predložiti model i programsku arhitekturu Web aplikacije s bazom podataka. Web aplikacija omogućuje analizu radnih performansi zaposlenika, digitalizaciju procesa regrutacije, naprednu analitiku plaća, administraciju godišnjih odmora i optimizaciju upravljanja izostancima. Tema rezervirana za: Karla Fehir
Datum ocjene pismenog dijela diplomskog rada od strane mentora:	21.01.2025.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	27.2.2025.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	27.02.2025.

**FERIT**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK****IZJAVA O IZVORNOSTI RADA**

Osijek, 27.02.2025.

Ime i prezime Pristupnika:

Karla Fehir

Studij:

Sveučilišni diplomski studij Računarstvo

Mat. br. Pristupnika, godina upisa:

D1282R, 07.10.2022.

Turnitin podudaranje [%]:

13

Ovom izjavom izjavljujem da je rad pod nazivom: **Web aplikacija za upravljanje ljudskim resursima i internim poslovnim procesima**

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 diplomskog rada.....	1
2. PREGLED PODRUČJA TEME	2
2.1. Workday.....	2
2.2. ADP Workforce Now	3
2.3. SAP SuccessFactors.....	3
2.4. Zoho People.....	4
3. IDEJNO RJEŠENJE I ARHITEKTURA WEB APLIKACIJE	6
3.1. Zahtjevi na web aplikaciju	6
3.1.1. Funkcionalni zahtjevi	6
3.1.2. Nefunkcionalni zahtjevi	6
3.2. Prijedlog arhitekture rješenja.....	7
3.3. Pregled korištenih tehnologija	8
3.3.1. Angular.....	8
3.3.2. .NET	9
3.3.3. Microsoft Sql.....	10
4. PROGRAMSKO RJEŠENJE	11
4.1. Postavljanje razvojne okoline	11
4.1.1. Angular.....	11
4.1.2. .NET	12
4.2. Povezivanje klijentske i poslužiteljske strane	13
4.3. Entity Framework i kreiranje baze podataka	14
4.4. Autentifikacija	15
4.5. Autorizacija.....	16
4.6. Upravljanje zaposlenicima	18
4.6.1. Dodavanje korisnika / zaposlenika.....	18
4.6.2. Pregled svih zaposlenika	19
4.6.3. Profil zaposlenika.....	21
4.6.4. Upravljanje plaćama.....	23
4.6.5. Administracija godišnjih odmora	24

4.7. Upravljanje poslovnim procesima	26
4.7.1. Upravljanje poslovnima	26
4.7.2. Upravljanje kandidatima	27
4.7.3. Upravljanje odjelima	29
4.7.4. Upravljanje projektima.....	30
5. KORIŠTENJE I ISPITIVANJE PROGRAMSKOG RJEŠENJA	32
5.1. Korisničko iskustvo Kandidata.....	32
5.2. Korisničko iskustvo uloge Admin	33
5.3. Korisničko iskustvo uloge Menadžer.....	38
5.4. Korisničko iskustvo uloge Zaposlenik.....	40
6. ZAKLJUČAK.....	42
LITERATURA	43
SAŽETAK.....	44
ABSTRACT	45

1. UVOD

U današnjem dinamičnom poslovnom okruženju, učinkovito upravljanje ljudskim resursima i poslovnim procesima ključno je za uspjeh organizacije. Tradicionalni pristupi upravljanju ljudskim resursima često su skloni pogreškama, što naglašava potrebu za digitalizacijom i automatizacijom ovih procesa.

Ovaj rad fokusira se na dizajn i implementaciju cjelovitog sustava za upravljanje ljudskim resursima i internim poslovnim procesima kroz razvoj web aplikacije. Predloženi sustav omogućit će organizacijama da poboljšaju svoje poslovanje analizom radnog učinka zaposlenika, digitalizacijom procesa zapošljavanja, naprednom analizom plaća, upravljanjem godišnjim odmorima i upravljanjem poslovnim procesima.

Web aplikacija osmišljena je kako bi korisnicima omogućila intuitivan i učinkovit način upravljanja svim aspektima ljudskih resursa. Predložena programska arhitektura osigurat će skalabilnost, sigurnost i pouzdanost sustava, omogućujući organizaciji da se prilagodi promjenama i rastu bez ugrožavanja funkcionalnosti.

Ovaj rad predstavlja detaljnu analizu modela sustava, funkcionalnosti, tehničkih specifikacija i poboljšanje ukupne učinkovitosti organizacije.

1.1. Zadatak diplomskog rada

U diplomskom radu potrebno je opisati i analizirati sustav za upravljanje ljudskim resursima i internim poslovnim procesima. Treba predložiti model i programsku arhitekturu Web aplikacije s bazom podataka. Web aplikacija omogućuje analizu radnih performansi zaposlenika, digitalizaciju procesa regrutacije, naprednu analitiku plaća, administraciju godišnjih odmora i optimizaciju upravljanja izostancima.

2. PREGLED PODRUČJA TEME

U modernom poslovnom okruženju upravljanje ljudskim resursima i internim poslovnim procesima ključno je za uspjeh organizacije. Mnoge web aplikacije već pružaju rješenja za te potrebe, integrirajući analitiku učinka zaposlenika, digitalizaciju procesa zapošljavanja, naprednu analitiku obračuna plaća, upravljanje godišnjim odmorima i optimizaciju upravljanja odsutnostima. U nastavku je opisano nekoliko aplikacija koje pružaju takve usluge i funkcionalnosti.

2.1. Workday

Workday je sveobuhvatna platforma za upravljanje ljudskim resursima i financijama namijenjena velikim poduzećima [1]. Njegova snažna arhitektura upravlja složenim HR procesima, uključujući praćenje učinka, upravljanje talentima, planiranje radne snage te ukupnu analitiku. Workday također nudi snažna rješenja za financijsko upravljanje, uključujući računovodstvo, proračun i financijsku analizu.

Aplikacija zaposlenicima nudi opcije za upravljanje osobnim podacima, zahtjevima za slobodne dane te druge HR funkcije pri čemu se smanjuje administrativno opterećenje.

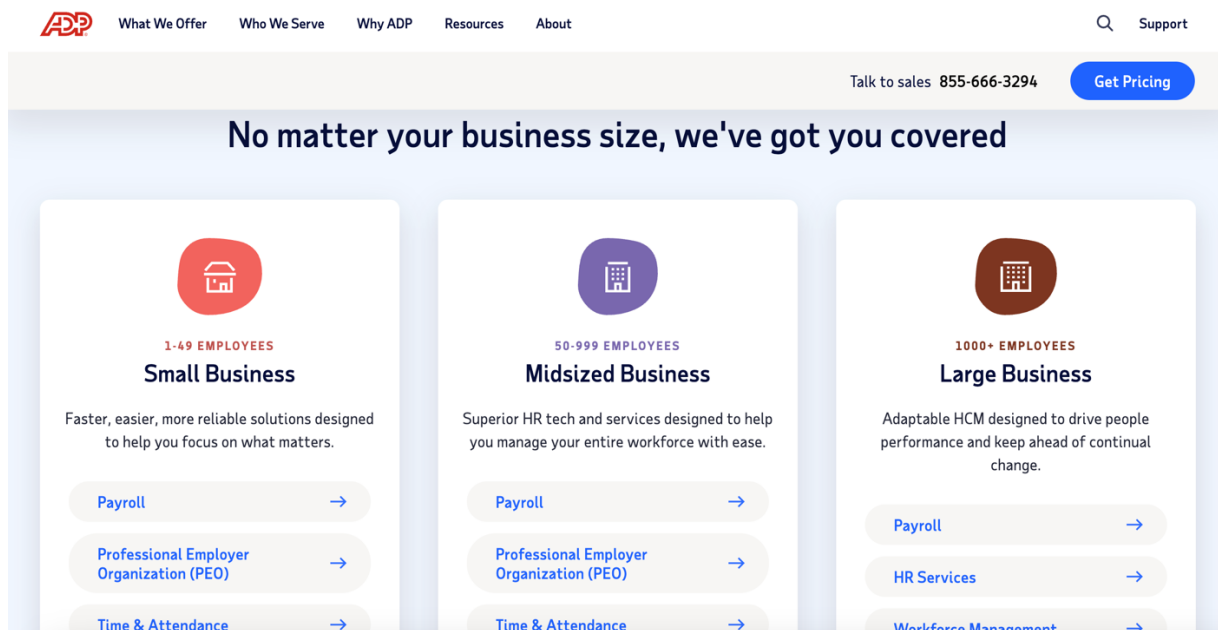
The image shows the Workday website interface. At the top, there is a navigation bar with the Workday logo, links for Products, Industries, Customers, Learn, Partners, and Company, and a 'Contact Sales' button. The main content area features a large heading: "How Workday keeps finance ahead of the curve." Below this, a sub-heading reads: "Read this report to see why FSN recognizes Workday as a leading strategic partner to help finance and IT future-proof their organization's finance function." A "Read Report" button is positioned below the text. To the right, a screenshot of the Workday financial dashboard is displayed, showing various charts and tables. The dashboard includes a donut chart for "Operating Expenses by Cost Center" with a total value of \$48,579,824, a bar chart for "Revenue per Headcount", and a table for "Monthly Reporting Binder". Below the dashboard, there are three promotional boxes: "Helping finance teams drive business forward." with a "Solutions for Finance" link, "A finance system made to deliver on your ROI." with a "Financial Management Overview" link, and "Drive value with a system built for modern finance leaders." with a "Read Guide" link and the Workday logo.

Slika 2.1. Sučelje aplikacije Workday

2.2. ADP Workforce Now

ADP Workforce Now je softver za upravljanje ljudskim resursima za srednje i velike tvrtke [2]. Pruža sveobuhvatne alate za upravljanje učinkom zaposlenika, zapošljavanjem, obračunom plaća, beneficijama te evidencijom radnog vremena. ADP Workforce Now također nudi napredne analitičke alate koji pomažu tvrtkama da bolje razumiju i upravljaju svojim ljudskim resursima putem detaljnih izvješća i uvida u podatke.

Također omogućuje zaposlenicima pristup osobnim podacima te praćenje evidencije radnog vremena. Uz to, pruža detaljne izvještaje i analitike za praćenje ključnih performansi.



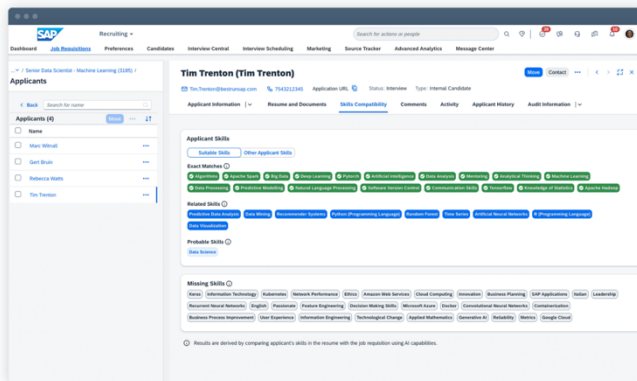
Slika 2.2. Sučelje aplikacije ADP Workforce Now

2.3. SAP SuccessFactors

SAP SuccessFactors je sveobuhvatan skup HR rješenja koja pomažu organizacijama da učinkovito upravljaju svojom radnom snagom [3]. Platforma uključuje module za upravljanje učinkom, zapošljavanje, planiranje nasljeđivanja i razvoj talenata, kao i obračun plaća i upravljanje vremenom. SAP SuccessFactors omogućuje tvrtkama da integriraju sve ključne HR procese u jednu platformu, olakšavajući upravljanje i poboljšavajući tijekom rada.

Nudi module koji omogućavaju organizacijama predviđanje i bolje upravljanje resursima.

Discover some of the latest innovations across SAP SuccessFactors HCM.



AI-assisted applicant screening helps recruiters find the right candidate quickly and fairly

Improve time to hire and candidate match by extracting and inferring applicant skills from resumes and matching them with the skills on a job requisition. Give recruiters insights into the matching, potential, and missing skills for every applicant within the applicant workbench.

[Watch the video >](#)

Enlarge

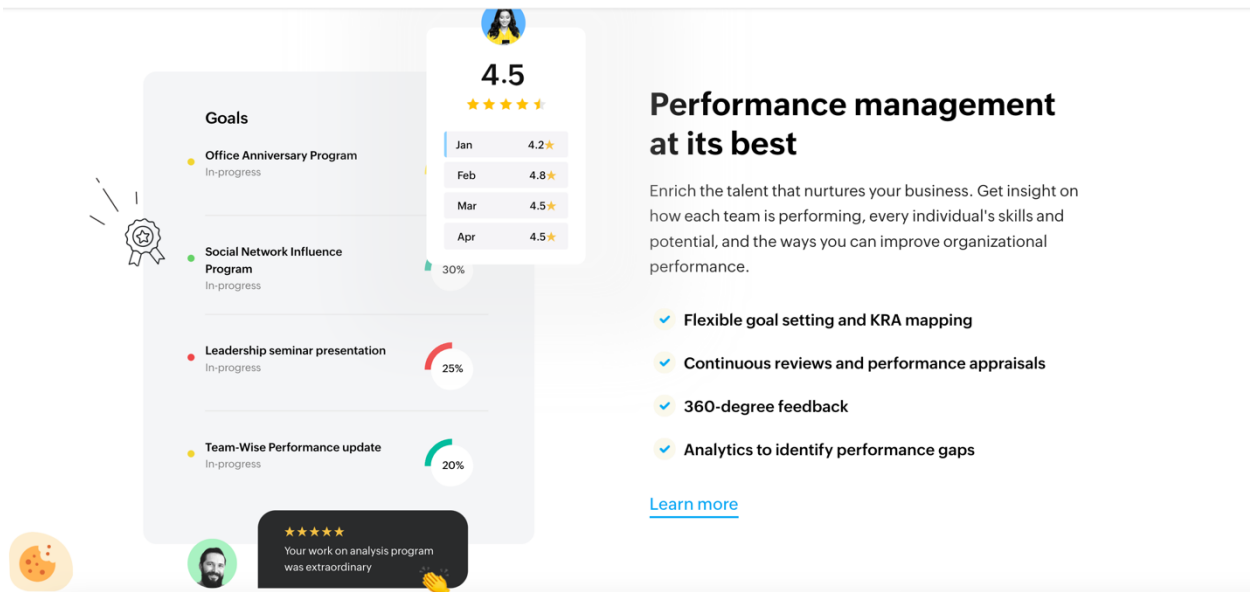
Contact us

Slika 2.3. Sučelje aplikacije SAP SuccessFactors

2.4. Zoho People

Zoho People je softver za upravljanje ljudskim resursima za tvrtke svih veličina [4]. Omogućuje značajke kao što su ocjenjivanje učinka, automatizacija zapošljavanja, upravljanje platnim listama te praćenje prisutnosti i odsutnosti zaposlenika. Zoho People omogućuje tvrtkama da digitaliziraju i automatiziraju HR procese, pomaže u povećanju učinkovitosti i smanjenju administrativnih zadataka.

Omogućuje zaposlenicima pristup osobnim podacima i praćenje ključnih performansi. Lako se integrira s ostalim Zoho programskim rješenjima i aplikacijama trećih strana što omogućuje povezivanje poslovnih procesa. Nudi automatizaciju pojedinih HR zadataka kao što su evidencija radnog vremena i odobravanje zahtjeva za slobodne dane.



Slika 2.4. Sučelje aplikacije Zoho People

3. IDEJNO RJEŠENJE I ARHITEKTURA WEB APLIKACIJE

U ovom poglavlju opisani su zahtjevi na web aplikaciju, arhitektura baze podataka te korištene tehnologije za razvoj web aplikacije.

3.1. Zahtjevi na web aplikaciju

U ovom poglavlju navedeni su i opisani svi ključni funkcionalni i nefunkcionalni zahtjevi za razvoj web aplikacije.

3.1.1. Funkcionalni zahtjevi

Web aplikacija za upravljanje ljudskim resursima pruža prilagođene funkcionalnosti za sljedeće kategorije korisnika: Administratore, Menadžere, Zaposlenike i Kandidate. Ovime se definiraju specifične potrebe korisnika i osigurava učinkovitost.

Administrator je primarni korisnik za upravljanje sustava koji osigurava kontrolu i nadzor sustava. Administratori imaju ovlasti upravljati korisnicima, konfigurirati postavke cijelog sustava i nadzirati aktivnosti unutar organizacije. Ključne značajke za administratore uključuju mogućnost stvaranja, ažuriranja i brisanja korisničkih računa te dodjele odgovarajućih uloga i dopuštenja. Time se osigurava da sustav odražava strukturu organizacije i sigurnosne protokole. Administratori također mogu definirati i upravljati obračunom plaća i zapošljavanjem.

Voditelj ima ključnu ulogu u nadgledanju odjela. Sustav pruža voditeljima alate za pojednostavljenje procesa zapošljavanja i upravljanje timskim rasporedima i odjelima. Voditelji imaju pristup značajkama vezanim uz zapošljavanje, kao što su objavljivanje ponuda za posao i pregled prijavljenih kandidata.

Zaposlenicima sustav pruža korisničko sučelje za obavljanje rutinskih zadataka i praćenje osobnih podataka. Zaposlenici mogu pregledavati svoje podatke na profilu.

Kandidati mogu pretraživati slobodna radna mjesta pomoću naprednih filtera i prijaviti se izravno putem aplikacije na način da ispune obrazac za prijavu.

3.1.2. Nefunkcionalni zahtjevi

Nefunkcionalni zahtjevi opisuju kako će web aplikacija raditi. Parametri koji ukazuju na kakav način bi sustav trebao raditi su performanse, pouzdanost, održivost te sigurnost.

Jedan od najvažnijih nefunkcionalnih zahtjeva ove aplikacije je **sigurnost**. Budući da aplikacija upravlja osjetljivim podacima kao što su osobni podaci zaposlenika, ugovori, plaće i povjerljivi dokumenti kandidata, potrebno je implementirati visoke sigurnosne standarde. Sustav mora

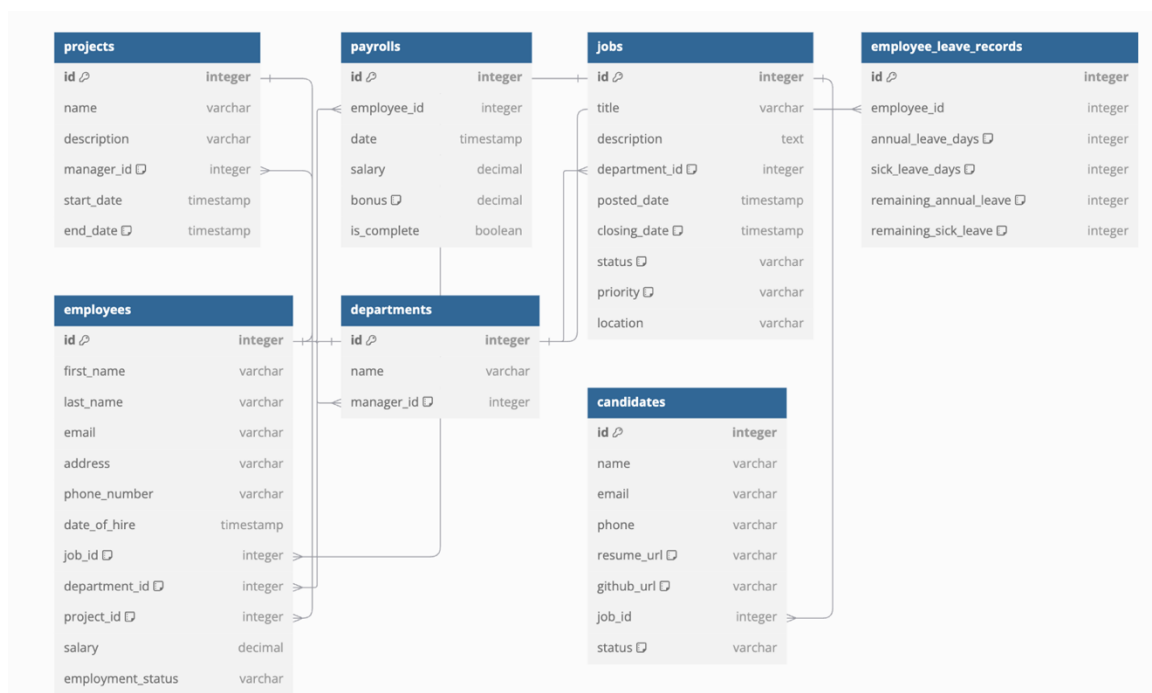
podržavati autentifikaciju i autorizaciju temeljenu na ulogama. To osigurava da svaki korisnik može pristupiti samo značajkama i podacima koji su relevantni za njegovu ulogu.

Kako bi aplikacija podržavala rastuće poslovne potrebe organizacija, mora osigurati visoku razinu **performansi i skalabilnosti**. Sustav bi trebao biti u stanju nositi se s velikim brojem korisnika bez značajnog smanjenja odziva. Performanse aplikacije mogu se poboljšati implementacijom asinkronih operacija. Sustav mora biti dizajniran da minimalizira prekide i osigura visoku dostupnost.

Kako bi korisnici mogli učinkovito koristiti aplikaciju, potrebno je osigurati intuitivno korisničko sučelje. Responzivni dizajn omogućuje korištenje aplikacije na različitim uređajima uključujući mobilne telefone i tablete.

3.2. Prijedlog arhitekture rješenja

Arhitektura aplikacije temelji se na relacijskoj bazi podataka koja uključuje tablice za zaposlenike, projekte, plaće, odjele, kandidate i radna mjesta. Na slici 3.1. prikazan je dijagram baze podataka.



Slika 3.1. Dijagram baze podataka

Tablica *employees* ključni je entitet baze, sadrži osnovne informacije o zaposlenicima, poput imena, prezimena, datuma zaposlenja, radnog mjesta i odjela kojem pripadaju. Veza sa projektima ostvaruje se putem stranog ključa *project_id*, dok se menadžer projekta evidentira u tablici *projects* putem polja *manager_id*, koje referencira tablicu zaposlenika.

Za vođenje evidencije plaća koristi se tablica *payrolls*, gdje su zapisani iznosi plaća, bonusi i statusi isplata za svakog zaposlenika. Ova tablica povezana je s tablicom zaposlenika preko polja *employee_id*. Podaci o odsutnosti zaposlenika, uključujući broj preostalih dana godišnjeg i bolovanja, pohranjuju se u tablici *employee_leave_records*, koja je također povezana s tablicom zaposlenika.

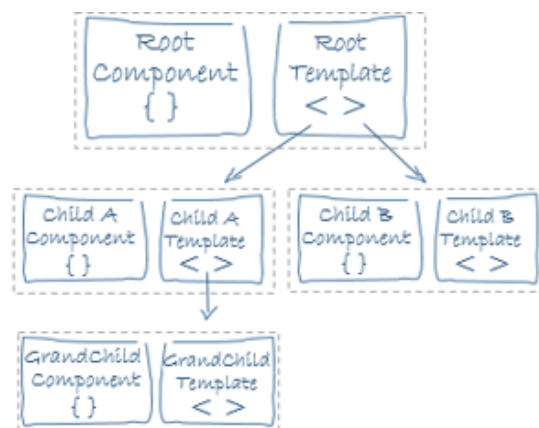
Organizacijska struktura poduzeća definirana je u tablici *departments*, u kojoj se bilježe podaci o odjelima, uključujući menadžera, povezanog putem polja *manager_id* s tablicom zaposlenika. Informacije o radnim mjestima evidentiraju se u tablici *jobs*, koja uključuje podatke poput naslova pozicije, opisa radnog mjesta i pripadnog odjela. Kandidati za radna mjesta pohranjuju se u tablici *candidates*, gdje se bilježe osnovni podaci kandidata te povezanost s oglasima za posao.

3.3. Pregled korištenih tehnologija

3.3.1. Angular

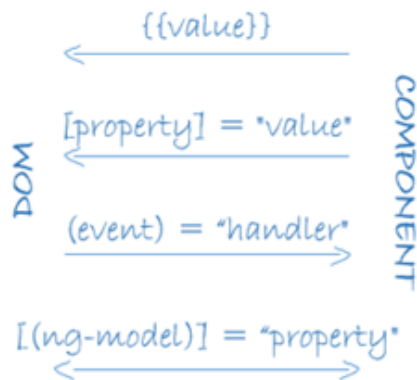
Angular (također poznat kao Angular 2) je okvir za izradu web aplikacija temeljen na TypeScriptu razvijen od strane Google-a [5]. Angular koristi arhitekturu temeljenu na komponentama, koja omogućuje izgradnju enkapsuliranih elemenata korisničkog sučelja koji se mogu ponovno koristiti. Svaka komponenta sadrži vlastiti HTML, CSS i TypeScript, što olakšava upravljanje i testiranje pojedinačnih dijelova aplikacije.

Prikaz (engl. *View*) komponente definiran je pratećim predloškom (engl. *Template*). Prikaz je oblik HTML-a koji govori Angularu kako prikazati komponentu. Prikazi su organizirani hijerarhijski, što omogućuje da se izmijene ili prikažu i sakriju cijeli dijelovi korisničkog sučelja prikazano na slici 3.2 [6].



Slika 3.2. Prikaz hijerarhijske strukture komponenti

Angular podržava dvosmjerno povezivanje podataka (engl. *two-way binding*), koje sinkronizira podatke između modela i prikaza prikazano na slici 3.3 [6]. Ovo osigurava da se sve promjene u pogledu automatski odražavaju na modelu i obrnuto.



Slika 3.3. Sinkronizacija podataka u Angular-u

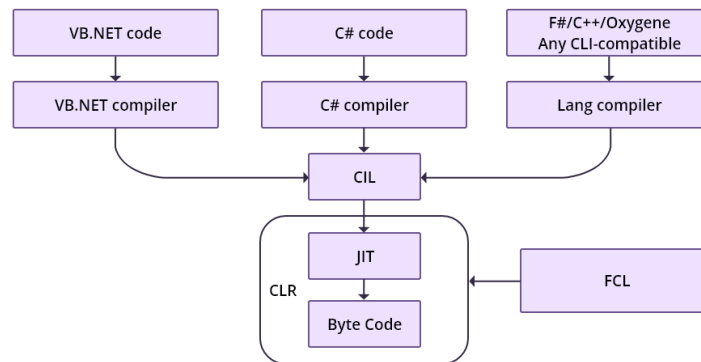
Angular CLI (engl. *Command Line Interface*) pruža skup alata za stvaranje, izgradnju, testiranje i implementaciju Angular aplikacija. Omogućuje brzo postavljanje aplikacije i pojednostavljuje zadatke tijekom razvoja.

3.3.2. .NET

.NET je platforma za razvoj softvera razvijena od strane Microsoft-a, dizajniran za širok raspon aplikacija, od desktop i mobilnih do web i *Cloud* rješenja [7]. Pruža biblioteke, alate i okruženja za izvođenje za stvaranje robusnih i skalabilnih aplikacija. .NET omogućuje razvoj aplikacija koje rade na Windows-u, macOS-u i Linuxu.

.NET podržava više programskih jezika, uključujući C#, F# i Visual Basic. C# se ističe kao jedan od najpopularnijih i najsvestranijih jezika. Također uključuje *runtime* poznat kao *Common Language Runtime* (CLR), koji omogućuje međuoperativnost aplikacija napisanih na različitim jezicima prikazan na slici 3.4.

HOW .NET LANGUAGES WORK



Slika 3.4. Prikaz Common Language Runtime u .NET-u

U središtu .NET-a nalazi se zbirka knjižnica nazvana *Base Class Library* (BCL), koja pruža unaprijed izgrađene funkcionalnosti za uobičajene programerske zadatke, poput rukovanja datotekama, povezivanja baze podataka i kriptografije. Osim BCL-a, .NET nudi specijalizirane okvire za različite vrste aplikacija, uključujući i Entity Framework za rad s bazama podataka koristeći objektno-relacijsko preslikavanje (ORM). .NET nudi sigurnost i pruža ugrađene mehanizme za autentifikaciju, autorizaciju i zaštitu podataka, kao što je podrška za HTTPS, enkripciju i protokole sigurnog pristupa.

3.3.3. Microsoft Sql

Microsoft SQL Server (MSSQL) je sustav za upravljanje relacijskim bazama podataka razvijen od strane Microsofta [8]. Dizajniran je za pohranu i upravljanje velikim količinama podataka te nudi visok stupanj performansi, sigurnosti i skalabilnosti.

MSSQL omogućuje definiranje, manipulaciju i dohvaćanje podataka koristeći jezik Transact-SQL (T-SQL). T-SQL pruža napredne funkcionalnosti poput kontrola toka i obrade pogrešaka. MSSQL podržava više razina sigurnosti, uključujući autentifikaciju korisnika, enkripciju podataka i kontrolu pristupa na razini tablica ili redaka.

SQL Server Management Studio (SSMS) nudi intuitivno sučelje za upravljanje bazom podataka, što uključuje pisanje upita, praćenje performansi i upravljanje sigurnosnim postavkama.

4. PROGRAMSKO RJEŠENJE

U ovom poglavlju predstavljeni su alati i tehnologije korišteni u izradi web aplikacije te programsko rješenje web aplikacije.

4.1. Postavljanje razvojne okoline

4.1.1. Angular

Za stvaranje Angular projekta potreban je Node.js, međuplatformsko JavaScript *runtime* okruženje koje omogućuje razvoj aplikacija na strani poslužitelja. Node.js dolazi s alatom *npm* (*Node Package Manager*), koji je potreban za upravljanje bibliotekama i paketima potrebnim za razvoj [9]. Također je potrebno instalirati Angular CLI (sučelje naredbenog retka) koji se instalira globalno pomoću *npm* naredbe *npm install -g @angular/cli*. Angular CLI pruža alate za učinkovito stvaranje, izgradnju i upravljanje Angular aplikacijama.

Za izradu novog Angular projekta koristi se naredba *ng new project-name*, koja inicijalizira strukturu Angular projekta. Struktura uključuje osnovne mape i datoteke potrebne za razvoj, poput direktorija *src* koji sadrži izvorni kod aplikacije, mape za komponente, *css* stilove i resurse, te osnovne datoteke kao što su *angular.json*, *package.json*, i *tsconfig.json*.

Početna generirana komponenta je *app.component*, koja predstavlja temeljnu komponentu aplikacije [10]. Služi i kao nadređena komponenta, u koju se integriraju sve ostale komponente i funkcionalnosti aplikacije.

Angular projekt temelji se na hijerarhijskoj strukturi komponenti, pri čemu svaka komponenta predstavlja izolirani dio korisničkog sučelja. Svaka komponenta sastoji se od tri ključne datoteke:

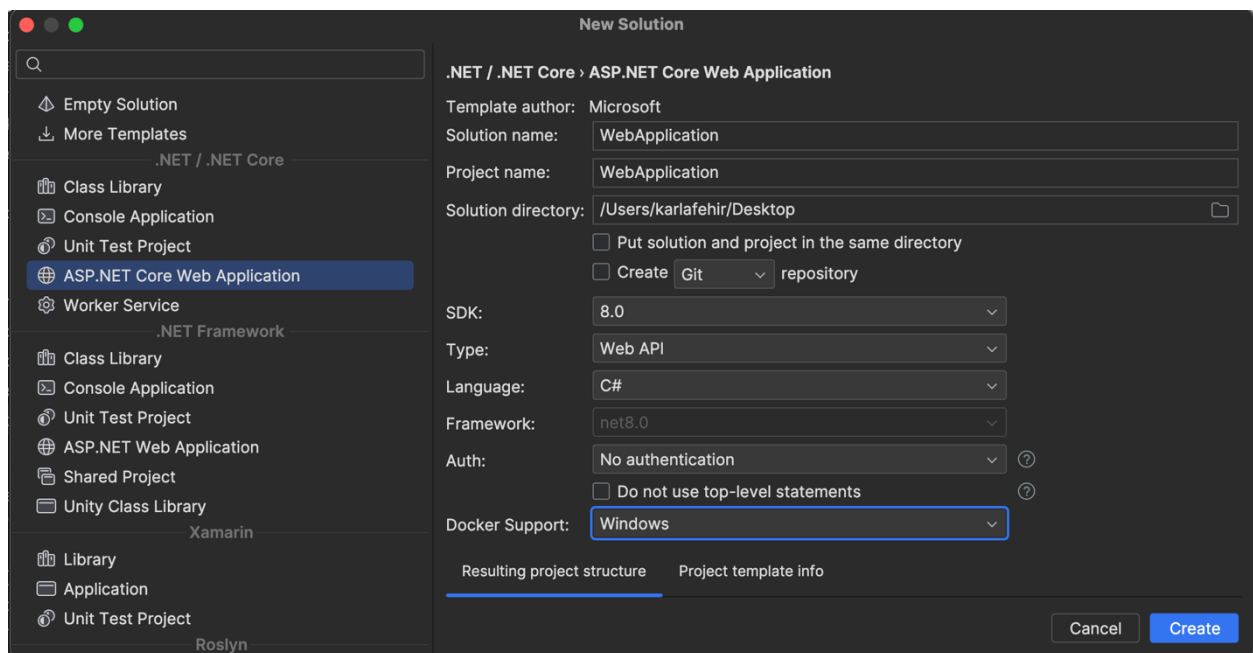
- **HTML datoteka:** Definiira izgled korisničkog sučelja komponente.
- **CSS/SCSS datoteka:** Sadrži stilove koji se primjenjuju na komponentu.
- **TypeScript datoteka:** Sadrži logiku i ponašanje komponente, uključujući metode, varijable i vezu s podacima (engl. *data binding*).

4.1.2. .NET

Za razvoj .NET Web API-ja potrebno je instalirati .NET 8 SDK-a. Za razvoj projekta koristi se Rider, integrirano razvojno okruženje (IDE) tvrtke JetBrains [11]. Korištenjem Riderovog čarobnjaka za projekte inicijalizira se ASP.NET Core Web API projekt, koji uključuje osnovnu strukturu potrebnu za izgradnju RESTful API-ja. Inicijalizacija Web API projekta prikazan je na slici 4.1.

Generirani projekt dolazi sa zadanim konfiguracijama i temeljnim elementima. Struktura projekta uključuje nekoliko ključnih direktorija i datoteka:

- **Controllers:** Sadrži kontrolere koji obrađuju HTTP zahtjeve i definiraju *endpointove*.
- **Models:** Koristi se za definiranje modela podataka koji predstavljaju entitete sustava.
- **Data:** Sadrži klasu ApplicationDbContext, koja služi za rad s bazom podataka pomoću Entity Frameworka.
- **Program.cs:** Datoteke koje sadrže konfiguraciju aplikacije, kao što su registracija servisa.



Slika 4.1. Inicijalizacija Projekta

4.2. Povezivanje klijentske i poslužiteljske strane

Nakon postavljanja klijentske aplikacije u Angularu i poslužiteljske aplikacije u .NET-u, potrebno je omogućiti komunikaciju između klijentske aplikacije i poslužiteljske aplikacije.

Unutar datoteke `launchSettings.json` .NET aplikacije definirana je konfiguracija za pokretanje aplikacije, koja uključuje `applicationUrl` koji određuje URL na kojem je API dostupan. Ova konfiguracija osigurava da se poslužitelj pravilno pokrene i da API bude dostupan klijentskoj aplikaciji. Konfiguracija poslužiteljske aplikacije prikazana je u primjeru koda 4.1.

```
{
  "$schema": "http://json.schemastore.org/launchsettings.json",
  "https": {
    "commandName": "Project",
    "dotnetRunMessages": true,
    "launchBrowser": true,
    "launchUrl": "swagger",
    "applicationUrl": "https://localhost:7209;http://localhost:5032",
    "environmentVariables": {
      "ASPNETCORE_ENVIRONMENT": "Development"
    }
  },
}
```

Primjer koda 4.1. Konfiguracija poslužitelja

Na poslužiteljskoj strani u `Program.cs` datoteci dodaje se CORS *middleware* u obradu zahtjeva kako bi se omogućio pristup iz Angular aplikacije [12]. *Middleware* provjerava svaki HTTP zahtjev i dopušta pristup samo s definirane liste dozvoljenih izvora. Primjer postavljanja CORS-a prikazan je u primjeru koda 4.2.

```
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAllOrigins", policy =>
    {
        policy.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader();
    });
});
```

Primjer koda 4.2. Prikaz CORS Middleware-a

U Angular aplikaciji, osnovni URL API-ja dodaje se u servis za komunikaciju s poslužiteljem kroz `HttpClientModule` prikazano u primjeru koda 4.3. Ova konfiguracija osigurava da klijentska aplikacija može slati zahtjeve poslužiteljskoj aplikaciji.

```

export class JobService {

    private apiUrl = 'http://localhost:5032';

    constructor(private http: HttpClient) { }

    getAllJobs(): Observable<Job[]> {
        return this.http.get<Job[]>(`${this.apiUrl}/api/Jobs/GetAllJobs`);
    }
}

```

Primjer koda 4.3. Slanja zahtjeva s klijentske aplikacije

4.3. Entity Framework i kreiranje baze podataka

Entity Framework Core korišten je kao alat za upravljanje bazom podataka koristeći *Code-first* pristup, što znači da su modeli definirani kao klase, a na temelju njih generirane su odgovarajuće tablice u bazi podataka [13]. Primjeri modela je klasa *Employee* prikazana u primjeru koda 4.4. Svaki model sadrži primarni ključ *Id*, dok ostala polja predstavljaju stupce u tablici baze podataka.

```

public class Employee
{
    [Key]
    public int Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Email { get; set; }
    public string Address { get; set; }
    public string PhoneNumber { get; set; }
    public DateTime DateOfHire { get; set; }

    //...
}

```

Primjer koda 4.4. Klasa Employee

Klasa *ApplicationDbContext* definira interakciju s bazom podataka. U klasi se definiraju tablice koje se pohranjuju u bazu i povezuju se s postojećim modelima. *ApplicationDbContext* predstavlja vezu s bazom i omogućuje operacije poput dodavanja, čitanja, ažuriranja i brisanja podataka. Također prati promjene na podacima i sprema ih u bazu te upravlja transakcijama. Na taj način osigurava da se sve operacije izvrše ispravno ili ponište u slučaju pogreške. Prikaz klase *ApplicationDbContext* nalazi se u primjeru koda 4.5.

```

public class ApplicationDbContext : IdentityDbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options) : base(options)
    {
    }

    public DbSet<Job> Jobs { get; set; }
    public DbSet<Candidate> Candidates { get; set; }
    public DbSet<Employee> Employees { get; set; }
    public DbSet<Department> Departments { get; set; }
    public DbSet<EmployeeLeaveRecord> EmployeeLeaveRecords { get; set; }
    public DbSet<Project> Projects { get; set; }
    public DbSet<Payroll> Payrolls { get; set; }
}

```

Primjer koda 4.5. Prikaz klase *ApplicationDbContext*

Za upravljanje promjenama u bazi podataka korišten je sustav **migracija**. Svaka promjena modela podataka, poput dodavanja novih polja ili tablica, zabilježena je kao migracija i primijenjena na bazu naredbom *dotnet ef database update*. Na taj način je omogućeno mijenjanje strukture baze bez ručnih izmjena.

Za bolju organizaciju koda primijenjen je **repozitorski** uzorak [14]. Za svaki model napravljen je odgovarajući repozitorij koji sadrži metode za dohvaćanje, dodavanje, izmjenu i brisanje podataka. Prvo se definira sučelje *IRepository* u kojem su definirane metode koje svaka repozitorski klasa mora implementirati. Sučelje *IRepository* prikazano je u primjeru koda 4.6.

```

public interface IRepository<T> where T : class
{
    Task<IEnumerable<T>> GetAllAsync(string? includeProperties = null);
    Task<T> GetByIdAsync(int id, string? includeProperties = null);
    Task AddAsync(T entity);
    void Update(T entity);
    void Delete(T entity);
}

```

Primjer koda 4.6. Prikaz *IRepository* sučelja

4.4. Autentifikacija

Autentifikacija korisnika izvedena je korištenjem ASP.NET Identity sustava, koji omogućuje siguran način prijave i registracije korisnika [15]. ASP.NET Identity koristi JSON Web Token (JWT) za autentifikaciju i autorizaciju korisnika. Nakon uspješne prijave, na poslužitelju se generira JWT token koji se šalje klijentskoj aplikaciji. Token sadrži ključne podatke o korisniku, poput njegovog identiteta i rola, a osiguran je kriptografskim potpisom kako bi se očuvao integritet podataka. Konfiguracija ASP.NET Identity sustava prikazana je u primjeru koda 4.7.

```

builder.Services.AddIdentity<IdentityUser, IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddDefaultTokenProviders();

builder.Services.AddAuthorization();

app.UseAuthentication();
app.UseAuthorization();

```

Primjer koda 4.7. Konfiguracija ASP.NET Identity sustava

Na klijentskoj strani koristi se servis *AuthService*, koji upravlja procesom autentifikacije. Ovaj servis omogućuje slanje zahtjeva za prijavu korisnika putem API-ja, dok se primljeni JWT token sprema u lokalnu pohranu preglednika za daljnje korištenje. Prikaz autentifikacije unutar servisa nalazi se u primjeru koda 4.8.

```

login(credentials: any): Observable<any> {
    return this.http.post<any>(`${this.apiUrl}/login`, credentials).pipe(
        tap(response => {
            if (isPlatformBrowser(this.platformId)) {
                localStorage.setItem('accessToken', response.accessToken);
                localStorage.setItem('refreshToken', response.refreshToken);
                this.loggedInSubject.next(true);
            }
        })
    );
}

```

Primjer koda 4.8. Autentifikacija unutar *AuthService*

4.5. Autorizacija

Upravljanje korisničkim pristupom u aplikaciji temelji se na sustavu rola. Role predstavljaju dozvole koje omogućuju korisnicima pristup određenim funkcionalnostima, kao što su administracija sustava ili pregled vlastitih podataka. U ovoj aplikaciji definirane su role *Administrator*, *Menadžer* i *Zaposlenik*. Svaka rola ima definirane privilegije, što osigurava kontrolu pristupa.

Kreiranje rola implementirano je korištenjem ASP.NET *Identity* sustava, gdje su role i korisnici međusobno povezani putem tablice *UserRoles*. Role su pohranjene u tablici *Roles*, dok tablica *UserRoles* omogućuje dodjeljivanje jedne ili više rola korisniku. Na ovaj način omogućeno je fleksibilno upravljanje dozvolama, na primjer, administrator ima pristup svim funkcionalnostima, dok zaposlenik može pristupiti samo svojim podacima. Role se definiraju i inicijaliziraju tijekom postavljanja baze podataka pomoću mehanizma za početno punjenje podataka (engl. *Seed Data*), što je prikazano u primjeru koda 4.9.

```

using (var scope = app.Services.CreateScope())
{
    var roleManager =
scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>();
    var userManager =
scope.ServiceProvider.GetRequiredService<UserManager<IdentityUser>>();
    await SeedRoles(roleManager);
    await SeedUsers(userManager, roleManager);
}

async Task SeedRoles(RoleManager<IdentityRole> roleManager)
{
    var roles = new[] { "Admin", "Manager", "Employee" };
    foreach (var role in roles)
    {
        if (!await roleManager.RoleExistsAsync(role))
        {
            await roleManager.CreateAsync(new IdentityRole(role));
        }
    }
}

```

Primjer koda 4.9. Postavljanje rola

Na razini API-ja, provjera korisničkih rola implementirana je pomoću atributa *[Authorize]* u ASP.NET-u. prikazano u primjeru koda 4.10.

```

[Authorize(Roles = "Manager")]
[HttpGet("GetAllProjects")]
public async Task<ActionResult<IEnumerable<Project>>> GetAllProjects()
{
    var projects = await _unitOfWork.Projects.GetAllAsync();
    return Ok(projects);
}

```

Primjer koda 4.10. Provjera rola pomoću atributa

Na klijentskoj strani koristi se logika za prilagodbu korisničkog sučelja ovisno o roli korisnika. Na primjer, opcije za dodavanje novih zaposlenika i poslova prikazuju se samo administratorima ili voditeljima odjela, dok zaposlenici mogu vidjeti samo opcije za pregled svojih podataka. Kontrola pristupa definirana je u *AuthGuard-u* u Angularu, prikazano u primjeru koda 4.11.

```

export class AuthGuard implements CanActivate {
  constructor(private authService: AuthService, private router: Router) {}

  canActivate(route: ActivatedRouteSnapshot): boolean {
    const role = this.authService.getRole();
    const routePath = route.routeConfig?.path;

    if (!this.authService.isLoggedIn()) {
      this.router.navigate(['login']);
      return false;
    }

    if (role === 'Admin') {
      return true;
    }

    if (role === 'Manager') {
      if (routePath === 'payroll' || routePath === 'departments') {
        this.router.navigate(['/']);
        return false;
      }
    }

    if (role === 'Employee') {
      const employeeId = this.authService.getEmployeeId();
      if (routePath === 'jobs') {
        this.router.navigate(['/employee-profile/${employeeId}']);
        return false;
      }
      if (employeeId && !routePath?.includes('employee-profile')) {
        this.router.navigate(['/employee-profile/${employeeId}']);
        return false;
      }
      return true;
    }

    return true;
  }
}

```

Primjer koda 4.11. Kontrola pristupa ovisno o stupnju autorizacije

4.6. Upravljanje zaposlenicima

4.6.1. Dodavanje korisnika / zaposlenika

Proces dodavanja novih korisnika započinje na klijentskoj strani gdje administrator unosi podatke putem forme. Forma uključuje polja za osnovne informacije, poput imena, prezimena, emaila, adrese, broja telefona, pozicije i odjela. Angular koristi reaktivne forme za provjeru unosa i

validaciju formata podataka.

Kada se pošalje zahtjev za dodavanje novog zaposlenika, podaci se šalju poslužiteljskoj strani gdje se provjerava ispravnost podataka i sprema ih se u bazu. Tijekom ovog procesa automatski se kreira korisnik u sustavu identiteta (*IdentityUser*), kojem se dodjeljuje početna rola *Zaposlenik*. Administrator može kasnije promijeniti rolu ako je potrebno.

Ako dodavanje korisnika uspije, ASP.NET *Identity* kreira korisnički račun s privremenom lozinkom, na primjer *Test123!*. Ako kreiranje korisničkog računa ne uspije, novi zaposlenik se briše iz baze. Dodavanje zaposlenika implementirano je metodom *AddEmployee*, koja koristi obrazac *UnitOfWork* za upravljanje podacima. Primjer metode za dodavanje zaposlenika prikazan je u primjeru koda 4.12.

```
[HttpPost("AddEmployee")]
public async Task<ActionResult<Employee>> AddEmployee(Employee employee)
{
    await _unitOfWork.Employees.AddAsync(employee);
    await _unitOfWork.SaveAsync();

    var identityUser = new IdentityUser
    {
        UserName = employee.Email,
        Email = employee.Email,
        EmailConfirmed = true
    };

    var predefinedPassword = "Test123!";
    var result = await _userManager.CreateAsync(identityUser,
predefinedPassword);

    if (!result.Succeeded)
    {
        _unitOfWork.Employees.Delete(employee);
        await _unitOfWork.SaveAsync();
        return BadRequest(result.Errors);
    }

    await _userManager.AddToRoleAsync(identityUser, "Employee");

    return CreatedAtAction(nameof(GetEmployeeById), new { id = employee.Id },
employee);
}
```

Primjer koda 4.12. Metoda za dodavanje zaposlenika

Kao dodatna sigurnosna mjera, početni administrator sustava automatski se dodaje u bazu podataka tijekom inicijalizacije aplikacije. Time se omogućuje nadzor i upravljanje korisnicima.

4.6.2. Pregled svih zaposlenika

Pregled zaposlenika obuhvaća funkcionalnosti za pregledavanje, uređivanje i brisanje podataka

unutar sustava. Na klijentskoj strani koristi se tablični prikaz podataka. Potrebno je prvo dohvatiti sve zaposlenike kako bi se mogli prikazati u tablici. Dohvaćanje podataka implementirano je u *Employee* servisu i *Employee* komponenti prikazano u primjerima koda 4.13. i 4.14.

```
getAllEmployees(): Observable<Employee[]> {  
  return this.http.get<Employee[]>(` ${this.apiUrl}/api/Employee/GetAllEmployees` );  
}
```

Primjer koda 4.13. Dohvaćanje svih zaposlenika u servisu

```
getAllEmployees(): void {  
  this.employeeService.getAllEmployees().subscribe(response => {  
    this.employees = response;  
  });  
}
```

Primjer koda 4.14. Pozivanje metode za dohvaćanje svih zaposlenika unutar komponente

Angular Material Table koristi se za prikaz zaposlenika, s podrškom za sortiranje, filtriranje i paginaciju. Tablica prikazuje osnovne informacije o zaposlenicima, poput imena, email adrese, rola i statusa zaposlenja. Korisnici s odgovarajućim dozvolama mogu odabrati zaposlenika i koristiti dodatne opcije, poput uređivanja podataka. *Angular Material Table* za zaposlenike prikazan je u primjeru koda 4.15.

```
<div class="employees-screen-wrapper" @fadeInAnimation>  
  <div class="employee-container">  
    <div class="filter-container">  
      <mat-form-field appearance="outline">  
        <mat-label>Search Employee</mat-label>  
        <input matInput [(ngModel)]="searchText" placeholder="Enter employee name">  
      </mat-form-field>  
      <div class="button-add" (click)="openDialog()">  
        + Add new employee  
      </div>  
    </div>  
  
    <table mat-table [dataSource]="getFilteredEmployees()">  
      <ng-container matColumnDef="name">  
        <th mat-header-cell *matHeaderCellDef> Name </th>  
        <td mat-cell *matCellDef="let employee" class="employee-name">  
          {{employee.firstName}} {{employee.lastName}} </td>  
      </ng-container>  
  
      <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
```

```

    <tr mat-row *matRowDef="let row; columns: displayedColumns;"
(click)="goToEmployeeProfile(row)">
    </tr>
</table>
</div>
</div>

```

Primjer koda 4.15. Angular Material Table za prikaz podataka o zaposlenicima

Osim toga, implementirana je funkcionalnost za pretraživanje zaposlenika kako bi korisnici mogli brzo pronaći traženog zaposlenika prema određenom kriteriju. Funkcija za pretraživanje zaposlenika prikazan je u primjeru koda 4.16.

```

getFilteredEmployees(): Employee[] {
  if (!this.searchText) {
    return this.employees;
  }
  return this.employees.filter(employee =>
    (employee.firstName + ' ' +
employee.lastName).toLowerCase().includes(this.searchText.toLowerCase())
  );
}

```

Primjer koda 4.16. Funkcija za pretraživanje zaposlenika u tablici

4.6.3. Profil zaposlenika

Klikom na zaposlenika u tablici prikazanoj u prethodnom poglavlju otvara se profil zaposlenika. Unutar *Angular Material* tablice dodana je funkcionalnost da se klikom na redak poziva *goToEmployeeProfile()* metoda, kojoj se prosljeđuje *Id* odabranog zaposlenika. Funkcionalnost je prikazana u primjerima koda 4.17. i 4.18.

```

<tr mat-row *matRowDef="let row; columns: displayedColumns;
(click)="goToEmployeeProfile(row)">

```

Primjer koda 4.17. Poziv metode i prosljeđivanje podataka

```

goToEmployeeProfile(employee: Employee): void {
  this.router.navigate(['employee-profile', employee.id]);
}

```

Primjer koda 4.18. Navigacija na profil zaposlenika

Na profilu su prikazane sve osnovne informacije o zaposleniku, a omogućeno je i uređivanje. Ova funkcionalnost omogućuje administratorima i ovlaštenim korisnicima ažurirati osnovne informacije o zaposlenicima, poput imena, prezimena, email adrese, kontakta, odjela, projekta, radnog mjesta i slobodnih dana.

Na klijentskoj strani koristi se prilagođena forma za prikaz i uređivanje podataka. Implementirana je pomoću reaktivnih formi u Angularu, što omogućuje validaciju podataka. Validacija uključuje provjeru formata adrese e-pošte, obaveznih polja i ograničenja dužine teksta u određenim poljima. Primjer validacije podataka prikazan je u primjeru koda 4.19.

```
<mat-form-field appearance="fill">
  <mat-label>Email</mat-label>
  <input matInput FormControlName="email" required>
  <mat-error *ngIf="employeeForm.get('email')?.hasError('email')">
    Invalid email format.
  </mat-error>
</mat-form-field>
```

Primjer koda 4.19: Validacija podataka unutar forme

Nakon uređivanja, promjene se spremaju slanjem zahtjeva prema poslužitelju. Na poslužiteljskoj strani obrađuje se zahtjev za ažuriranje, provjerava valjanost poslanih podataka te ažurira zapis u bazi. U primjeru koda 4.20. prikazano je ažuriranje korisnika unutar repozitorija, u primjeru koda 4.21. prikazano je ažuriranje korisnika unutar kontrolera.

```
public void Update(T entity)
{
  var entry = _context.Entry(entity);
  if (entry.State == EntityState.Detached)
  {
    _dbSet.Attach(entity);
  }

  entry.State = EntityState.Modified;
}
```

Primjer koda 4.20: Ažuriranje korisnika unutar repozitorija

```

[HttpPut("UpdateEmployee/{id}")]
public async Task<IActionResult> UpdateEmployee(int id, Employee employee)
{
    if (id != employee.Id)
    {
        return BadRequest("Employee ID mismatch.");
    }

    _unitOfWork.Employees.Update(employee);

    try
    {
        await _unitOfWork.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!(await EmployeeExists(id)))
        {
            return NotFound("Employee not found.");
        }
        else
        {
            throw;
        }
    }

    return NoContent();
}

```

Primjer koda 4.21: Ažuriranje zaposlenika unutar kontrolera

4.6.4. Upravljanje plaćama

Upravljanje plaćama omogućuje pregled, unos i ažuriranje podataka o plaćama zaposlenika. Svaki zaposlenik ima definiranu osnovnu plaću koja je spremljena u modelu *Employee*, a koja se koristi prilikom generiranja evidencije plaća. Za upravljanje plaćama koristi se *Payroll* model prikazan u primjeru koda 4.22.

```

public class Payroll
{
    [Key]
    public int Id { get; set; }
    public int EmployeeId { get; set; }
    public Employee Employee { get; set; }
    public DateTime Date { get; set; }
    public decimal Salary { get; set; }
    public decimal? Bonus { get; set; }
    public bool IsComplete { get; set; }
}

```

Primjer koda 4.22. Prikaz modela *Payroll*

Svaki mjesec generiraju se zapisi u tablici *Payroll* za sve postojeće zaposlenike. Generirani zapisi preuzimaju osnovnu plaću iz modela *Employee*, a dodaju se i podaci *EmployeeId*, *Bonus* i

IsComplete koji predstavlja status isplate. Generiranje plaća prikazano je u primjeru koda 4.23.

```
[HttpPost("GenerateMonthlyPayroll")]
public async Task<IActionResult> GenerateMonthlyPayroll()
{
    var currentDate = DateTime.UtcNow;

    // Check if payroll has already been generated for this month
    var existingPayrolls = await _unitOfWork.Payrolls
        .GetAllAsync(p => p.Date.Year == currentDate.Year && p.Date.Month ==
currentDate.Month);

    if (existingPayrolls.Any())
        return BadRequest("Payrolls for this month have already been
generated.");

    var employees = await _unitOfWork.Employees.GetAllAsync();
    if (!employees.Any())
        return NotFound("No employees found.");

    foreach (var employee in employees)
    {
        var payroll = new Payroll
        {
            EmployeeId = employee.Id,
            Date = currentDate,
            Salary = employee.Salary,
            Bonus = 0,
            IsComplete = false
        };

        await _unitOfWork.Payrolls.AddAsync(payroll);
    }

    await _unitOfWork.SaveAsync();

    return Ok();
}
```

Primjer koda 4.23. Metoda za generiranje plaća

Unutar aplikacije, plaće se prikazuju u tabličnom formatu kako bi administratori mogli pregledati trenutne podatke. Tablica omogućuje filtriranje prema imenu zaposlenika te uređivanje polja *Bonus* i *IsComplete*. Svaka izmjena plaće bilježi se u tablici *Payroll* što omogućuje praćenje promjena plaća zaposlenika kroz vrijeme. Ovaj pristup osigurava pregled svih povijesnih podataka o plaćama zaposlenika i pojednostavljuje proces upravljanja plaćama.

4.6.5. Administracija godišnjih odmora

Upravljanje odsutnostima zaposlenika omogućuje administratorima pregled i ažuriranje podataka o godišnjim odmorima, bolovanjima i neiskorištenim danima. Prilikom dodavanja novog zaposlenika, sustav automatski dodjeljuje 20 dana godišnjeg odmora i 10 dana bolovanja za tekuću godinu. Prikaz dodjeljivanja dana pri kreiranju korisnika nalazi se u primjeru koda 4.24.

```

[HttpPost("AddEmployee")]
public async Task<ActionResult<Employee>> AddEmployee(Employee employee)
{
    if (employee.EmployeeLeaveRecord == null)
    {
        employee.EmployeeLeaveRecord = new EmployeeLeaveRecord
        {
            AnnualLeaveDays = 20,
            SickLeaveDays = 10,
            RemainingAnnualLeave = 20,
            RemainingSickLeave = 10
        };
    }

    await _unitOfWork.Employees.AddAsync(employee);
    await _unitOfWork.SaveChangesAsync();

    //...
}

```

Primjer koda 4.24. Dodjeljivanje godišnjeg odmora pri kreiranju zaposlenika

Podaci o odsutnostima pohranjuju se u tablici *EmployeeLeaveRecord*, koja je povezana s modelom *Employee* putem stranog ključa (engl. *Foreign Key*). Ovo omogućuje direktno ažuriranje stanja odsutnosti zaposlenika kroz njegov profil u aplikaciji. Ažuriranje godišnjeg odmora prikazan je u primjeru koda 4.25.

```

if (employee.EmployeeLeaveRecord != null)
{
    if (existingEmployee.EmployeeLeaveRecord != null)
    {
        existingEmployee.EmployeeLeaveRecord.AnnualLeaveDays =
employee.EmployeeLeaveRecord.AnnualLeaveDays;
        existingEmployee.EmployeeLeaveRecord.SickLeaveDays =
employee.EmployeeLeaveRecord.SickLeaveDays;
        existingEmployee.EmployeeLeaveRecord.RemainingAnnualLeave =
employee.EmployeeLeaveRecord.RemainingAnnualLeave;
        existingEmployee.EmployeeLeaveRecord.RemainingSickLeave =
employee.EmployeeLeaveRecord.RemainingSickLeave;
    }
    else
    {
        existingEmployee.EmployeeLeaveRecord = new EmployeeLeaveRecord
        {
            AnnualLeaveDays = employee.EmployeeLeaveRecord.AnnualLeaveDays,
            SickLeaveDays = employee.EmployeeLeaveRecord.SickLeaveDays,
            RemainingAnnualLeave =
employee.EmployeeLeaveRecord.RemainingAnnualLeave,
            RemainingSickLeave =
employee.EmployeeLeaveRecord.RemainingSickLeave,
            EmployeeId = existingEmployee.Id
        };
    }
}

```

Primjer koda 4.25. Prikaz ažuriranja godišnjeg odmora zaposlenika

Unutar korisničkog sučelja, administrator može pregledati i ažurirati broj iskorištenih i preostalih dana godišnjeg odmora i bolovanja. Ove informacije su dostupne na profilu zaposlenika, gdje se mogu ažurirati slobodni dani. Osim pregleda kroz profil zaposlenika, prikaz svih zaposlenika i njihovih odsutnosti nalazi se u tablici koja omogućuje filtriranje prema imenu. Na ovaj način administratori mogu jednostavno upravljati podacima o odsutnostima za sve zaposlenike.

4.7. Upravljanje poslovnim procesima

4.7.1. Upravljanje poslovima

Upravljanje poslovima omogućuje kreiranje oglasa za nova radna mjesta, ažuriranje, brisanje oglasa te pregled prijavljenih kandidata. Proces započinje na klijentskoj strani gdje administrator ili menadžer kroz formu unosi detalje o poslu. Forma uključuje polja poput naziva posla, opisa, potrebnih kvalifikacija i roka za prijavu. Kod za kreiranje forme s pripadajućim validacijama prikazan je u primjeru koda 4.26.

```
this.jobForm = this.fb.group({
  title: [data?.title || '', Validators.required],
  description: [data?.description || '', Validators.required],
  departmentId: [data?.departmentId || '', Validators.required],
  postedDate: [data?.postedDate || new Date().toISOString().split('T')[0],
  Validators.required],
  closingDate: [data?.closingDate || ''],
  priority: [data?.priority || 2, Validators.required],
});
```

Primjer koda 4.26. Podaci potrebni za kreiranje oglasa za posao

Kako bi se omogućio odabir odjela prilikom kreiranja posla, potrebno je dohvatiti sve odjele pozivom metode `getAllDepartments()` prikazano u primjeru koda 4.27. Ova metoda dohvaća popis odjela s poslužitelja, koji se zatim koristi za popunjavanje padajućeg u formi za unos posla.

```
loadDepartments(): void {
  this.departmentService.getAllDepartments().subscribe(
    (departments) => {
      this.departmentOptions = departments;
    },
    (error) => {
      console.error('Error loading departments:', error);
    }
  );
}
```

Primjer koda 4.27. Dohvaćanje odjela i popunjavanje izbornika unutar forme

Nakon unosa podataka, šalje se zahtjev prema poslužiteljskoj strani gdje se podaci provjeravaju i pohranjuju u bazu u tablicu *Jobs*. Nakon što su poslovi kreirani, korisnici ih mogu pregledavati kroz korisničko sučelje.

4.7.2. Upravljanje kandidatima

Pregled kandidata omogućuje administratorima i menadžerima uvid u prijave za pojedine poslove i upravljanje procesom selekcije. Osobama koje nisu prijavljene u sustav također je omogućeno pregledavanje dostupnih poslova. Na ovaj način osigurano je jednostavno i digitalizirano rješenje za prijave na posao. Nakon što pregledaju ponuđene poslove, klikom na gumb *Prijavi se* otvara se forma u koju kandidati unose osnovne podatke: ime, email, broj mobitela, *GitHub* profil te životopis u *.pdf* formatu. Formi se automatski prosljeđuje *jobId* odabranog posla.

Kada kandidat popuni formu i pošalje prijavu, podaci se spremaju u tablicu *Candidates*. Kreira se zapis za kandidata kojem se postavlja početni status *New Applied*. Istovremeno, životopis kandidata sprema se na server radi kasnijeg pregleda. Životopis se sprema na način da se prvo provjerava i pohranjuje u direktorij *wwwroot/resumes*, gdje se generira jedinstveno ime datoteke pomoću *GUID*-a. Nakon što se datoteka uspješno spremi, kreira se *URL* do životopisa koji se sprema u bazu podataka kao dio podataka o kandidatu. Na ovaj način osigurava se sigurna pohrana i jednostavan pristup dokumentima tijekom procesa selekcije. Funkcija za dodavanje kandidata i spremanje životopisa nalazi se u primjeru koda 4.28.

```

[HttpPost("AddCandidateWithFile")]
public async Task<ActionResult<Candidate>> AddCandidateWithFile([FromForm]
Candidate candidate, IFormFile resumeFile)
{
    if (resumeFile != null && resumeFile.Length > 0)
    {
        var uploadPath = Path.Combine("wwwroot", "resumes");
        if (!Directory.Exists(uploadPath))
            Directory.CreateDirectory(uploadPath);

        var fileName = Guid.NewGuid().ToString() +
Path.GetExtension(resumeFile.FileName);
        var filePath = Path.Combine(uploadPath, fileName);

        using (var stream = new FileStream(filePath, FileMode.Create))
        {
            await resumeFile.CopyToAsync(stream);
        }

        var baseUrl = $"{Request.Scheme}://{Request.Host}";
        candidate.ResumeUrl = $"{baseUrl}/resumes/{fileName}";
    }

    await _unitOfWork.Candidates.AddAsync(candidate);
    await _unitOfWork.SaveAsync();
    return CreatedAtAction(nameof(GetCandidateById), new { id = candidate.Id
}, candidate);
}

```

Primjer koda 4.28. Prikaz funkcije za dodavanje kandidata i spremanje životopisa

Podaci o prijavljenim kandidatima prikazuju se kao kartice koje sadrže osnovne informacije o kandidatu. Klikom na određenu karticu moguće je pregledati priložene dokumente kandidata, poput životopisa i poveznice na *Github* profil kandidata. Funkcija *viewResume()* prikazana u primjeru koda 4.29. dohvaća *URL* životopisa kandidata koji je spremljen u bazi podataka u polju *resumeUrl*. Ako *URL* već započinje s *http*, koristi ga direktno, ako ne, dodaje osnovni *URL* servera kako bi stvorila punu putanju do datoteke na serveru. Nakon toga, *URL* se otvara u novom prozoru preglednika kako bi korisnik mogao pregledati životopis. Ako životopis nije priložen, funkcija prikazuje poruku upozorenja.

```

viewResume(candidate: Candidate): void {
    const baseUrl = 'http://localhost:5032';

    if (candidate.resumeUrl) {
        const fullUrl = candidate.resumeUrl.startsWith('http')
            ? candidate.resumeUrl
            : `${baseUrl}${candidate.resumeUrl}`;
        window.open(fullUrl, '_blank');
    } else {
        this.snackBar.open('No resume file uploaded for this candidate.', 'Close', {
            duration: 3000,
            verticalPosition: 'top',
        });
    }
}
}

```

Primjer koda 4.29. Funkcija za prikaz životopisa

Osim pregleda, administratori i menadžeri mogu ažurirati status prijave kandidata (npr. *Intervju*, *Onboarding*, *Zaposlen*, *Odbijen*) čime se olakšava praćenje kandidata kroz različite faze zapošljavanja.

4.7.3. Upravljanje odjelima

Upravljanje odjelima omogućuje administratorima dodavanje, uređivanje i pregled odjela unutar sustava. Administrator može dodati novi odjel putem forme na klijentskoj strani. Forma sadrži polja za unos naziva odjela i odabir menadžera odjela. Menadžer se bira iz padajućeg izbornika koji prikazuje samo korisnike s dodijeljenom rolom *Menadžer*. Funkcija unutar repozitorija *EmployeeRepository* za dohvaćanje zaposlenika ovisno o roli prikazana je u primjeru koda 4.30.

```

public List<EmployeeWithRoleDto> GetEmployeesWithRoles(string roleName =
null)
{
    var query = from e in _context.Employees
                join u in _context.Users on e.Email equals u.Email
                join ur in _context.UserRoles on u.Id equals ur.UserId
                join r in _context.Roles on ur.RoleId equals r.Id
                where roleName == null || r.Name == roleName
                select new EmployeeWithRoleDto
                {
                    Id = e.Id,
                    FirstName = e.FirstName,
                    LastName = e.LastName,
                    Email = e.Email,
                    Role = r.Name
                };

    return query.ToList();
}

```

Primjer koda 4.30. Dohvaćanje zaposlenika ovisno o roli

Kako bi se omogućio odabir menadžera odjela unutar forme, potrebno je dohvatiti sve menadžere pozivom metode *GetEmployeesWithRoleManager()* te popuniti padajući izbornik u formi. Metoda *GetEmployeesWithRoleManager()* prikazana je u primjeru koda 4.31.

```
GetEmployeesWithRoleManager() {
    this.employeeService.GetEmployeesWithRoles("Manager").subscribe(
        (response) => {
            this.managers = response;
        },
        (error) => {
            console.error('Error fetching managers:', error);
        }
    );
}
```

Primjer koda 4.31. Prikaz metode *GetEmployeesWithRoleManager()*

Nakon popunjavanja forme, podaci se šalju prema poslužitelju, gdje se validiraju i pohranjuju u bazu u tablicu *Departments*. Novi odjel automatski dobiva praznu listu zaposlenika, koja se kasnije ažurira dodavanjem ili uklanjanjem zaposlenika iz odjela. Pregled odjela omogućuje administratorima i menadžerima uvid u osnovne informacije, poput naziva odjela, menadžera i trenutnog broja zaposlenika. Osim pregleda, administrator može ažurirati podatke o odjelu ili obrisati odjel.

4.7.4. Upravljanje projektima

Upravljanje projektima omogućuje administratorima i menadžerima organizaciju i praćenje radnih zadataka unutar aplikacije. Projekti povezuju zaposlenike s određenim zadacima.

Stranica za upravljanje projektima pruža pregled svih postojećih projekata kroz kartični prikaz. Svaka kartica prikazuje osnovne informacije o projektu, poput naziva, opisa i datuma početka i završetka. Klikom na karticu otvara se detaljan prikaz projekta, gdje je vidljiv i popis zaposlenika na projektu. Metoda za dohvaćanje svih zaposlenika na projektu prikazana je u primjeru koda 4.32.

```
public async Task<IEnumerable<Employee>> GetEmployeesByProjectId(int
projectId)
{
    return await _context.Employees
        .Where(e => e.ProjectId == projectId)
        .ToListAsync();
}
```

Primjer koda 4.32. Metoda za dohvaćanje zaposlenika na projektu

Kod dodavanja novog ili ažuriranja postojećeg zaposlenika, moguće je dodijeliti ili ažurirati projekt zaposlenika. Prilikom učitavanje forme za dodavanje odnosno ažuriranja zaposlenika poziva se metoda *loadProjects()* koja popunjava padajući izbornik za odabir projekata prikazana u primjeru koda 4.33.

```
loadProjects() {
  this.projectService.getAllProjects().subscribe(
    (projects) => {
      this.projects = projects;
    },
    (error) => {
      console.error('Error fetching projects:', error);
    }
  );
}
```

Primjer koda 4.33. Prikaz metode za dohvaćanje postojećih projekata

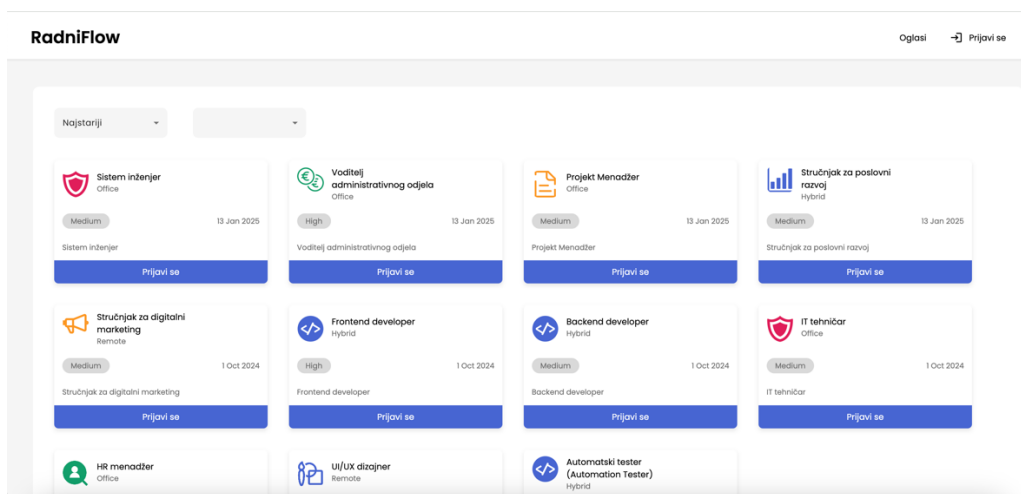
Menadžerima i administratorima omogućeno je dodavanje novih projekata te uređivanje postojećih. Dodavanje projekta obavlja se putem forme, gdje se unose ključni podaci poput naziva, opisa, voditelja projekta te datuma početka i završetka. Nakon spremanja, projekt se automatski prikazuje na stranici s projektima, a zaposlenicima se omogućuje dodjela u novi projekt.

5. KORIŠTENJE I ISPITIVANJE PROGRAMSKOG RJEŠENJA

U ovom poglavlju prikazan je rad web aplikacije za upravljanje ljudskim resursima i poslovnim procesima. Korisničko iskustvo web aplikacije ovisi o prethodno opisanim ulogama.

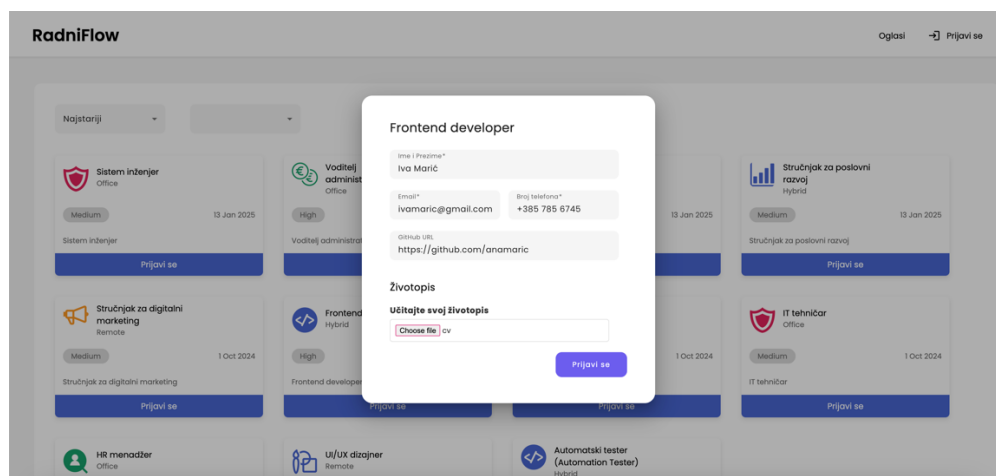
5.1. Korisničko iskustvo Kandidata

Početni zaslon aplikacije prikazuje aktivne oglase za posao u obliku kartica i navigacijsku traku. Na navigacijskoj traci nalazi se gumb *Prijavi se* koji omogućuje korisnicima prijavu u sustav ukoliko imaju postojeći račun. Početni zaslon aplikacije prikazan je na slici 5.1.



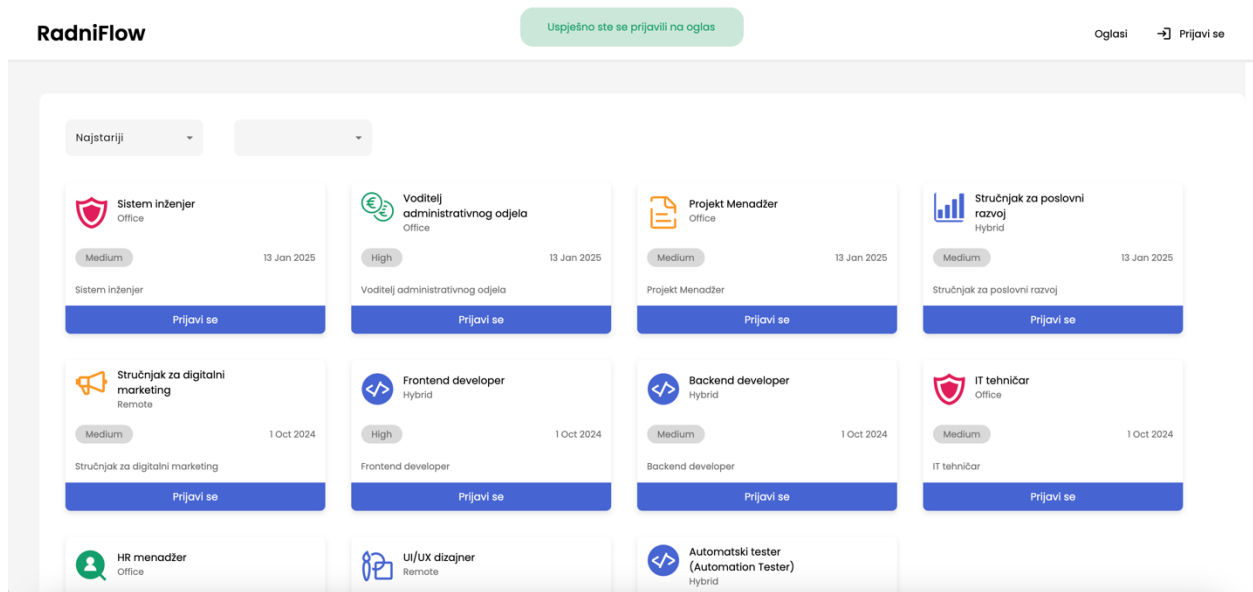
Slika 5.1. Prikaz početnog zaslona aplikacije

Svaka kartica sadrži osnovne informacije o poslu i gumb *Prijavi se*. Klikom na gumb otvara se forma za prijavu. Kandidati u formu unose svoje ime, prezime, email, telefon, poveznicu na *GitHub* profil te životopis u *PDF* formatu. Forma za prijavu kandidata na posao prikazana je na slici 5.2.



Slika 5.2. Prikaz forme za prijavu na oglas

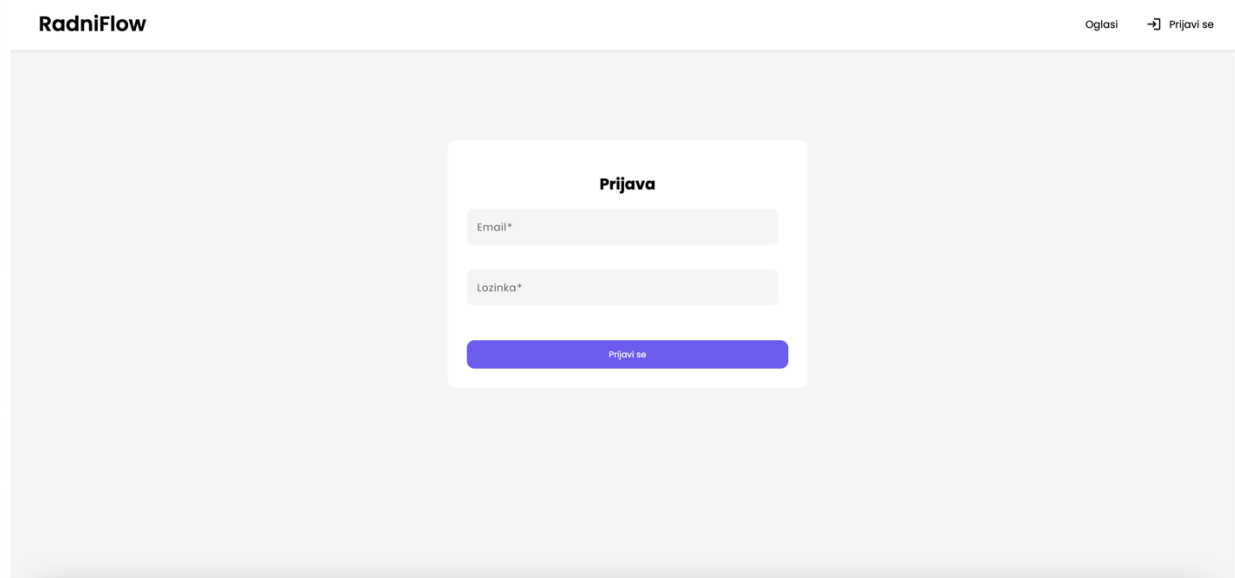
Nakon što se forma uspješno ispuni, prikazuje se obavijest *Uspješno ste se prijavili na oglas* prikazana na slici 5.3. Podaci kandidata automatski se spremaju u sustav, gdje ih administrator i menadžeri kasnije mogu pregledati, zajedno s priloženim dokumentima.



Slika 5.3. Prikaz obavijesti

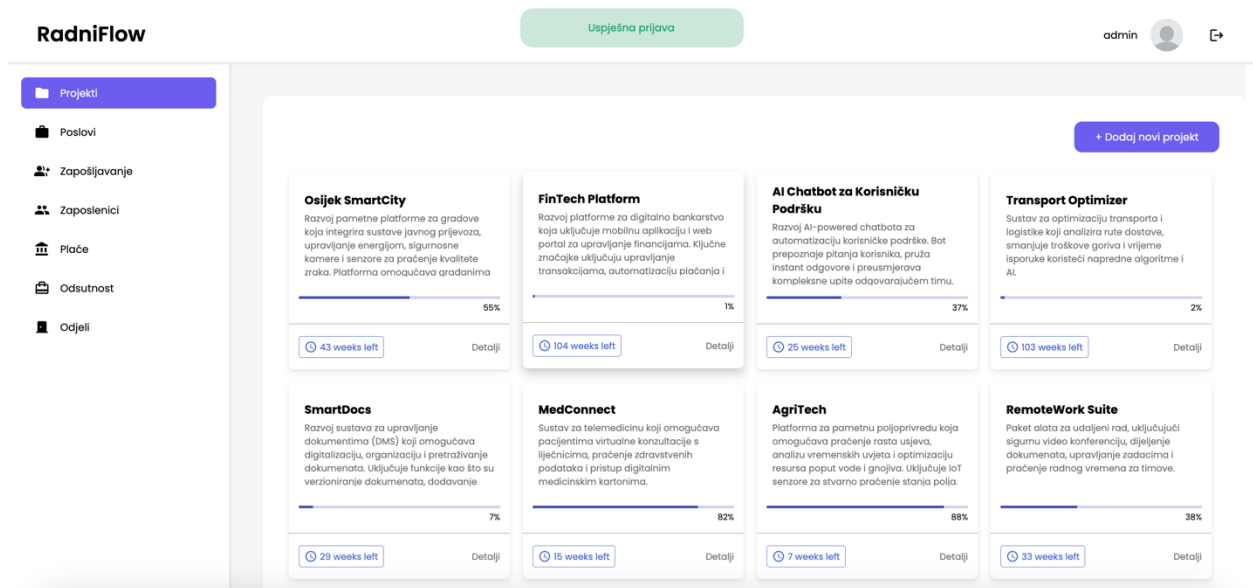
5.2. Korisničko iskustvo uloge Admin

Na navigacijskog traci početnog zaslona, klikom na gumb *Prijavi se* otvara se forma za prijavu korisnika u sustav prikazana na slici 5.4. Za prijavu je potrebno unijeti adresu e-pošte i lozinku. Ukoliko korisnik unese neispravan format adrese e-pošte prikazuje se poruka o grešci.



Slika 5.4. Prikaz zaslona za prijavu korisnika

Nakon što se administrator prijavi, otvara se sučelje prilagođeno njegovim ovlastima prikazano na slici 5.5. Na desnoj strani nalazi se izbornik s poveznicama na projekte, poslove, zapošljavanje, zaposlenike, plaće, odsustva i odjele.



Slika 5.5. Prikaz sučelja administratora

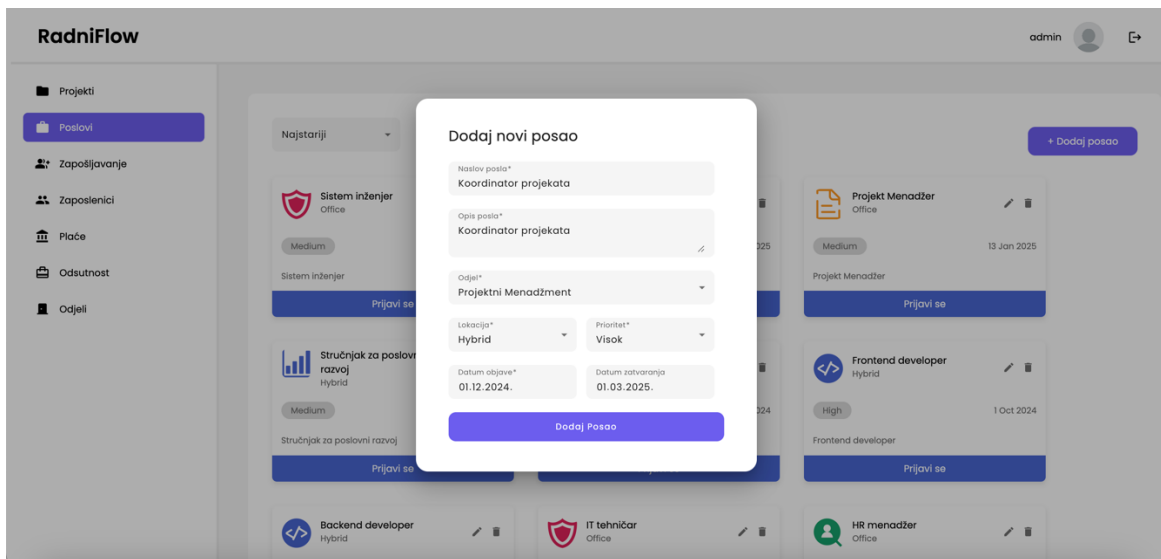
Administrator prvo kreira odjele tako da unosi naziv odjela i odabire menadžera koji će ga voditi. Kreirani odjeli prikazani su u tablici u kojoj administrator ima mogućnost pregleda i pretraživanja odjela. Klikom na odjel u tablici otvara se prikaz detalja odjela te opcije za ažuriranje i brisanje odjela. Tablični prikaz odjela nalazi se na slici 5.6.

The dashboard displays a table of departments. The table has the following columns: Naziv, Menadžer, Menadžer Kontakt, and Broj zaposlenika. The data is as follows:

Naziv	Menadžer	Menadžer Kontakt	Broj zaposlenika
Financije	Ivan Horvat	+385 91 345 6789	0
Marketing	Marko Horvat	+385 98 456 7890	0
Razvoj (Development)	Ivan Perić	+385 97 567 8901	5
Prodaja	Mihael Živković	+385 92 678 9012	1
Ljudski Resursi (HR)	Marko Horvat	+385 98 456 7890	1
Administracija	Kristijan Blažević	+385 92 890 1235	1
Dizajn	Petra Kovač	+385 97 567 8865	2
Testiranje (QA)	Luka Babić	+385 97 767 8901	1
Projektni Menadžment	Marko Horvat	+385 98 456 7890	1
Sigurnost	Marko Horvat	+385 98 456 7890	1

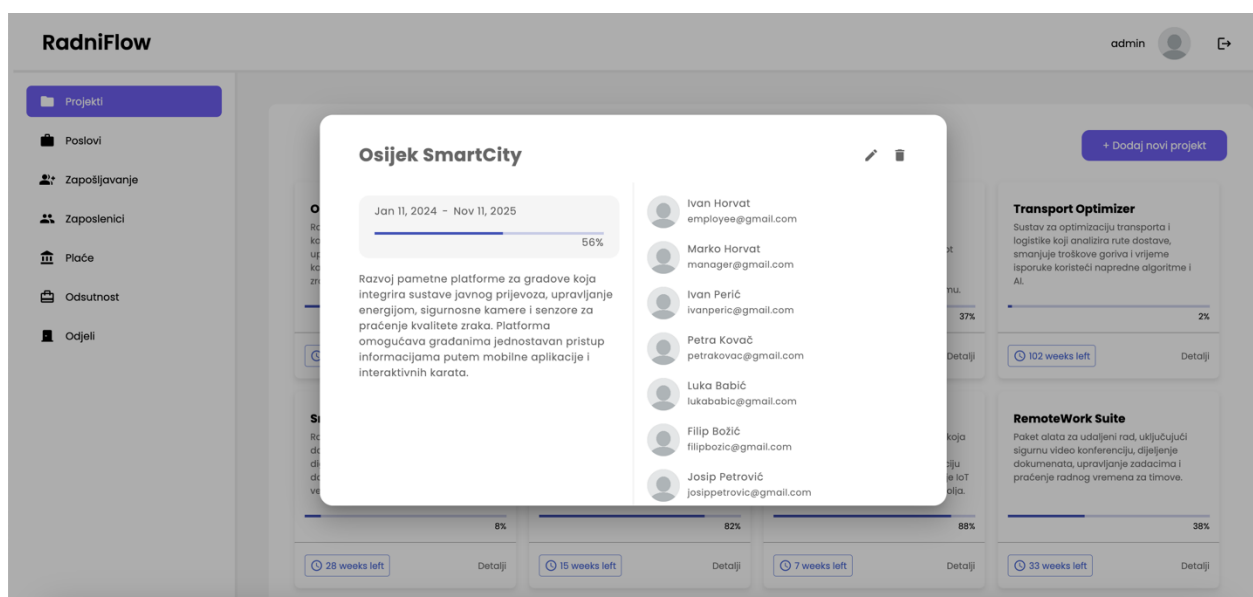
Slika 5.6. Tablični prikaz odjela

Nakon toga prelazi se na kreiranje posla ispunjavanjem forme s nazivom, opisom, odjelom, datumima objave i zatvaranja te prioriteta i lokacijom. Poslovi su prikazani u obliku kartica s opcijama za pregled, uređivanje i brisanje. Forma za kreiranje posla nalazi se na slici 5.7.



Slika 5.7. Forma za kreiranje posla

Nakon što su poslovi definirani, administrator može kreirati i pregledavati projekte. U formu se unosi naziv i opis projekta, odabire se menadžer ili voditelj te se postavlja datum početka i završetka projekta. S obzirom na datum početka i završetka projekta, formira se traka napretka projekta. Projekti su prikazani u obliku kartica s opcijama za pregled, uređivanje i brisanje. Klikom na karticu, administrator može vidjeti dodatne o projektu. Kartični prikaz projekta nalazi se na slici 5.8.



Slika 5.8. Detaljni prikaz projekta

Sljedeći korak za administratora je kreiranje zaposlenika. Kroz formu se unose osnovni podaci poput imena, prezimena, adrese e-pošte, telefona i adrese. Svaki zaposlenik povezuje se s poslom, odjelom i projektom te mu se dodjeljuje osnovna plaća i broj dana godišnjeg odmora i bolovanja. Prilikom unosa sustav automatski generira korisnički račun za zaposlenika. Forma za kreiranje zaposlenika prikazana je na slici 5.9.

The screenshot shows the 'Dodaj Novog Zaposlenika' (Add New Employee) form in the RadniFlow application. The form is divided into several sections: 'Osobni Podaci' (Personal Data) with fields for 'Ime*' (Name), 'Prezime*' (Surname), 'Email*', 'Adresa' (Address), 'Broj Telefona' (Phone Number), and 'Datum Zapošljavanja*' (Start Date) set to 01.12.2024. A red note indicates 'Ime je obavezno.' (Name is mandatory). 'Informacije o Poslu' (Job Information) includes dropdowns for 'Posao' (Job), 'Odjel' (Department), and 'Projekt' (Project), along with 'Plaća 0' (Salary) and 'Status Zaposlenja*' (Employment Status) set to 'Aktivan'. The 'Uloga' (Role) dropdown is set to 'Employee'. 'Podaci o Godišnjem i Bolovanju' (Annual and Sick Leave Data) shows 'Dani Godišnjeg Odsustva 20' (Annual Leave Days) and 'Dani Bolovanja 10' (Sick Leave Days). A blue 'Dodaj Zaposlenika' button is at the bottom right. The background shows a table of existing employees with columns for name, phone, department, and project.

Slika 5.9. Prikaz forme za kreiranje zaposlenika

Podaci o zaposlenicima prikazuju se u tablici, gdje administrator može pregledavati, uređivati ili brisati informacije. Klikom na zaposlenika unutar tablice otvara se zaslon na kojem su prikazani detalji o zaposleniku. S desne strane nalazi se gumb *Uredi zaposlenika* koji otvara formu u kojoj administrator ima mogućnost ažurirati informacije o zaposleniku. Moguće je promijeniti adresu e-pošte, adresu, odjel, projekt zaposlenika i slično. Prikaz profila zaposlenika nalazi se na slici 5.10.

The screenshot shows the employee profile for 'Luka Babić', an 'Automatski tester (Automation Tester)'. The profile includes a 'Uredi Zaposlenika' (Edit Employee) button. Key information is displayed in a structured layout: 'Preostali Dani Godišnjeg Odmora 20 / 20' (Remaining Annual Leave Days) and 'Preostali Dani Bolovanja 10 / 10' (Remaining Sick Leave Days). Personal details include 'Odjel: Testiranje (QA)', 'Projekt: Osijek SmartCity', 'Lokacija: Vukovarska 105, Osijek', 'Telefon: +385 97 767 8901', 'Email: lukababic@gmail.com', 'Plaća: €1,300.00', and 'Datum Zapošljavanja: 01.10.2024'.

Slika 5.10. Prikaz profila zaposlenika

Administrator ima uvid u plaće zaposlenika, svaki mjesec ima mogućnost generirati evidenciju plaća za sve postojeće zaposlenike klikom na gumb *Generiraj plaće za tekući mjesec*. Evidencija je prikazana je u tabličnom formatu, gdje administrator može pregledavati sve zapise, dodavati bonus i mijenjati status isplate. Prikaz evidencije plaća nalazi se na slici 5.11.

Uspješno generirane plaće

admin

Pretraži plaće

+ Generiraj plaće za tekući mjesec

Ime	Plaća	Bonus	Status
Ivan Horvat	\$1,200.00	0	U procesu
Marko Horvat	\$1,700.00	0	U procesu
Ivan Perić	\$1,300.00	0	U procesu
Petra Kovač	\$1,200.00	0	U procesu
Luka Babić	\$1,300.00	0	U procesu
Filip Božić	\$1,400.00	0	U procesu
Josip Petrović	\$1,400.00	0	U procesu
Marin Ivanković	\$1,400.00	0	U procesu
Nikolina Đukić	\$1,300.00	0	U procesu
Dora Bilić	\$1,500.00	0	U procesu
Kristijan Blažević	\$1,800.00	0	U procesu

Slika 5.12. Prikaz evidencije plaća zaposlenika

Na zaslonu za upravljanje odsutnostima administrator može vidjeti stanje godišnjeg odmora i bolovanja svih zaposlenika. Podaci su prikazani u tablici koja omogućuje pregled i pretraživanje. Klikom na zaposlenika u tablici, administrator se preusmjerava na profil zaposlenika, gdje može urediti broj dana godišnjeg odmora i bolovanja. Evidencija odsutnosti prikazana je na slici 5.13.

RadniFlow

admin

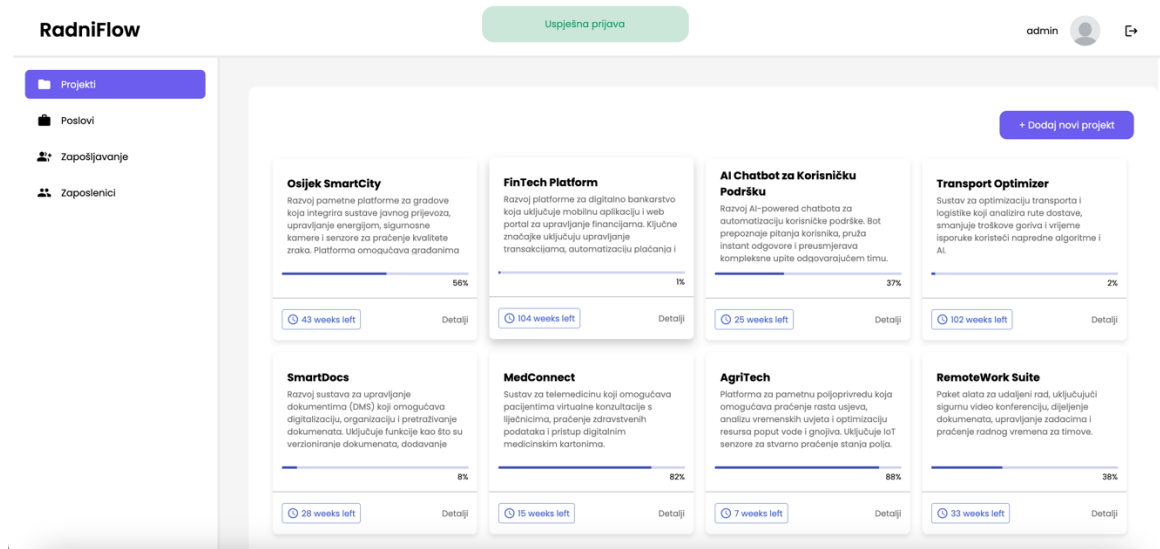
Pretraži zaposlenike

Ime	E-mail	Broj telefona	Preostalo bolovanje	Preostalo godišnji odmor
Ivan Horvat	employee@gmail.com	+385 91 345 6789	10 / 10	20 / 20
Marko Horvat	manager@gmail.com	+385 98 456 7890	10 / 10	20 / 20
Ivan Perić	ivanperic@gmail.com	+385 97 567 8901	10 / 10	20 / 20
Petra Kovač	petrakovac@gmail.com	+385 97 567 8865	10 / 10	20 / 20
Luka Babić	lukababic@gmail.com	+385 97 767 8901	10 / 10	20 / 20
Filip Božić	filipbozic@gmail.com	+385 97 901 2345	10 / 10	20 / 20
Josip Petrović	josippetrovic@gmail.com	+385 91 123 4567	10 / 10	20 / 20
Marin Ivanković	marin.ivankovic@email.com	+385 91 901 2346	10 / 10	20 / 20
Nikolina Đukić	nikolina.dukic@email.com	+385 91 234 5679	10 / 10	20 / 20
Dora Bilić	dora.bilic@email.com	+385 98 123 4568	10 / 10	20 / 20
Kristijan Blažević	kristijan.blazevic@email.com	+385 92 890 1235	10 / 10	20 / 20

Slika 5.13. Evidencija odsutnosti

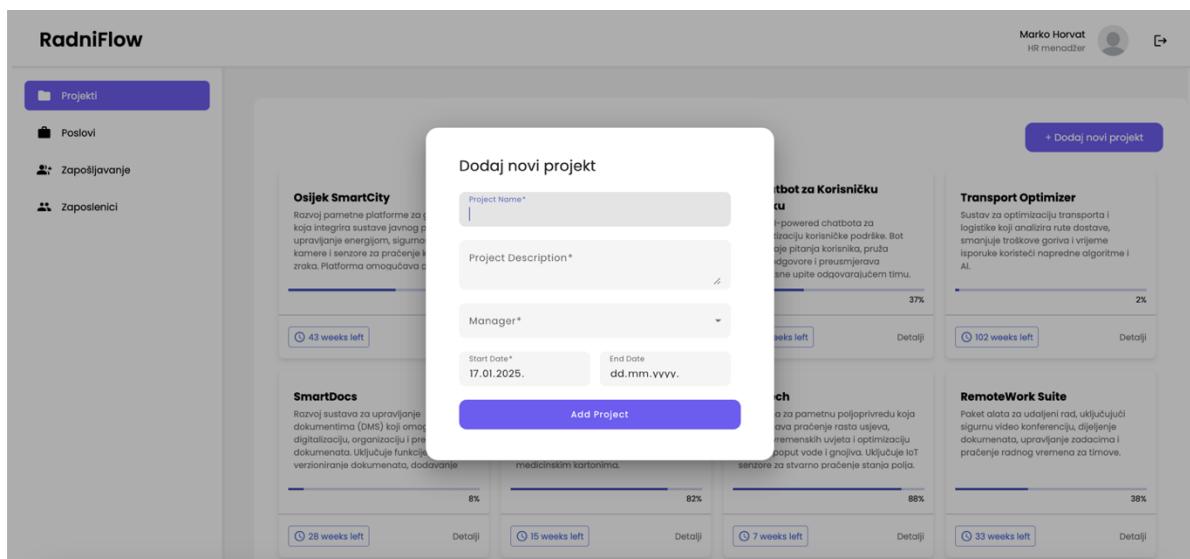
5.3. Korisničko iskustvo uloge Menadžer

Menadžer se prijavljuje unosom adrese e-pošte i lozinke. Nakon uspješne prijave, otvara se korisničko sučelje prilagođeno ulozi menadžera, prikazano na slici 5.14. Na desnoj strani nalazi se izbornik s poveznicama na projekte, poslove, zapošljavanje i zaposlenike.



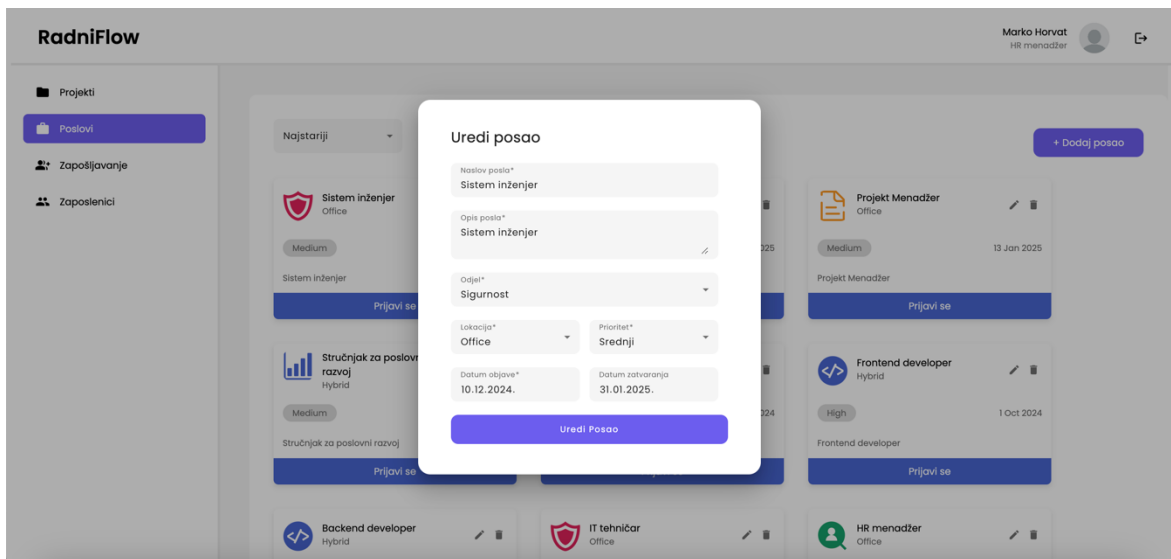
Slika 5.14. Korisničko sučelje prilagođeno ulozi Menadžer

Menadžer ima mogućnost pregleda, dodavanja, uređivanja i brisanja projekata. Projekti su prikazani u obliku kartica s osnovnim informacijama. Klikom na karticu otvara se detaljni prikaz projekta, gdje menadžer može ažurirati podatke poput naziva projekta, opisa, menadžera, datuma početka i završetka. Također, dostupna je opcija brisanja projekta. Forma za kreiranje projekta nalazi se na slici 5.15.



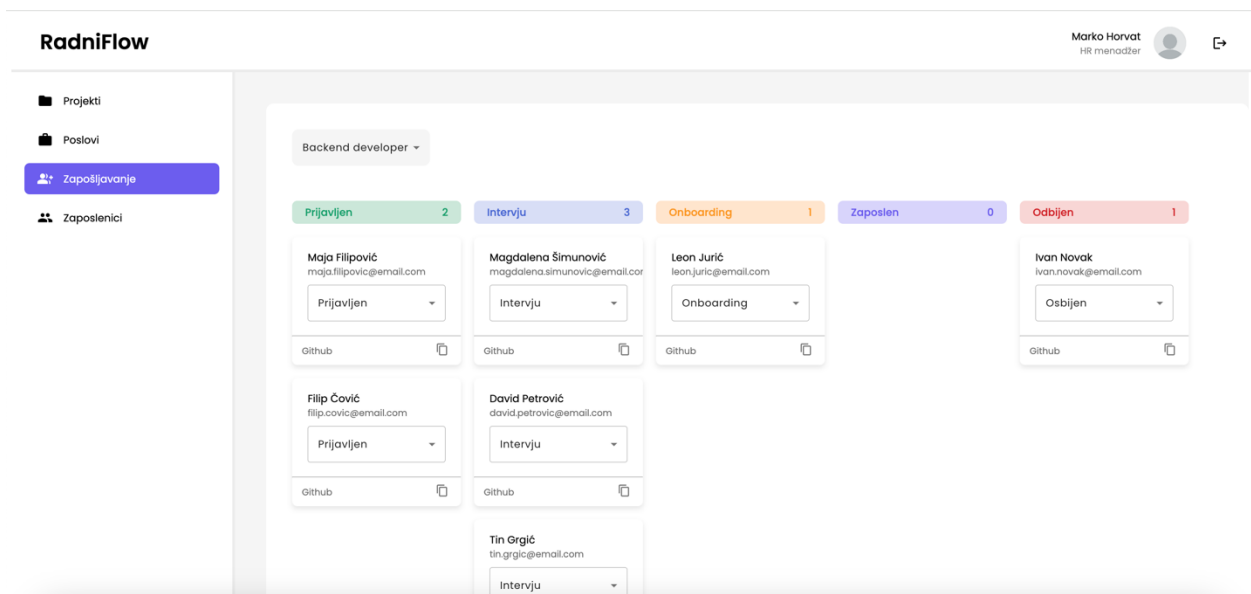
Slika 5.15. Forma za kreiranje projekta

Menadžer može pregledavati i dodavati poslove te brisati postojeće. Poslovi su prikazani u obliku kartica s osnovnim informacijama o svakom poslu. Forma za ažuriranje posla prikazana je na slici 5.16.



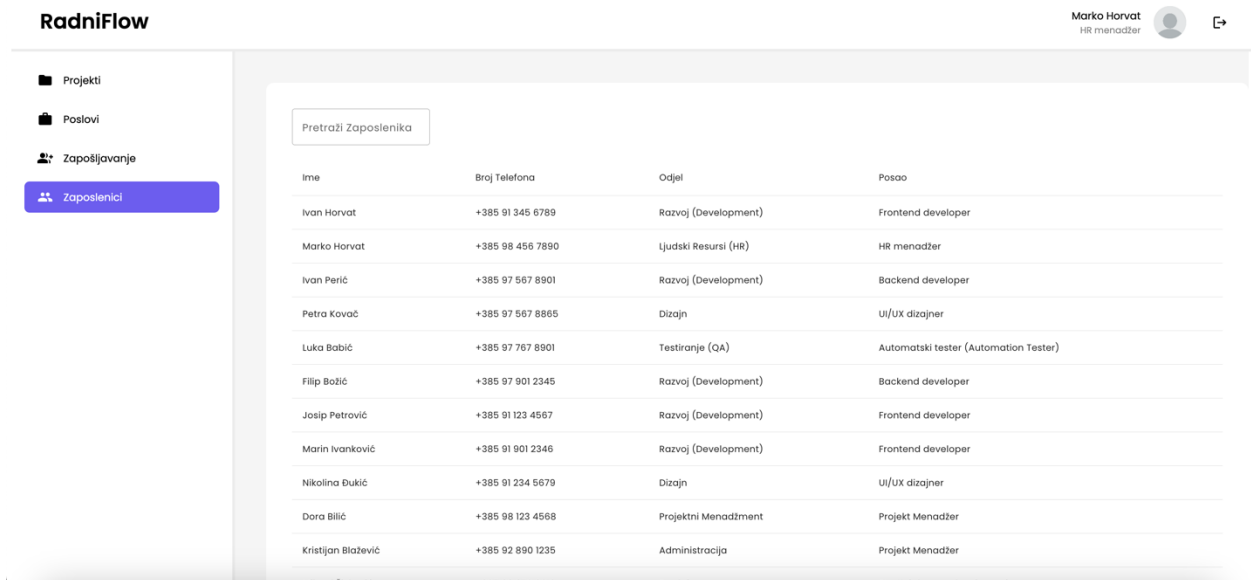
Slika 5.16. Prikaz forme za ažuriranje posla

Menadžer ima mogućnost pregleda zapošljavanja i kandidata. Kandidati su prikazani u obliku kartica i kategorizirani prema statusu zaposlenja. Svaka kartica sadrži osnovne informacije o kandidatu, poput imena, prezimena, adrese e-pošte, poveznice na *GitHub* profil i priloženog životopisa. Klikom na ikonu dokumenta otvara se novi prozor u kojem menadžer može pregledati životopis kandidata. Prikaz zaslona za zapošljavanje nalazi se na slici 5.17.



Slika 5.17. Prikaz kandidata

Menadžer može pregledavati informacije o zaposlenicima, ali nema mogućnost dodavanja, uređivanja ili brisanja njihovih podataka. Prikaz zaposlenika organiziran je u tabličnom formatu prikazan na slici 5.18, gdje menadžer ima uvid u osnovne podatke poput imena, prezimena, pozicije i odjela. Klikom na zaposlenika moguće je pregledati dodatne detalje bez mogućnosti uređivanja.



RadniFlow

Marko Horvat
HR menadžer

Projekti
Poslovi
Zapošljavanje
Zaposlenici

Pretraži Zaposlenika

Ime	Broj Telefona	Odjel	Posao
Ivan Horvat	+385 91 345 6789	Razvoj (Development)	Frontend developer
Marko Horvat	+385 98 456 7890	Ljudski Resursi (HR)	HR menadžer
Ivan Perić	+385 97 567 8901	Razvoj (Development)	Backend developer
Petra Kovač	+385 97 567 8865	Dizajn	UI/UX dizajner
Luka Babić	+385 97 767 8901	Testiranje (QA)	Automatski tester (Automation Tester)
Filip Božić	+385 97 901 2345	Razvoj (Development)	Backend developer
Josip Petrović	+385 91 123 4567	Razvoj (Development)	Frontend developer
Marin Ivanković	+385 91 901 2346	Razvoj (Development)	Frontend developer
Nikolina Đukić	+385 91 234 5679	Dizajn	UI/UX dizajner
Dora Bilić	+385 98 123 4568	Projektni Menadžment	Projekt Menadžer
Kristijan Blažević	+385 92 890 1235	Administracija	Projekt Menadžer

Slika 5.18. Prikaz tablice zaposlenika

5.4. Korisničko iskustvo uloge Zaposlenik

Zaposlenik se može prijaviti u sustav klikom na gumb *Prijavi se* na početnom zaslonu aplikacije. Prilikom prijave potrebno je unijeti adresu e-pošte i lozinku. Nakon uspješne prijave, zaposleniku se otvara korisničko sučelje prilagođeno njegovoj ulozi. Sučelje omogućuje pregled osobnog profila i povezanih podataka. Zaposlenik može vidjeti osnovne informacije o sebi, uključujući ime, prezime, email, telefon i adresu. Također, prikazane su informacije o njegovom radnom mjestu i uvjetima, kao što su naziv odjela kojem pripada, trenutno dodijeljeni projekt, osnovna plaća, te stanje preostalih dana godišnjeg odmora i bolovanja.



Nikolina Đukić
UI/UX dizajner



Preostali Dani
Godišnjeg Odmora

20 / 20



Preostali Dani
Bolovanja

10 / 10

Odjel

Dizajn

Projekt

Osijek SmartCity

Lokacija

Šetalište Kardinala 1, Osijek

Telefon

+385 91 234 5679

Email

nikolina.dukic@email.com

Plaća

€1,300.00

Datum Zapošljavanja

01.09.2024

Slika 5.19. Prikaz profila zaposlenika

6. ZAKLJUČAK

Predstavljena je web aplikacija za upravljanje ljudskim resursima i internim poslovnim procesima koja rješava ključne probleme tradicionalnih pristupa poput sporosti, sklonosti pogreškama i neučinkovitosti. Digitalizacijom i automatizacijom procesa aplikacija omogućuje organizacijama značajno smanjenje administrativnog opterećenja, brže donošenje odluka i poboljšanu točnost u radu s osjetljivim podacima.

Aplikacija rješava problem upravljanja zaposlenicima, procesima zapošljavanja i kroz funkcionalnosti poput napredne analitike plaća, administracije godišnjih odmora te upravljanja korisničkim profilima i prijavama na slobodna radna mjesta. Posebna pažnja posvećena je sigurnosnim aspektima kako bi se osigurala pouzdanost i zaštita povjerljivih podataka.

Razvoj aplikacije temelji se na suvremenim tehnologijama koje omogućuju skalabilnost i responzivnost, pružajući organizacijama dugoročno održivo rješenje. Aplikacija pruža intuitivno korisničko iskustvo prilagođeno različitim kategorijama korisnika.

Uz rješavanje postojećih problema, aplikacija postavlja temelj za buduće inovacije, uključujući napredne analitičke alate i personalizaciju, čime dodatno osigurava prilagodljivost promjenjivim poslovnim potrebama i dugoročnu vrijednost za organizacije.

LITERATURA

- [1] Workday, <https://www.workday.com/>, pristupljeno 20.04.2024.
- [2] ADP Workforce Now, <https://www.adp.com/>, pristupljeno 20.04.2024.
- [3] SAP SuccessFactors, <https://www.sap.com/products/hcm.html>, pristupljeno 20.04.2024.
- [4] Zoho People, <https://www.zoho.com/people/>, pristupljeno 20.04.2024.
- [5] Angular, <https://v17.angular.io/guide/what-is-angular>, pristupljeno 20.4.2024.
- [6] Angular Introduction to components and templates, <https://v17.angular.io/guide/architecture-components>, pristupljeno 20.4.2024.
- [7] ASP.NET, <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet>, pristupljeno 20.04.2024.
- [8] Microsoft SQL, <https://www.microsoft.com/en-us/sql-server>, pristupljeno 20.04.2024.
- [9] Node Package Manager, <https://docs.npmjs.com/about-npm>, pristupljeno 20.04.2024.
- [10] Angular Components, <https://v17.angular.io/guide/component-overview>, pristupljeno 20.04.2024.
- [11] Rider JetBrains, <https://www.jetbrains.com/rider/features/>, pristupljeno 20.04.2024.
- [12] CORS, <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>, pristupljeno 20.04.2024.
- [13] Entity Framework Core, <https://learn.microsoft.com/en-us/ef/core/>, pristupljeno 20.04.2024.
- [14] Repository Pattern, <https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30>, pristupljeno 20.04.2024.
- [15] ASP.NET Identity, <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-9.0&tabs=visual-studio>, pristupljeno 20.04.2024.

SAŽETAK

Ovaj diplomski rad prikazuje razvoj web aplikacije za upravljanje ljudskim resursima i internim poslovnim procesima. Cilj aplikacije je digitalizacija i optimizacija ključnih aktivnosti poput upravljanja zaposlenicima, procesima zapošljavanja, upravljanja plaćama i administracijom godišnjih odmora. Aplikacija omogućuje prilagođeno korisničko iskustvo za administratore, voditelje, zaposlenike i kandidate, pružajući funkcionalnosti poput upravljanja korisničkim profilima, praćenja plaća te prijave na slobodna radna mjesta.

Razvoj aplikacije temelji se na suvremenim tehnologijama, osiguravajući skalabilnost, sigurnost i visoku razinu performansi. Implementirani su visoki sigurnosni standardi, uključujući autentifikaciju i autorizaciju temeljenu na ulogama, dok je responzivni dizajn omogućio prilagodljivost aplikacije na različite uređaje.

Aplikacija predstavlja značajan korak prema modernizaciji upravljanja ljudskim resursima, omogućujući organizacijama povećanje učinkovitosti i smanjenje mogućnosti za pogreške, dok istovremeno pruža intuitivno korisničko sučelje.

Ključne riječi: Angular, digitalizacija, Entity Framework, ljudski resursi, upravljanje zaposlenicima, Web aplikacija.

ABSTRACT

Web application for managing human resources and internal business processes

This thesis presents the development of a web application for human resources management and internal business processes. The goal of the application is to digitize and optimize key activities such as employee management, recruitment processes, payroll management and vacation administration. The application provides a customized user experience for administrators, managers, employees and candidates, providing functionalities such as user profile management, payroll tracking and job application.

The application development is based on modern technologies, ensuring scalability, security and a high level of performance. High security standards have been implemented, including role-based authentication and authorization, while responsive design has enabled the application to be adaptable to different devices.

The application represents a significant step towards the modernization of human resources management, enabling organizations to increase efficiency and reduce the possibility of errors, while at the same time providing an intuitive user interface.

Keywords: Angular, digitalization, Entity Framework, employee management, human resources, Web application.