Dizajn i analiza aplikacije za vladanje projektima

Dombaj, Romeo

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet u Puli

ROMEO DOMBAJ

DIZAJN I ANALIZA APLIKACIJE ZA VLADANJE PROJEKTIMA

Završni rad

Pula, rujan 2023. godine

Sveučilište Jurja Dobrile u Puli

Tehnički fakultet u Puli

ROMEO DOMBAJ

DIZAJN I ANALIZA APLIKACIJE ZA VLADANJE PROJEKTIMA

Završni rad

JMB: 0303096597, redoviti student

Studijski smjer: prijediplomski sveučilišni studij Računarstvo

Predmet: Programsko inženjerstvo

Znanstveno područje: Tehničke znanosti

Znanstveno polje: Računarstvo

Znanstvena grana: Programsko inženjerstvo

Mentor: prof. dr. sc. Tihana Galinac Grbac

Pula, rujan 2023. godine

Tehnički fakultet u Puli

Ime i prezime studenta/ice Romeo Dombaj

JMBAG 0303096597

Status: ☒ redoviti ☐ izvanredni

PRIJAVA TEME ZAVRŠNOG RADA

Tihana Galinac Grbac
Ime i prezime mentora

Računarstvo
Studij

Programsko inženjerstvo
Kolegij

Potvrđujem da sam prihvatio/la temu završnog/diplomskog rada pod naslovom:

Dizajn i analiza aplikacije za vladanje projektima

(na hrvatskom jeziku)

Design and analysis of project governance application

(na engleskom jeziku)

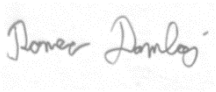
Datum: 22.3.2023.

IZJAVA O KORIŠTENJU AUTORSKOGA DJELA

Ja, **Romeo Dombaj**, dajem odobrenje Sveučilištu Jurja Dobrile u Puli, nositelju prava korištenja, da moj završni rad pod nazivom „Dizajn i analiza aplikacije za vladanje projektima“ upotrijebi da tako navedeno autorsko djelo objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te preslika u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu sa Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Potpis

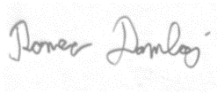


U Puli, 13.9.2023.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani **Romeo Dombaj**, kandidat za prvostupnika računarstva ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student



U Puli, 13.9.2023.

Sadržaj

1. UVOD

U posljednjih par desetljeća svjedočimo ubrzanom razvoju informacijsko-komunikacijskih tehnologija. Razvoj Interneta, kao globalne komunikacijske mreže omogućio je ostvarivanje komunikacije između udaljenih uređaja (osobnih računala, mobitela, tableta, raznih senzorskih uređaja) uz jedinu pretpostavku da su spojeni na Internet mrežu. Iako je razvoj Interneta omogućio globalno umrežavanje krajnjih uređaja tek razvoj World Wide Web tehnologija pojednostavio je razmjenu sadržaja između raznih uređaja. Web tehnologije unose niz prednosti u raznim primjenama. Web aplikacija dostupna je na bilo kojem mjestu koje ima pristup internetu. Osim lokacijskih povlastica web aplikacije također lako su prilagodljive na različite uređaje, tako web aplikaciji možemo pristupiti na bilo kojem uređaju dali je to bilo računalo, mobitel ili tablet. Pristup može biti prioritiziran na temelju uloge koju svaki korisnik obnaša u organizaciji [1]. Na početku razvoja Web tehnologija Internetom se izmjenjivao uglavnom statički sadržaj kao što je to prijenos dokumenta sa poslužiteljskog računala (engl. server) na klijentsko računalo (engl. client) uz odgovarajući prikaz na klijentskom ekranu. Međutim današnje Web tehnologije omogućavaju razvoj dinamičnih Web mjesta kojima se mogu razmjenjivati i drugi mediji poput slike, videa, i programa koji će se moći izvoditi unutar klijentskog Web preglednika. Web tehnologije stoga su našle veliku primjenu u različitim domenama primjene poput sustava rezervacija, sustava za instant komunikaciju, sustavi za elektronički marketing i prodaju, pa do sustava za potporu demokratskom odlučivanju. Pristup web aplikacije može se zaštiti pomoću autentifikacije i autorizacije.

Ovaj završni rad posvećuje se temi primjena Web tehnologija u "Vladanju projektima" (engl. Project governance). Cilj ovog rada je dizajn i analiza Web aplikacije za potrebe vladanja projektima u nekoj organizaciji.

Vladanje projektima pruža okvir za etičko donošenje odluka i upravljačke akcije unutar organizacije koje se temelje na transparentnosti, odgovornosti i jasno definiranim ulogama [2]. Upravljanje u organizacijama koje se bave razvojem softvera često je podijeljeno po 4P principu na upravljanje ljudima (engl. People managment), upravljanje procesima (engl.

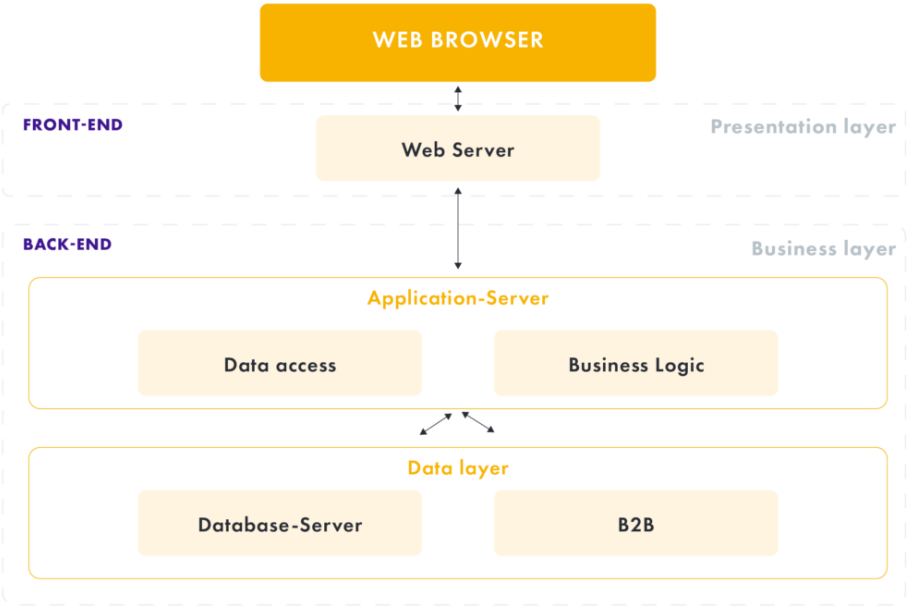
Process managment), upravljanje produktima (engl. Product Management) i upravljanje projektima (engl. Project managment) [3]. Svaka od navedenih grana modela važan je dio za upravljanje svakom organizacijom pa se upravljačke aktivnosti svake od navedenih grana kontinuirano usklađuju sa organizacijskim strateškim ciljevima. U gotovo svim organizacijama, pogotovo većim, možemo naići na nekakav oblik vladanja projektima sa manjim varijacijama, no vrlo česti problem je njihova međusobna nepovezanost, slaba komunikacija i tako gubljenje vremena na ljudske nesporazume između njih [2]. Primjenom Web tehnologija zajedno sa modelom vladanja projektima pokušavamo riješiti taj problem na način da sva komunikacija ide preko istog sustava koji određuje, odobrava i ograničava raspoložive resurse za neki projekt (npr. da ne dođe do toga da se jedan zaposlenik doda na dva različita zadatka u isto vrijeme). U sljedećim podnaslovima završnog rada biti će objašnjeno i prikazano kako se rješavaju određeni problemi da bi aplikacija i sam model vladanja projektom mogli raditi.

1.1. Ciljano tržište

Ova web aplikacija cilja i upotpunjuje potrebe svake veće tvrtke koja je projektno orijentirana, pospješuje upravljanje zaposlenicima i projektima, te međusobnu komunikaciju različitih upravitelja neke tvrtke. Aplikacija ima veliku prednost jer zapravo sličnih nema toliko mnogo na tržištu, većina njih radi odvojeno linijsko upravljanje i projektno upravljanje.

2. WEB TEHNOLOGIJE

U ovom dijelu završnog rada prikazat će se sve metode i tehnologije kojim je zadatak izvršen, te osnovna struktura cijele aplikacije. Za izradu web aplikacije potrebna su nam tri



Slika 1 Prikaz strukture web aplikacija

osnovna sloja pomoću kojih se ostvaruje funkcionalnost aplikacije. Slojevitost dijelimo na front-end; dio aplikacije koji se izvršava na samom računalu korisnika (klijentski dio Web aplikacija), back-end; dio aplikacije koji se izvršava na serveru (poslužiteljski dio Web aplikacije), te baza podataka; baza koja sadržava sve potrebne podatke koji se obrađuju i prikazuju.

2.1. Front-end

Kao što je već spomenuto front-end je dio aplikacije koji se pokreće na samom računalu korisnika i odnosi se na klijentski dio Web aplikacije. Detaljnije front-end je dio aplikacije koji prikazuje sve podatke, informacije i različite funkcije obrade samom korisniku kroz Web preglednik. Poznatiji Web preglednici su Internet Explorer, Microsoft Edge, Google Crome, Mozilla Firefox.

2.1.1. Planiranje izgleda – figma

Planiranje izgleda aplikacije izvršeno je pomoću web aplikacije Figma. Figma sa svojim „drag & drop“ funkcionalnošću vizualnih elemenata omogućava lako, jednostavno i brzo strukturiranje osnovnog izgleda aplikacije prema kojem se kasnije izrađuje sama aplikacija.

2.1.2. Izrada front-enda web aplikacije – html, css, javascript

Za izgradnju front-enda web aplikacije korištena su tri standardna jezika za izradu web stranica, HTML, CSS i JavaScript. Osim tri osnovna jezika korištena je JavaScript biblioteka React.js za izradu korisničkog sučelja, te još nekolicina dodatnih tehnologija koje upotpunjuju korisničko iskustvo poput Charts.js koji sliži za prikaz različitih grafova, crtanje PERT dijagrama pomoću kojeg računamo kritični put. U nastavku opisati će se svaka od korištenih tehnologija i tehnika.

2.1.2.1. HTML

HTML nije programski jezik već prezentacijski jezik što znači da je to zapravo sustav za identifikaciju i opisivanje različitih komponenata dokumenta poput naslova, ulomaka i lista [4]. HTML ili „Hyper Text Markup Language“ se koristi za stvaranje dokumenata web stranice čiji je sufiks oblika .html. Pomoču HTML-a možemo stvarati i pozicionirati različitie elemenate na stranici, npr. prozora, gumbova, tekstualnih okvira i slično. HTML je prvi jezik koji se koristio za izradu korisničkog sučelja kod web aplikacija. Osnovna značajka bila je „Hyper text“ odnosno linkovi. Linkove nazivamo neki skup znakova, riječi koji u pozadini sadrže informacije za prelazak na drugu web mjesto. Klikom na link zapravo odlazimo na novu web mjesto.

2.1.2.2. CSS

Dok HTML služi za opisivanje sadržaja web stranice, Cascading Style Sheets (CSS) nam služi za opis kako će okvir sadržaja izgledati [4]. Često imamo kolekciju stranica koje se nalaze na jednom Web sjedištu (engl. Web site) te pomoću CSS tehnologije omogučuen nam je jedinstven dizajn svim povezanim stranicama. CSS je stilski jezik koji nam omogućuje promjenu stilova elemenata poput boje i veličine pozadine, te da im se može dodati neka animacija. Nakon razvitka HTML-a dodatno je razvijen CSS kako bi se promijenio čisti statični izgled HTML-a u oblikovane stranice kakve danas poznajemo. Osim stiliziranja sadržaja web stranice CSS nam omogućuje da prilagođavamo izgled istog sadržaja na ekrane različitih dimenzija. Trenutna najnovija verzija CSS-a je CSS3 koji nadodaje mogućnost animiranja sadržaja [6].

2.1.2.3. JavaScript

JavaScript je skriptni programski jezik koji se koristi kako bi se web stranici ugradile dodatne dinamične funkcije. JavaSkript se koristi za dinamičko upravljanje elementima web stranice, dinamičko upravljanje stilovima web stranica ili pak upravljanje postavkama samog web preglednika [4]. Neke od mogućnosti koje nam JavaScript omogućuje su provjeravanje različitih korisničkih unosa na klijentskoj strani, dinamičku promjenu stilova nekih elemenata (npr. boja pozadine iz plave u crvenu), spremanje podataka u samom web pregledniku lokalno na klijentskoj strani, te kreiranje različitih značajki i menija. JavaSkript kao ni CSS nisu potrebni za kreaciju web stranica no dodaju važnu ulogu o preglednosti i dinamičkoj prilagodbi Web stranice zasnovanu na interakciji sa korisnikom, pa time uzrokuje povećanju zadovoljstva korisnika [7]. Jedna od novijih značajki JavaScripta je DOM ili *Document Object Model* [8]. DOM je skup različitih sučelja koja omogućuju upravljanje elementima u Web dokumentu odnosno HTML elementima. DOM nam omogućuje da aplikacije budu dinamičke što znači da svaki put kada se nešto poput boje pozadine promjeni stranica se ne mora ponovo učitati već se to obrađuje automatski u stvarnom vremenu. Važno je da je DOM standardiziran pa se može koristiti zajedno i sa drugim programskim jezicima koji se primjenjuju u Web tehnologijama poput PHP, Ruby, Python, C++, Java, Perl.

2.1.3. React.js

Uz osnovne jezike, koristili smo React.js biblioteku. React je JavaScript biblioteka otvorenog koda za izradu interaktivnog korisničkog sučelja koju je razvio Facebook uz osnovni zahtijev da napravi biblioteku koja omogućava brže izvođenje aplikacije i bržu interakciju sa korisnikom. React proširuje JavaScript dodavanjem mnogih dodatnih značajki i mogućnosti te značajno mijenja način izrade web aplikacija [9]. U Reactu, HTML se zamjenjuje Java Script proširenom sintaksom za XML, JSX kôdom koji je gotovo identičan HTML-u, ali s nekim sitnijim različitim svojstvima. Osnovna ideja Reacta je dekompozicija aplikacije na manje dijelove, tj. da se cijela aplikacija podijeli na manje komponente, pri čemu svaka komponenta ima što manje funkcionalnosti, po mogućnosti samo jednu. Ova struktura omogućuje čist i pregledan kôd te omogućuje ponovno korištenje komponenata bez ponovnog pisanja koda.

Za razliku od JavaScript model objekta dokumenta (DOM) React koristi React model objekta dokumenta (DOM) za prijenos, u osnovi ove dvije tehnologije potpuno su iste tj. obje se koriste za upravljanje elementima dokumenta u stvarnom vremenu. Razlikuju se u vremenskoj funkciji odnosno React DOM je virtualni DOM što možemo prikazati kao virtualna mašina na nekom pravom sustavu (DOM-u). Virtualni DOM se osvježava kroz cijelo vrijeme, no tek kad se dese neke varijacije u izgledu tek onda se virtualni DOM preslikava u „realni“ DOM, [8].

Primjer komponente može biti najmanji element, poput gumba, koji se može prilagoditi i proširiti prema našim potrebama i dodati mu specifične funkcionalnosti. Ovaj pristup omogućuje ponovnu upotrebu gumba ili bilo koje druge komponente kroz cijelu aplikaciju. Svaka komponenta strukturirana je kao JavaScript funkcija koja vraća JSX kôd. Komponente također mogu sadržavati druge komponente unutar sebe, stvarajući tako složene hijerarhije. Sve počinje s glavnom komponentom, obično nazvanom index.js, koja se direktno uključuje u index.html. Primjer složene komponente može biti lista podataka koja predstavlja jednu komponentu, a ta lista može sadržavati pojedinačne komponente za svakog zaposlenika unutar nje kao što je to predstavljeno u kodu na Slici 2.

Osim komponenata važno je napomenuti i ostale dodatne funkcionalnosti Reacta poput „hook“ -ova. Hookovi su funkcije za upravljanjem stanja React komponenti, a promjene stanja moraju pratit određena pravila da bi onda radila u skladu sa ostalim elementima. Dva osnovna hooka Reacta jesu „useState()“ i „useEffect()“. UseState() prima dva argumenta, ime varijable i funkcija koja sprema podatak na mjesto te varijable. UseState() funkcionira na način spremanja nekog stanja neke varijable. UseEffect() na drugu rugu ima potpuno drugačiju funkciju. UseEffect() je funkcija koja se izvršava kad je komponenta React-a osvježena i pokreće sve što u nju stavimo. Osim početnog pokretanja useEffect() koristimo i za pokretanje redaka koda ako dođe do promjene ovisnosti koje smo promijenili.

```
import styles from "./LineList.module.css";
import LineListElement from "./LineListElement";

const LineList = (props) => {
  const data = props.data;
  const path = props.path;
  const search = props.search;
  const isEmployee = props.isEmployee;

  return (
    <div className={styles.wrapper}>
      <div className={styles.list}>
        {data.map((item, i) => {
          if (item.name.includes(search)) {
            return (
              <LineListElement      You, 20 hours ago • line managment created
                isEmployee={isEmployee}
                key={i}
                path={path}
                i={i}
                data={item}
              />
            );
          }
        })}
      </div>
    </div>
  );
};

export default LineList;
```

Slika 2 Prikaz komponente sa komponentom unutar nje

2.1.4. Chart.js

Chart js je JavaScript biblioteka otvorenog koda (engl. Open source) koja nam omogućuje detaljan prikaz zadanih skupova podataka pomoću različitih grafova. Chart js jednostavno se integrira sa Reactom pomoću npm (node package manager) upravitelja paketa sa komandom „npm install chart.js react-chartjs-2“ [10]. Ova biblioteka nam nudi velik izbor različitih grafova poput stupičastih, linijskih, pita, radar i drugih oblika. U ovom projektu koristio se radar graf za prikaz određenih podataka.

2.2. Back-end

Back-end ili dio web aplikacije koji se pokreće na samom serveru je poveznica između korisnika i podataka. Na njemu se izvršavaju različite funkcije poput uzimanja podataka iz baze i slanje na front-end, autentifikacije korisnika, kodiranje podataka i druge funkcije kako bih se korisnik učinio čim sigurnijim i njegovo računalo bez previše opterećenja.

2.2.1. Node.js

Node.js je radno okruženje za izvođenje JavaScript koda. Node se koristi kod programiranja back-enda ondosno serverskog dijela web aplikacije. Pokreće se na V8 JavaScript mašini koju istu koriste Google Chrome i ostali web preglednici. Node je također otvorenog koda, a izrađen je u c++ programskom jeziku. Jedna od Node najvažnijih značajki je „npm“ (Node Package Manager) pomoću kojeg možemo instalirati svakakve različite tehnologije i pakete koje trebamo koristiti u svojoj web aplikaciji, [11]. Za primjer ove web aplikacije potrebna su dva veća paketa React i Express, način njihove instalacije je prikazan na dnu dokumenta.

2.2.2. Express.js

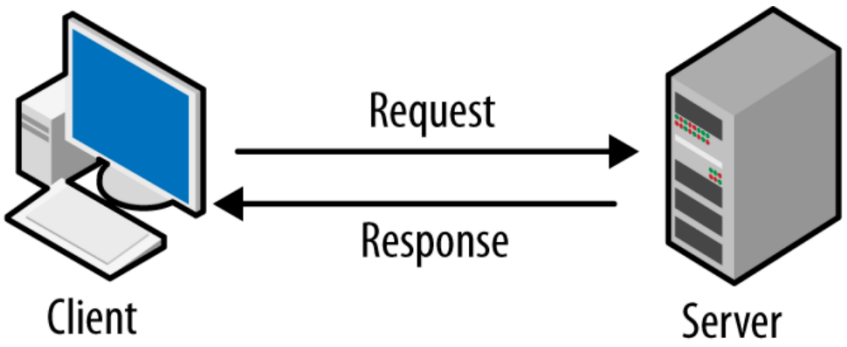
Express js je najpopularniji okvir za back-end aplikacije. Koristiti se u Node radnom okruženju te je zapravo baziran na JavaScriput, pa se smatra i JavaScript bibliotekom [12].

Express nudi različite alate i značajke pomoću kojih možemo lakše i kvalitetnije izraditi web aplikaciju. Nudi skup ruta koje koristimo kako bismo s front-end mogli dohvačati ili spremiti određene podatke određene baze i tablice podataka. Rute koriste osnovne HTTP zahtjeve: GET, POST, PATCH, PUT, DELETE. Express nam omogućuje i razvoj „middlewarea“ odnosno softvera koji dolazi između front-enda i back-enda, primjer takvog softverskog rješenja može biti autorizacija gdje middleware funkcija provjerava ima li korisnik koji želi dohvatiti ili spremiti neke podatke ovlasti nad njima.

2.2.3. Rest api

Za komunikaciju između korisnika i poslužitelja koristi se REST „Representional state transfer“ sučelje kao što je prikazano na Slici 6. REST funkcionira na osnovi HTTP zahtjeva (engl. Request) i odgovora (engl. Response) između poslužitelja (engl. server) i korisnika (engl. client) s dodatkom od nekoliko pravila koja se moraju poštivati. U pravila ubrajamo komunikaciju klijent-poslužitelj, poslužitelj ne smije spremati stanje odnosno trenutne podatke i mora koristiti uniformnog sučelja kako bi bio dostupan različitim korisnicima odnosno klijentima. Zato postoji norma koja definira izgled HTTP zahtjeva [13]. Postoji pet osnovnih HTTP zahtjeva:

1) GET – dohvaćanje podataka
2) POST – upisivanje retka podataka u bazu
3) PATCH – mijenja neki podatak u određenom retku u bazi
4) PUT – mijenja cijeli redak podataka u bazi
5) DELETE – briše redak podataka iz baze



Slika 3 Komunikacija klijent-poslužitelj

2.3. Baza podataka

Kao baza podataka korištena je baza MongoDB koji za upite (engl. query“) koristi MQL (MongoDB Query Language) jezik. Za razliku od klasičnih baza podataka MongoDB baza podataka umjesto tablice koristi kolekciju (engl. collection), a umjesto redaka koristi dokumente (engl. documents) [14]. Na osnovi kolekcije i dokumenti funkcioniraju isto kao i tablice te redci. Dokumenti su u obliku JSON standardnog formata --> { atribut : podatak }. Za povezivanje back-enda na bazu podataka koristi se Mongovo sučelje s autentifikacijom. MongoDB je besplatan za osobno korištenje, i dolazi s nekoliko korisnih alata kao što je CompassDB gdje možemo grafički pratiti sve baze podataka, te vidjeti, brisati i spremati različite podatke u njima. Upravljanje nad bazom, kolekcijama, dokumentima i podacima moguće je i preko naredbenog retka uz malo dodatnog podešavanja.

3. VLADANJE PROJEKTIMA U ORGANIZACIJAMA RAZVOJA SOFTVERA

Vladanje projektima (engl. Project governance) je sustav skupova određenih pravila i uputa koje se trebaju slijediti kako bih se povećala uspješnost nekog projekta odnosno uspješnost upravljanja nad nekim projektom i potrebnim resursima o kojima taj projekt ovisi [1]. Nije dovoljno imati definirana pravila, već da bi se postigla uspješnost neke organizacije pravila se moraju primjenjivati u radu organizacije [3].

Tvrtke koje su usmjerene na razvoj softvera često su projektno orijentirane, raspolažu velikim brojem projekata i različitim resursima. Najvažniji resurs u ovakvoj vrsti organizacije su ljudi odnosno djelatnici čije se kompetencije, vještine i sposobnosti koriste za stvaranje samih projekata. Sposobnost neke organizacije da privede projekte uspjehu jest razina sposobnosti planiranja i sustavnog pristupa razvoja kompetencija svojih ljudi odnosno usklađivanje vještina i sposobnosti zaposlenika sa budućim nadolazećim i trenutnim projektima [5, 15, 16].

Za takvu razinu kontrole potreban je sustavan pristup vladanja projekatima, što znači sustavan nadzor i kontrolu usklađenosti napretka projekta sa organizacijskim ciljevima. Veliki problem u softverskim organizacijama predstavlja i sustavno upravljanje kompetencijama radnika. Tehnologije u računalnoj industriji brzo se mijenjaju i teško je održavati korak sa istima. Gotovo svakodnevno se postižu nova dostignuća, razvijaju se nove tehnologije i ažuriraju trenutne. Usprkos brzini razvoja tehnologija organizacije moraju nesmetano izvoditi projekte sa uvijek dostupnim resursima ljudi koji će biti sposobni za razvoj svog sljedećeg planiranog projekta i isporučiti ga bez nepotrebnog gubljenja vremena i dodatnih kašnjenja. Istovremeno, organizacije razvoja moraju planirano provoditi edukaciju ljudi kako bi imala dovoljno vještina za nadolazeće projekte.

Projekti se organiziraju na manje dijelove tj. pakete koji se fokusiraju na razvoj određene značajke (engl. feature) u projektu (u ovom slučaju softveru). Te značajke mogu ovisiti jedna o drugoj i tako se određuje vremenski redoslijed obavljanja istih. Značajke koje nisu međusobno povezane mogu se izvršavati paralelno uz uvjet da su njihove uvjetne značajke već izvršene [17, 18].

3.1. Upravljačke uloge u organizaciji

U ovakvim organizacijama nalazi se više različitih upravljačkih uloga. Ovdje će se opisati samo one koje koristimo u Web aplikaciji.

Portfolio upravitelj ima zadatak selekcije i prioritizacije projekata i programa. Ima uvid u sve projekte, te može vidjeti koji kasne i njihovu kritičnost kašnjenja. Njihov zadatak je optimizirati projekte koji kasne, planirati vremenske rokove, te dodavanje različitih resursa poput ljudi na neki projekt da bi se vremenski rokovi zadovoljili.

Uvid u ljudske kompetencije i planiranje upotrebe kompetencija ljudi za buduće projekte izvršava linijski upravitelj (engl. Line manager). Svakom različitom grupom ljudi upravlja jedan linijski upravitelj. Grupe ljudi raspoređuju se ovisno o smjeru njihovih kompetencija. Linijski upravitelj odobrava i šalje zaposlenike na različite projekte ovisno o uvjetima koje upravitelj projektima traži.

Navigacijska skupina (engl. Steering Commitee) pomaže usmjeravanje projekta u smjeru završetka u slučaju da dođe do nekih neplaniranih neizbježnih događaja, nudeći različite savjete i rješenja za poboljšanja puta prema cilja.

Programski upravitelj (engl. Program manager) provodi različite analize nad trenutnim projektima. Analizira aspekte kašnjenja, korištenja resursa, prednosti i svrstava sve u prikaz ishoda rezultata svih tih podataka.

Upravitelj projektima (engl. Project manager) stvara i uklanja projekte. On planira projekte, određuje njihove dijelove odnosno značajke, vrijeme trajanja projekata, status značajki a time i status izvršenosti projekta. Može upravljati odgodama značajki zbog neplaniranih situacija i zatražuje resurse od linijskog upravitelja, te količinu resursa određenih uvjeta za svaku značajku.

Strateški ured upravitelja projekata (engl. Project Management Office) provodi različite zadatke kako bi unaprijedio učinkovitost izvršavanja strateških ciljeva. Mjeri korporativne strategije i razine zrelosti projekta koji se provode u organizacije te provjerava njihovu

usklađenost sa strateškim ciljevima. Također može imati zadatak organizacije portfolija projekata.

Taktički ured upravitelja projekata (PMO) kontrolira usklađenosti korištenih standarda i procesa rada s definiranim poželjnim standardima i procesima rada. Mogu osposobljavati različite zaposlenike za potrebne kompetencije, većinom su fokusirani osposobljavanje projektnih upravitelja.

3.2. Upravljačke tehnike

U ovom poglavlju opisati će se upravljačke tehnike koje se koriste u organizacijama za vladanje projektima a koje smo razvili u sklopu Web aplikacije za vladanje projektima. Od korištenih tehnika ovdje će se opisati PERT dijagram, izračun kritičnog puta te tehnologija prikaza statusa napretka projekta pomoću tzv. Balanced Scorecard grafa.

3.2.1 PERT dijagram

PERT ili „Program Evaluation Review Technique“ je tehnika koja se koristi kod upravljanja projektima koja nam pomaže vizualizirati podatke odnosno aktivnosti nekog projekta u procjenjivanje rizika kašnjenja aktivnosti na osnovu međusobno povezanim odnosima aktivnosti, u smislu koja aktivnost prethodi kojoj. PERT dijagram jako je koristan alat kada imamo veze između aktivnosti i mnogo tih aktivnosti da se lako mogu organizirat prikazat na pregledan način. Primjer PERT dijagrama u ovoj web aplikaciji je graf aktivnosti tj. značajki na panelu upravitelja projektima, na istom grafu kasnije se vrši računanje kritičnog puta. PERT tehniku ima elemenate poput Vrhova, Lablea, i Bridova. U referenci [5] vrhovi su definirani kao čvorovi grafa i predstavljaju zadatke a labele vrhova prikazuju trajanja zadataka. Bridovi (veze) grafa povezuju zadatak sa svim zadacima koji o njmu ovise.

PERT tehnika se izrađuje u dva koraka kao što je prikazano na slici 3.

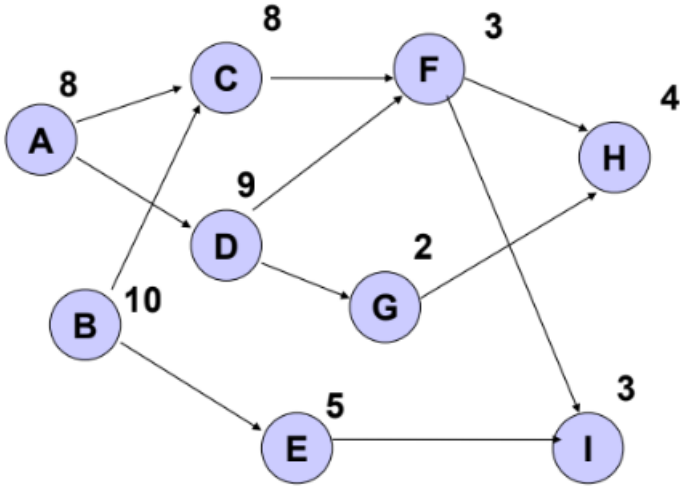
1) **Prvi korak je kreiranje tablice trajanja i ovisnosti zadataka koja se sastoji od 3 stupca:**
    a. Ime zadatka (koriste se mala slova)
    b. Predviđeno trajanje aktivnosti
    c. Uvjetni zadaci koji se moraju izvršiti prije nego trenutni započne.

I	T	P
A	8	
B	10	
C	8	A, B
D	9	A
E	5	B
F	3	C, D
G	2	D
H	4	F, G
I	3	E, F

Slika 4 Prikaz tablice trajanja i ovisnosti (preuzeto iz [5])

2) **Drugi korak je iscrtavanje grafa ovisnosti kao što je prikazano na slici 4.**
    a. Graf se crta od vrhova koji predstavljaju neovisne aktivnosti
    b. Zatim se crtaju aktivnosti koje ovise samo o njima i između njih se kreiraju veze

c. Prethodni koraci se ponavljaju dok se ne iscrtaju sve aktivnosti

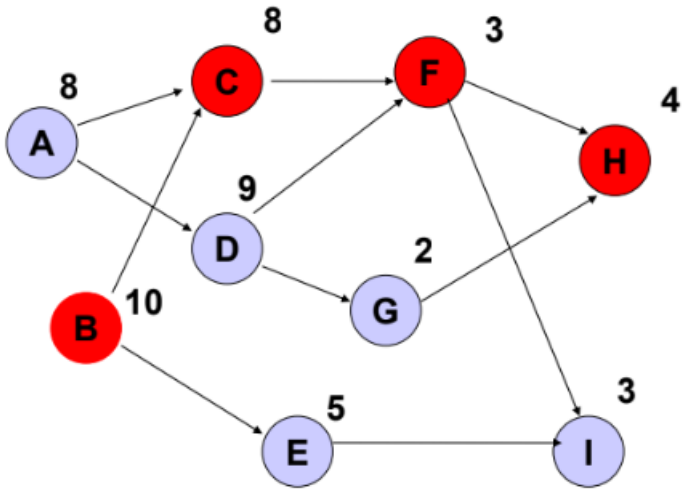


Slika 5 Prikaz grafa ovisnosti (preuzeto iz [5])

3) **Treći korak je izračun dozvoljenih vremena kašnjenja.**
   Dozvoljena kašnjenja se izračunavaju za svaku aktivnost tako da se odredi put sa aktivnosti od početka projekta do kraja sa najdužim trajanjem pa se dozvoljena kašnjenja svake aktivnosti računaju usporedo sa tim putom.

3.2.2 Kritični put

Kritični put (engl. Critical path) je put odnosno redoslijed izračunat na temelju međusobnoj ovisnosti aktivnosti (u ovom slučaju značajki) i njihovih vremena trajanja. [5] Prikazuje nam najduži mogući redoslijed aktivnosti pomoću kojeg bi projekt morao završiti u predviđeno vrijeme [19]. Na temelju PERT prikaza ovisnosti aktivnosti može se odrediti dozvoljena kašnjenja u izvođenju svake od aktivnosti te kritični put odnosno sve aktivnosti koje leže na kritičnom putu su one koje ne smiju kasniti kako se ne bi produžilo vrijeme trajanja projekta, dok ostale aktivnosti su o aktivnosti koje se izvode paralelno sa kritičnim mogu kasniti jer njihovo dozvoljeno kašnjenje ne utječe na završetak projekta [20].



Slika 6 Prikaz kritičnog puta (preuzeto iz [5])

4. OPIS I IZRADA WEB APLIKACIJE

4.1. Opis i zahtjevi aplikacije za vladanje projektima

Kako bih se ostvario sustav vladanja projektima zahtjev aplikacije je ostvarivanje više točaka upravljanja nekom organizacijom. Potreban je odjel koji se bavi zapošljavanjem radnika koje dijelimo na neke radne grupe i dodjeljujemo radnicima neke sposobnosti koje posjeduju. Svaka grupa mora imati glavnog jednog upravitelja tj. linijskog upravitelja. Linijski upravitelj mora imati mogućnosti uvida u sve zaposlenike koji se nalaze u njegovoj grupi, mora imati uvid u sve zahtjeve koje mu šalje upravitelj projekata i mogućnost odgovaranja na iste. Kao važan dodatak poželjno je da linijski upravitelj ima uvid o potrebama kompetencijama svojih radnika kako bi ih mogao pripremiti postojeće ili podnesti zahtjev za zapošljavanje novih. Upravitelj projektima mora imati mogućnosti dodavanja novi projekata i uvid u postojeće projekte. Mogućnost određivanja datuma početka projekta. Dijeljenje projekta u mnogo malih značajki po posebnim specifikacijama potreba. Nakon kreiranja značajki tu je i potreba za generiranjem nekog redoslijeda izvršavanja istih, te zatraživanje određenih zaposlenika koji će izvršiti te značajke.

4.2. Opis web aplikacije

Za opis web aplikacije počinjemo s opisom baze podataka kako bi razumjeli osnovnu strukturu svih podataka i kako su oni međusobno povezani. Nakon baze podataka opisuje se back-end dio aplikacije kako bi dobili bolji uvid u obradu spremanje i dohvaćanje podataka, te nekoliko dodatnih funkcija koje se trebaju izvršiti. Web aplikacija radi na temelju komunikacije poslužitelj-klijent, gdje poslužitelj nudi klijentu sve podatke dok ih klijent samo zatražuje tj. šalje zahtjeve.

4.3. Baza podataka

Kod baze podataka izgrađeno je šest različitih tablica odnosno u MongoDb pod nazivom kolekcija. U njih ubrajamo „work_groups“; ovdje se spremaju radne grupe zaposlenika, „empployees“; u nju se spremaju zaposlenici i sve njihove informacije, „projects“: koleikcija projekata, „features“; ovdje se nalaze informacije svih značajki, „position_requests“ zahtjev koji se šalju između projektnog i linijskog upravitelja i „criticalPaths“; u koju se spremaju svi generirani kritični putevi.

4.3.1. Work_groups

Ova kolekcija sadrži polja svog identifikacijskog broja tj. ID-a kao primarni ključ. Koristi polje „name“ tipa string (niz znakova) koje definira naziv pojedine radne grupe. Treće polje je polje „employees_id“ koje označava identifikacijski broj zaposlenika, u nju se nadodaje identifikacijski broj glavnog linijskog upravitelja neke grupe.

4.3.2. Kolekcija zaposlenici (engl. employees)

Kolekcija zaposlenika sadrži različite informacije zaposlenika imena, prezimena, vještine, i ime grupe, svi ovi podaci su tipa string. Osim osnovnih podataka zaposlenika sadrži i ID grupe koji je zapravo ID pojedine grupe definiran u tablici work_groups i svoj vlastiti ID kao primarni ključ.

4.3.3. Kolekcija projekti (engl. Projects)

Kolekcija projekti sadrži informacije nekog projekta. Informacije uključuju vlastiti ID kao primarni ključ cjelobrojnog tipa (integer), naziv projekta, početni datum, datum kreiranja, datum završetka, te datum kasnog završetka u obliku string, te kašnjenje oblika integer.

4.3.4. Kolekcija značajke (engl. Features)

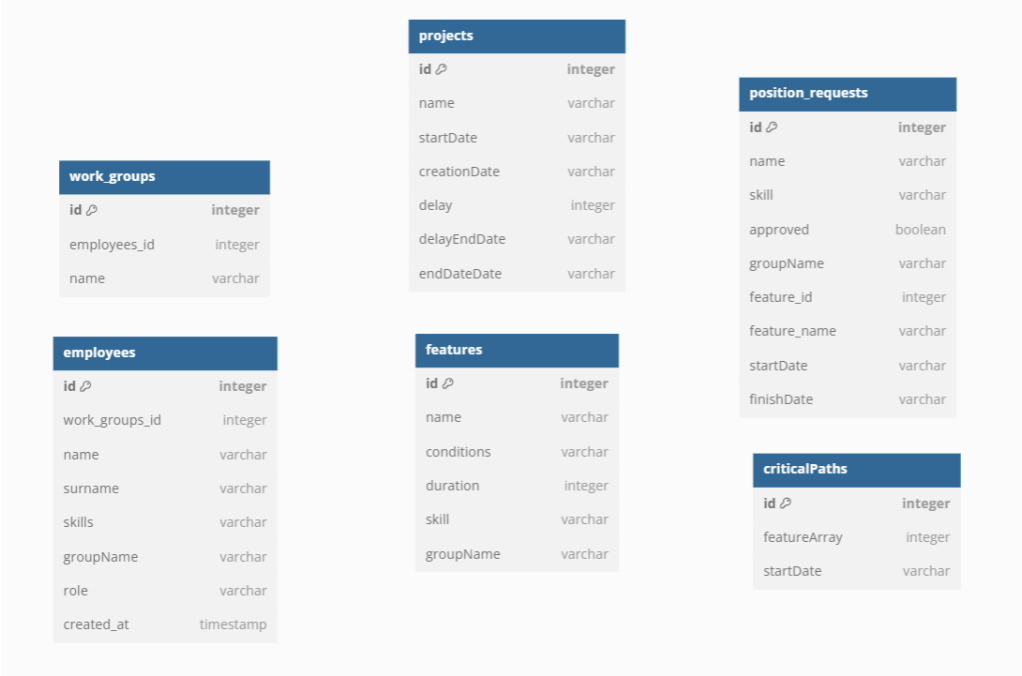
U kolekciji značajke nalaze se svi podaci nekih značajki, počevši od samog ID-a. Ova kolekcija sadrži još i sljedeće informacije: naziv, uvjeti, trajanje, vještine i ime grupe; tipa string.

4.3.5. Izračun kritičnog puta (engl. CriticalPaths)

Izračun kritičnog puta sadrži svoj identifikacijski broj kao primarni ključ, datum početka u obliku stringa, te polje određenih značajki.

4.3.6. Position_requests

Position_requests kolekcija označava kolekciju za zahtjeve između projektnog upravitelja i linijskog upravitelja, a sadrži osnovne informacije poput ID-a, naziva i vještine, te dodatnih kao što su potvrđenost tipa boolean i ime grupe, ID neke značajke, ime značajk, početni i završni datumi tipa string.



Slika 7 Prikaz strukture baze podataka

4.4. Back-end

Serverski dio aplikacije sastoji se od osnovnog dokumenta app.js koji pokreće slušatelja (engl. listener) na portu 5000 te ostale rute poziva pomoću Expressovog routera. Posebno su napravljene rute prema routerima za dohvaćanje i spremanje podataka sa i na bazu podataka. Routeri su raspoređeni na način da je svaki fokusiran na posebnu tablicu tj. kolekciju. Da bi se prenamijenio kod stvoren je još jedan dodatni JavaScript file sa nazivom fetchData.js koji sadrži logiku osnovnih funkcija za izvršavanje zahtjeva i slanje odgovora. Posljednje dvije funkcije jesu db.js koja samo spaja back-end na bazu podataka i calculateCritical.js koja određuje i računa kritični put nad danim podacima.

Kako su ostvarene osnovne HTTP funkcije:

4.4.1. GetData() - funkcija za dobavljanje podataka, funkcija se spaja na bazu podataka i dobavlja podatke sa iste. Funkcija kao dodatan parametar prima filtar pomoću kojeg se iz baze podataka mogu dohvatiti specifični podaci (npr. ako trebamo dohvatiti samo specifični projekt, tako prosljeđujemo ovim parametrom njegov ID)

4.4.2. PostData() – funkcija za spremanje retka podataka, funkcija se spaja na bazu podataka i sprema redak tj. dokument podataka koji su poslani zajedno sa zahtjevom. Ako je uspješno spremljeno program nam vraća odgovor na front-end sa statusom „ok (code: 200) “, na taj način možemo na front-endu vidjeti da li je došlo do greške.

4.4.3. PatchData() – funkcija za promjenu podataka nekog retka, funkcija pomoću ID-a proslijeđenog zahtjevom ažurira proslijeđene podatke u specifičnom redu odnosno dokumentu u bazi podataka. Kao i postData() šalje odgovor na front-end ovisno o statusu izvršenja.

4.4.4. DeleteData() – funkcija brisanje retka podataka, funkcija pomoću određenog ID-a dokumenta briše taj dokument, te kao i ostale funkcije vraća odgovor o uspješnosti izvršenja.

4.5. Front-end

Korisničko sučelje je dizajnirano tako da korisnik prvo bude predstavljen s odabirom od 4 opcije ovisno o njegovoj funkciji:

1) Human resources – Ljudski resursi (Upravljanje zaposlenicima)
2) Line managment – Linijsko upravljanje
3) Project managment – Upravljanje projektima

Trenutno korisnici sami odabiru svoju upravljačku ploču no na to se kasnije dodaje da svaki korisnik ima svoju prijavu koja ga autentifikacijom i autorizacijom „baca“ na pravu upravljačku ploču na kojoj ima ovlasti.

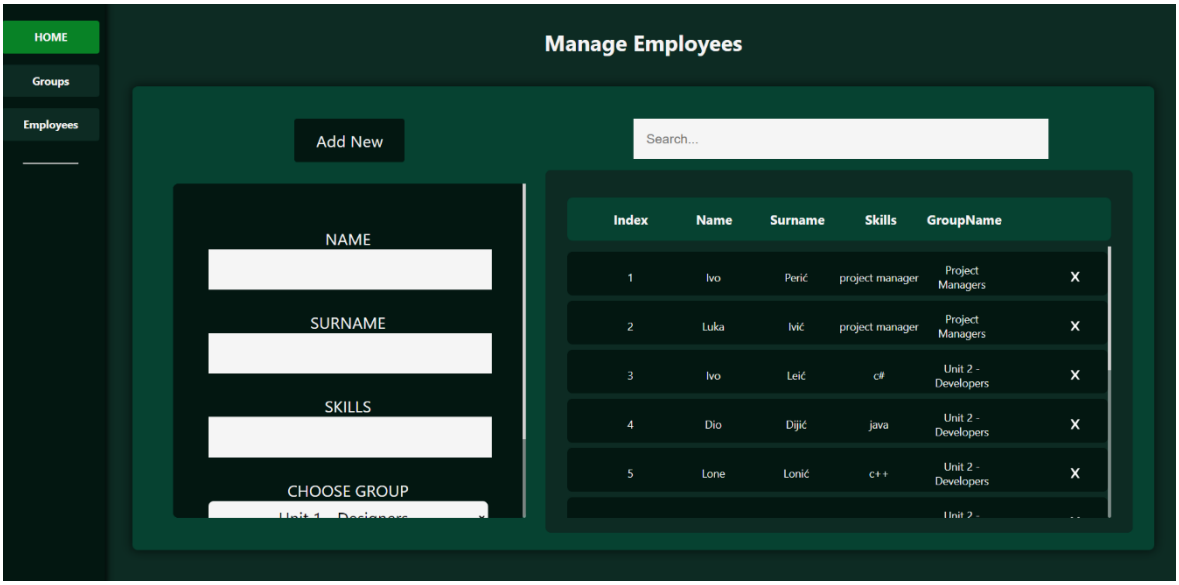
4.5.1. Upravljanje ljudskim resursima

Upravljanje ljudskim resursima služi za uvid, dodavanje i uređivanje zaposlenika, u koju oni radnu grupu spadaju, te koje oni sposobnosti odnosno kompetencije posjeduju. Ova grana vladanja projektima treba imati mogućnost dodavanja zaposlenika tj. zapošljavanja, uklanjanje zaposlenika tj. otkaz, svrstavanje zaposlenika u neke druge organizacijske jedinice. I uvid u pojedinog zaposlenika. Ova upravljačka ploča je izvedena tako da imamo odabir od dvije podgrupe „groups“ (hrv. grupe) i „employees“ (hrv. zaposlenici). Ulaskom u upravljačku ploču inicijalno se automatski ažuriraju i dobavljaju trenutni podaci svih zaposlenika i radnih grupa sa HTTP zahtjevom GET, svaka podgrupa ima svoju listu podataka. Ulaskom u jednu od podgrupa ponovo se dobavljaju podaci, ali samo podaci te podgrupe tako da podaci uvijek ostanu potpuno ažurirani, ovo je velika važnost iz mogućnosti da imamo više upravitelja zaposlenicima u isto vrijeme na posebnim računalima.

Svaka podgrupa ima vrlo slično korisničko sučelje. Kod zaposlenika na lijevoj strani vidimo „Add new“ (engl. dodaj novi) gumb s kojim se otvara forma za dodavanje novih grupa odnosno zaposlenika. Traže se samo osnovne informacije, kod grupa ime grupe, a kod

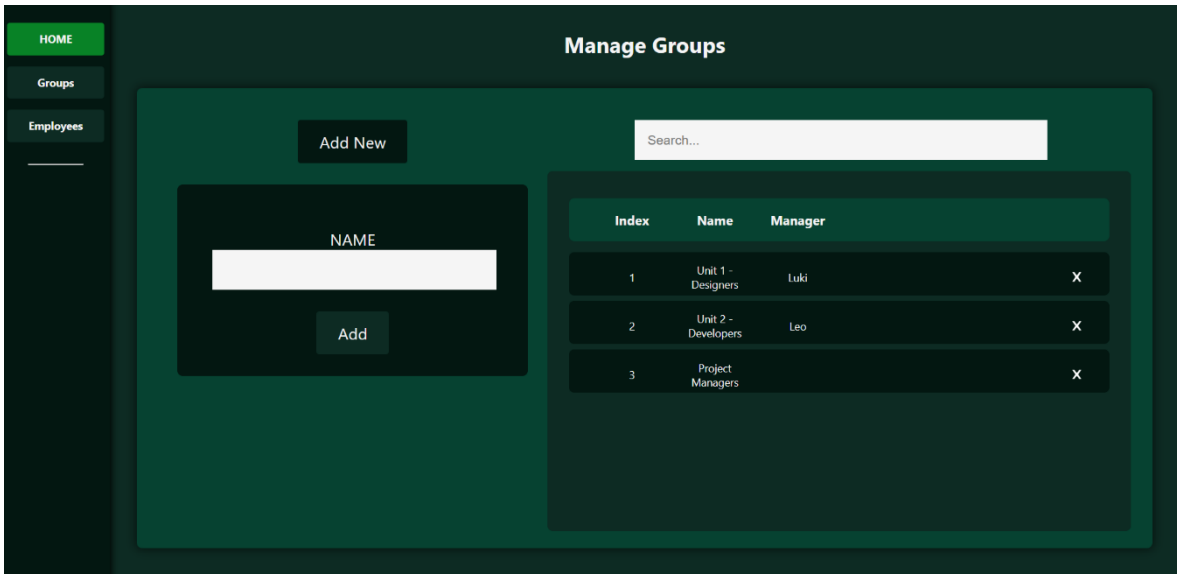
zaposlenika dodatni padajući izbornik s listom postojećih grupa da bih ga svrstali u jednu od njih. U formi se prije slanja podataka provjeravaju neki osnovni uvjeti kao jesu li podaci ispunjeni ili su polja prazna.

S desne strane upravljačke ploče nalazi se lista postojećih grupa odnosno zaposlenika sa nekoliko osnovnih informacija, te gumbom za brisanje istih. U osnovne informacije uključeni su ime, prezime, vještine i ime grupe kojim su ti zaposlenici dodijeljeni. Klikom na gumb za brisanje šalje se novi zahtjev „DELETE“ za brisanje odabranog korisnika pomoću njegova identifikacijskog broja. (Slika 8)



Slika 8 Prikaz HR - Zaposlenici

Kod podgrupe radnih grupa može se primijetiti da nema toliko podataka kao kod zaposlenika. Za dodavanje radne grupe u formi se nalazi samo upis imena grupe, a s lijeve strane možemo vidjeti samo ime grupe, te glavnog linijskog upravitelja koji upravlja tom grupom. (Slika 9)



Slika 9 Prikaz HR - Grupe

Za upis podataka u ljudskim resursima prvo se dodaju radne grupe, te tek onda možemo dodati zaposlenike koje svrstavamo u određene radne grupe kod samog njihovog kreiranja. Dodavanjem zaposlenika koji sadrže kompetenciju linijski upravitelj automatski se prikazuju u radnim grupama kao upravitelj grupe kojoj je dodan.

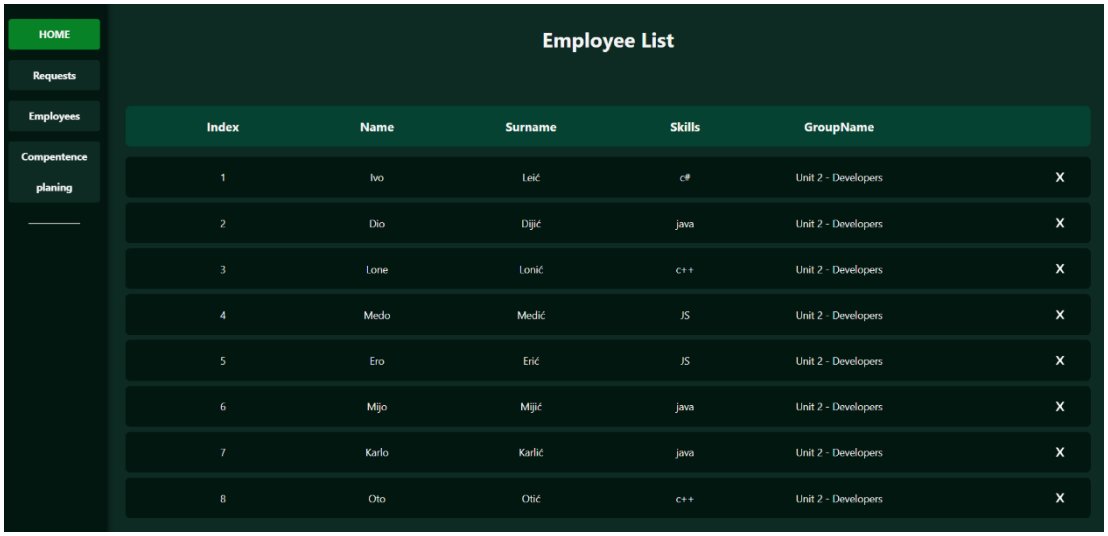
4.5.2. Linijsko upravljanje

Linijski upravitelj kao što je već prije spomenuto dodijeljen je nekoj grupi zaposlenika. Nad dodijeljenom grupom ima potpun nadzor zaposlenika, nadzor nad zahtjevima koji dolaze od upravitelja projekata, te grafički prikaz dostupnosti zaposlenika i njihov raspored prema sposobnostima.

U prvoj kartici tj. podgrupi linijskog upravitelja možemo vidjeti sve zahtjeve koje nam šalje upravitelj projekata u kojima upravitelj projekata potražuje zaposlenike po vještinama potrebnim za izvedbu neke značajke iz projekta. U tom zahtjevu također je određeno u koje točno vrijeme je zaposlenik potreban na projektu, i na koliko dugo. Prema informacijama zahtjeva linijski upravitelj dodjeljuje traženu količinu slobodnih zaposlenika u tom određenom vremenu sa tom određenom kompetencijom. (Slika 10)

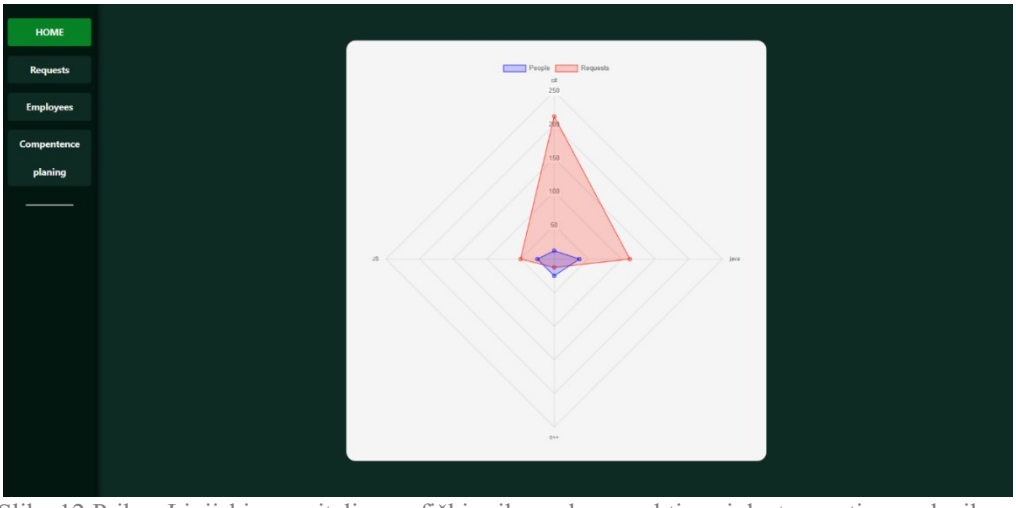
Slika 10 Prikaz Linijski upravitelj - zahtjevi

U drugoj kartici linijski upravitelj ima uvid u sve zaposlenike koji se nalaze u njegovoj grupi, a vrste informacija identične su onima koje ima i sam upravitelj zaposlenicima (Human resources). (Slika 11)



Slika 11 Prikaz Linijski upravitelj – zaposlenici grupe

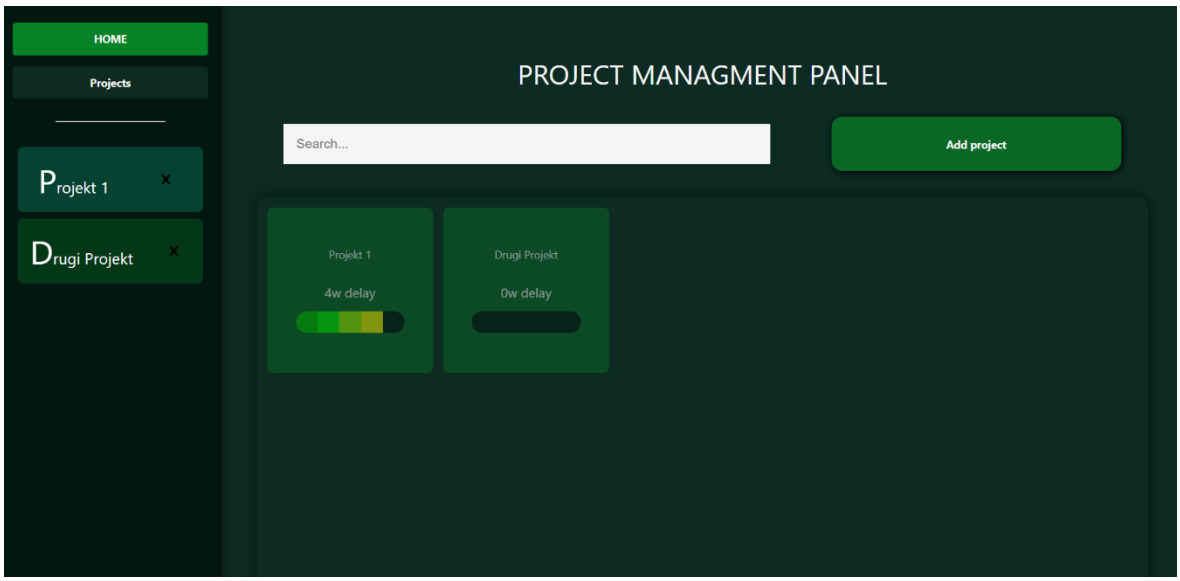
Treća kartica linijskom upravitelju služi za planiranje razvoja kompetencija zaposlenika, daje grafički uvid u postocima trenutno zaposlen postotak ljudi na temelju njihovih kompetencija (označeno plavom bojom na grafu), te postotak potrebe ljudi prema zatraženim zahtjevima za određenim kompetencijama (označeno crvenom bojom na grafu). Graf je oblika radara, a prikazan je pomoću biblioteke Chart.js. Nakon importa JavaScript biblioteke stvaramo graf na posebnoj komponenti i šaljemo mu podatke koji će se koristiti. Na slici u prikazu možemo vidjeti da je crveni dio puno veći što znači da ima premalo ljudi sa tim kompetencijama koliko se traži za nadolazeće projekte, to daje informaciju linijskom upravitelju da pokrene zahtjev za edukacijom ljudi sa nedostajućim kompetencijama ili u nedostatku slobodnih ljudi alarmiranje odjela za upravljanje zaposlenicima za zapošljavanje ljudi sa traženim kompetencijama. (Slika 12)



Slika 12 Prikaz Linijski upravitelj – grafički prikaz odnosa zahtjeva i dostupnosti zaposlenika

4.5.3. Upravljanje projektima

Kod upravljačke ploče *Project Management* prikazana je lista već postojećih projekata. Tu listu možemo pretraživati prema imenu pomoću polja za pretraživanje iznad same liste. Pokraj polja za pretraživanje nalazi se i gumb za dodavanje novog projekta. (Slika 13)

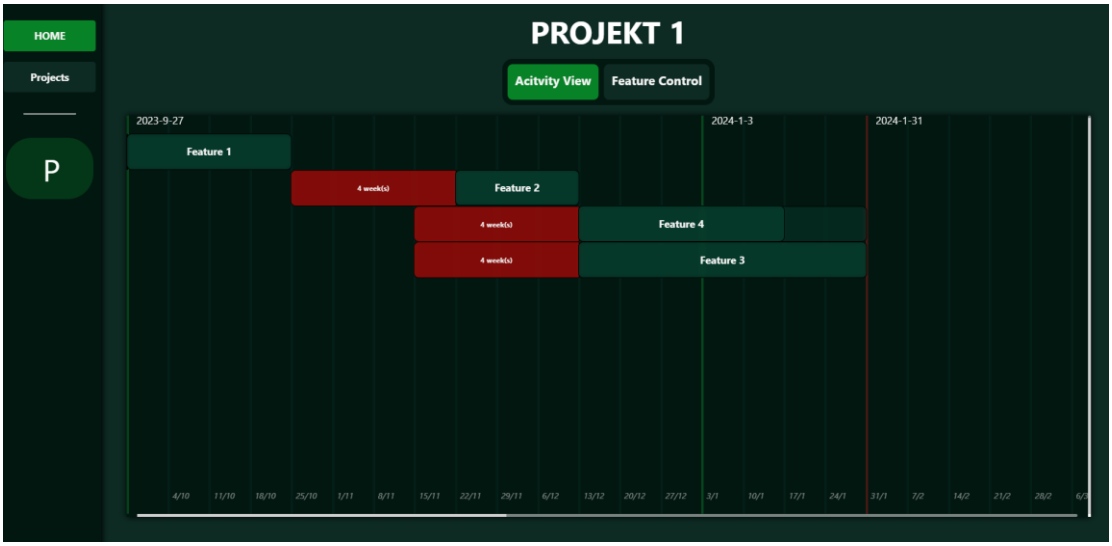


Slika 13 Prikaz Upravitelj projekatima – lista projekata

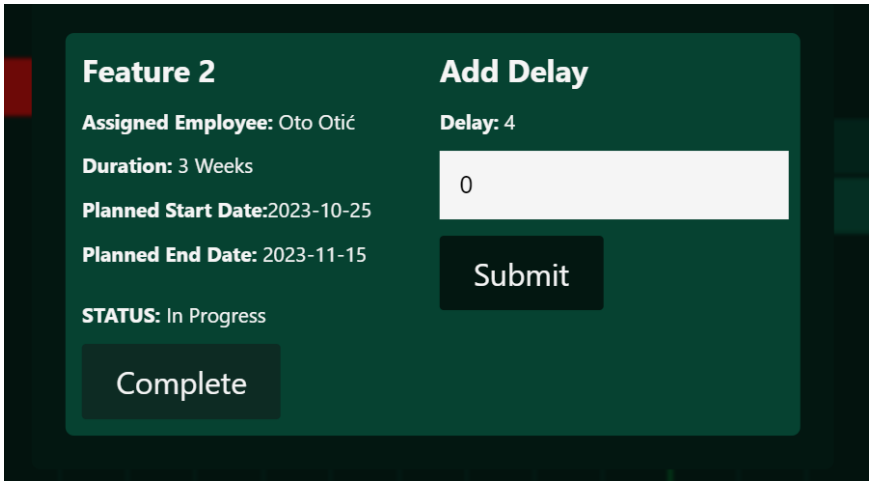
Elementi liste projekata sastoje se od naziva projekta i ispod njih podatak koliko tjedana projekt kasni, sa dodatnim grafičkim prikazom kašnjenja. Grafička linija određena je sa pet različitih boja (od zelene do crvene) koje se uključuju ovisno o duljini kašnjenja, sve od pet tjedana nadalje označeno je crvenom. Prikaz stanja svih projekata na istom pregledniku omogućava strateški nadzor i usklađivanje sa ciljevima organizacije.

Klikom na jedan element liste, odnosno projekt, sa lijeve strane na navigacijskoj traci otvara se određeni projekt i prikazuje prozor s detaljima projekta.

Ako postoji već generirani kritični put tog projekta on se prikazuje na prvoj kartici „Activity View“ (hrv. pogled aktivnosti) u obliku grafa sa vremenskim intervalima trajanja, ranih i kasnih početaka, te ranih i kasnih završetaka svake „značajke“ (engl. feature). Klikom na jednu od značajki otvara se mali prozor sa nekoliko osnovnih korisnih informacijama svake značajke, statusom te značajke i zaposlenikom koji radi na njoj. Osim informacija ovaj prozor omogućuje nam upravljanje nad značajkom odnosno možemo označiti ako je značajka gotova, te možemo dodati kašnjenje. Kašnjenja se na grafu prikazuju crvenom bojom u obliku sjene svake značajke, te kašnjenje jedne značajke utječe na kašnjenje svih sljedećih koje su njome povezane. (Slike 14 i 15)

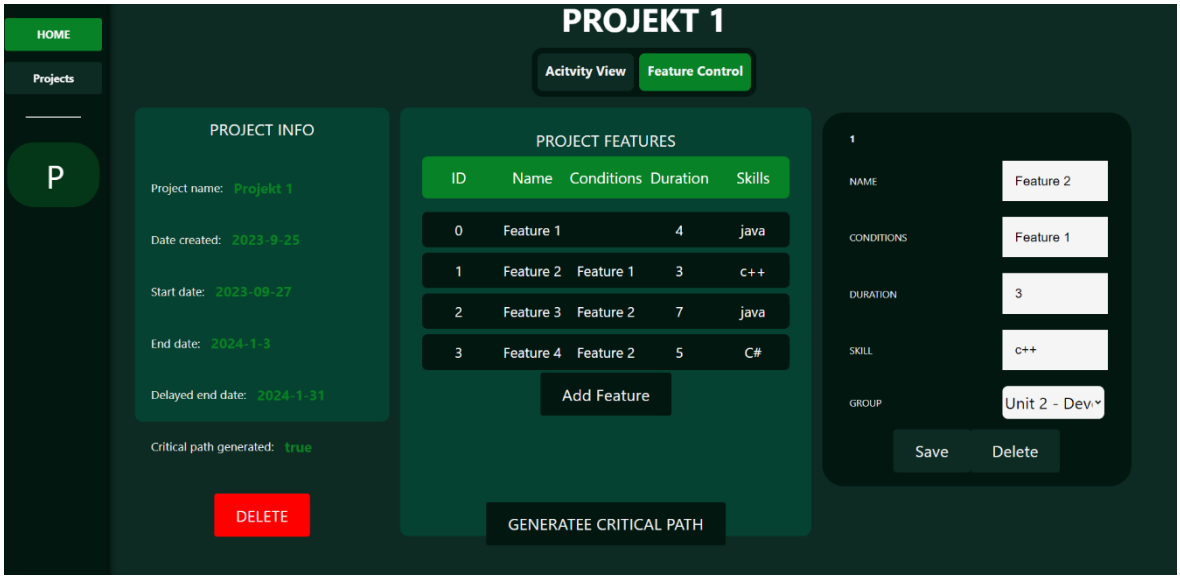


Slika 14 Prikaz Upravitelj projekatima – graf značajki



Slika 15 Prikaz Upravitelj projekatima – uređivanje značajke

Na drugoj kartici „Feature Control“ (hrv. kontrola značajki) možemo naći više informacija o trenutnom projektu koja nas informira o imenu projekta, njegov planirani početni datum, planirani završni datum, završni datum ako se dogode nekakva kašnjenja sa izvršavanjem zadatka, te je li kritični put već generiran ili ne. Na ovoj kartici možemo pronaći i listu značajki pojedinog projekta koje možemo dodati, urediti i brisati. Pri dodavanju značajki potrebno je definirati ime značajke, značajke koje se moraju izvršiti prije nje kao uvjeti, trajanje značajke, vještinu odnosno sposobnost koja ta značajka treba da bi je neki zaposlenik izvršio i odabir grupe kako bi se ista kasnije poslala kao zahtjev pravom linijskom upravitelju. (Slika 16)



Slika 16 Prikaz Upravitelj projekatima – prikaz upravljanja značajkama

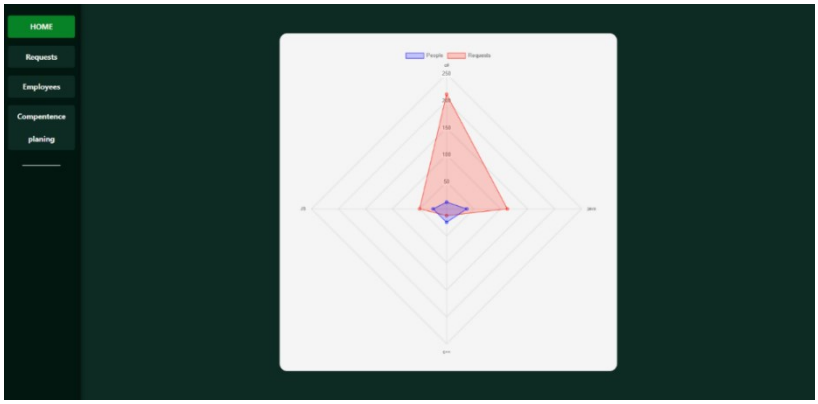
Na dnu okvira značajki nalazi se gumb „Generate critical path“ (hrv. generiraj kritični put). Klikom na gumb odvija se proces slanja liste značajki trenutnog projekta na back-end gdje se računa kritični put koji se sprema u bazu podataka i odmah nakon toga dohvaća na front-end kako bi se prikazao u obliku grafa. Nakon izračuna kritičnog puta generiraju se zahtjevi za linijske upravitelje, generira se jedan zahtjev po značajki u kritičnom putu.

5. PRIMJERI REALISTIČNIH SCENARIJA

U ovom poglavlju prikazat ćemo nekoliko mogućih realističnih problema iz upravljanja organizacijom, te kako ih ova web aplikacija može riješiti.

5.1. Strateško planiranje kompetencija

Nove tehnologije nekontrolirano se pojavljuju na tržištu i da bi se održao korak na tržištu proizvoda koje jedna organizacija razvoja softvera plasira, ta organizacija mora stalno ulagati u edukaciju svojih zaposlenika u novim tehnologijama. Istovremeno, zaposlenici moraju obavljati poslove u tekućim projektima. Kada se redovito ne uspijeva educirati zaposlenike zbog kontinuiranog preopterećena zaposlenika u tekućim projektima jedan linijski upravitelj može se naći u vrlo zbunjujućoj situaciji. Otvorivši svoj panel zahtjeva ugleda nove zahtjeve za određenu vrstu vještina koja je nedostajuća u njegovoj grupi zaposlenika a istovremeno potražnja iz dana u dan raste. Linijski upravitelj ni sam nije siguran ima li dovoljno zaposlenika sa traženim vještinama u narednim projektima pod svojom linijskom organizacijom kojom upravlja. Pomoću web aplikacije vladanje projekata linijski upravitelj može osmišljavati plan razvoja kompetencija djelatnika u svojoj organizacijskoj jedinici. Na jednostavan klik na karticu „Competance planing“ upravitelju se otvara automatsko dinamično generirani radar dijagram. Dijagram prikazuje dvije vrste podataka, plavom bojom označeno je relativan postotak vještina koje djelatnici iz organizacijske jedinice kojom upravlja posjeduju, te crvenom površinom označene su potrebe nadolazećih projekata iz njegove grupe koji je potreban za buduće projekte.

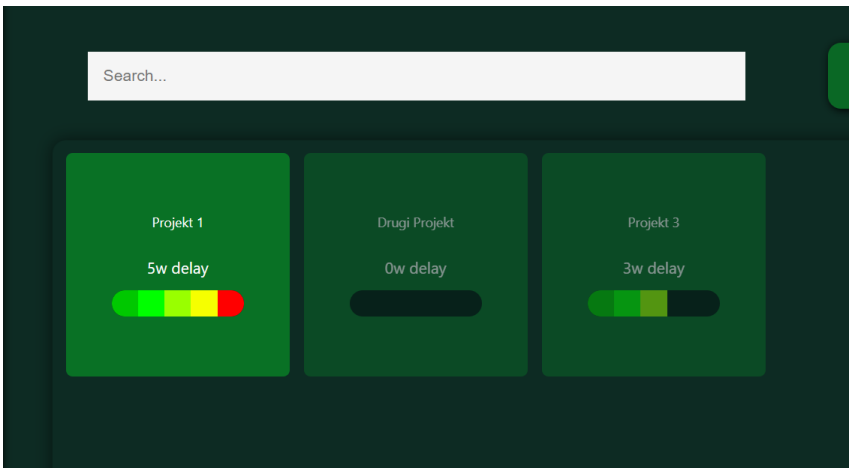


Slika 17 Prikaz radar dijagrama

Ako graf bude kao sljedeći (Slika 17) linijski upravitelj mora hitno poslati svoje ljude na usavršavanje u nedostajućim vještinama ili ukoliko nema trenutno slobodnih ljudi za usavršavanje treba obavijestiti ljudske resurse da se zaposle novi. Na sljedećem grafu jasno je vidljivo da zbog brzog širenja tvrtke na nova tržišta sama potražnja ove radne grupe i specifičnih kompetencija je znatno porasla i prerasla broj zaposlenih i njihovih mogućnosti.

5.2. Vladanje projektima i nadzor usklađenosti sa strateškim ciljevima organizacije

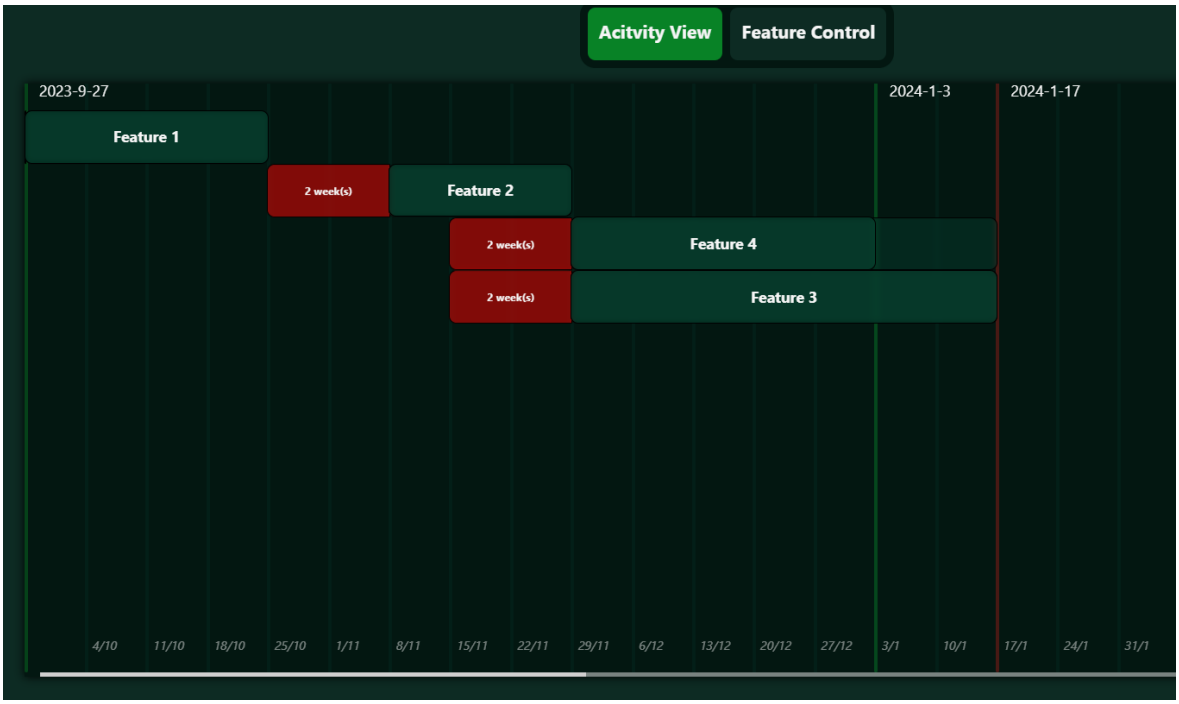
Portfolio upravitelj pojavio se u situaciji da se u njegovoj organizaciji nalazi previše projekata koji se izvršavaju u isto vrijeme da bi ih on stigao sve pregledati i kvalitetno nadzirati. Nadalje, pregled statusa svih projekta potreban je kako bi se uvidjelo ide li oragnizacija prema ostvarivanju strateških ciljeva i ukoliko ne ide kolika su odstupanja u mjerljivim indikatorima. Da ne bi morao otvarati svaki projekt njemu su samo potrebni pod nadzor samo kritični. Kako bi imao lakši uvid u kritične projekte preko web aplikacije vladanje projektima, projektni upravitelj može koristiti alat „scorecard“ tj. liniju koja prikazuje kritičnost već pri pregledu liste svih projekata. Linija kritičnosti ovisi o tome koliko pojedini projekt kasni u tjednima. Tako ne trošeći previše vremena ima trenutni uvid u sva kašnjenja i stanja projekata. Prikaz kritičnosti Slika 18.



Slika 18 Prikaz kritičnosti

5.3. Nadzor projekta i procjena rizika kašnjenja

Projektni upravitelj dodaje, uređuje i briše samostalno značajke nekog projekta. Nakon što mu se dodjeli projekt, projektni menadžer planira aktivnosti u projektu tako što unosi sve značajke koje su planirane za isporuku projektom na koji je dodijeljen. Isto tako za svaku značajku definiraju se preduvjeti (značajke koje moraju biti dovršene da bi se određena značajka mogla započeti razvijati), i spreman je započeti projekt, projektni upravitelj na klikom na gumb može lako i jednostavno generirati kritični put odnosno redoslijed značajki kojim se moraju izvršavati da vrijeme projekta ne bi kasnilo. No ipak, usprkos računalnom računanju savršenog rasporeda vremena, u realnosti se uvijek događaju nepredvidive situacije koje mogu dovesti do kašnjenja u razvoju značajki. Za vrijeme nadzora tijeka odvijanja projekta, tj. može unositi kašnjenja u izvedbi određenih značajki. Projektni upravitelj može odrediti neko kašnjenje koje se trenutno prikaže u crvenoj boji kao „trag“ značajki. Pomoću takvog sustava dobro je vidljivo koliko zapravo pojedina značajka kasni i kako međusobno značajke utječu jedna na drugu. Pomicanjem vremena značajki pomiče se i izvršno vrijeme projekta, što nam omogućuje dodatnu mogućnost određivanja rizika kašnjenja u smislu određivanja troška u danima zauzetosti zaposlenika uslijed čekanja na preduvjete za određivanje svoje značajke. Pored toga omogućeno je replaniranje kako će se dalje taj projekt izvršiti. Na slici 19 prikazan je PERT dijagram nakon što je projektni upravitelj izvršio nadzor i ustanovio kašnjenja kod određenih značajki. Takvim uvidom može se procjenjivati rizik kojeg kašnjenje u isporuci određene značajke unosi na cijeli projekt.



Slika 19 Prikaz PERT dijagrama

6. Pokretanje aplikacije na vlastitom računalu/serveru

Kako bi se aplikacija pokrenula na vlastitom računalu potrebno je poduzeti nekoliko sljedećih koraka:

1) Instalacija Node js paket upravitelja
   ➔ `sudo yum install nodejs`

2) Instalacija Reacta
   ➔ `sudo npm install -g create-react-app`

3) preuzimanje front-enda/back-enda sa github repozitorija čiji je link objavljen na samom kraju ovoga rada
   ➔ git clone

4) Za lakše pokretanje back-enda dobro je instalirati i nodemon, ali nije nužno
   ➔ `sudo npm install -g nodemon`

5) Back-end aplikacija pokreće se na naredbu
   ➔ `nodemon app.js`; na localhostu na portu 5000

6) Front-end aplikacija pokreće se naredbom
   ➔ `npm start`; na localhostu na portu 3000

7. MOGUĆI BUDUĆI RAZVOJ APLIKACIJE

Budućnost aplikacije leži u njenoj samoj nadogradnji, osnovna baza aplikacije je potpuno izvršena što nam omogućuje vrlo laku nadogradnju s mogućnošću prenamjene gotovo svega koda. Kao što je spomenuto web aplikacija izrađena je u Reactu koji je vrlo fleksibilan sa prenamjenom svojih komponenata. Komponente poput lista i formi mogu se lako pozvati sa svojim dinamičkim prijenosom podataka. Back-end dio aplikacije također je izrađeni s tehnikom prenamjene u planu, tako cijeli back-end ima zapravo samo pet različitih dinamičkih funkcija koje odgovaraju dohvaćanje i postavljanje podataka na bilo koju tablicu odnosno kolekciju baze podataka.

Nadogradnja se može razgranati u nekoliko smjerova. Počevši od proširivanja same aplikacije na još više panela za upravljanje nad nekom tvrtkom. Primjer novog panela bio bi panel za upravljanje procesima omogućujući korisnicima veću kontrolu nad tvrtkom i bolju integraciju jedne točke upravljanja sa drugom.

Smjer nadogradnje postojećih elemenata bio bi drugi smjer budućnosti ove web aplikacije. U ovom smislu može se dodati i popraviti mnogo stvari. Mogućnost popravka nekakvih kvarova u aplikaciji i nepotpune funkcije nekih značajka, mogućnost dodavanja dodatnih alata poput drugačijih grafova za pregled različitih podataka.

Nadogradnja sigurnosti, u ovom smislu dodavanje autorizacije i autentifikacije korisnika. Na taj način se otvara nova mogućnost pristupa samom sistemu sa interneta, a ne već samo sa lokalnog servera.

Mogućnosti dodatnih komunikacija, gdje se razvija komunikacija između samih upravitelja, ali i ostalih zaposlenika. Za primjer može se dodati chat, kreiran na razini neke grupe, projekta ili pak između dvije osobe. Dodavanje sučelja za ostale zaposlenike bila bi dodatna mogućnost gdje bi zaposlenici mogli učitati rezultate svojih zadataka i na temelju feedback sistema ispraviti svoje pogreške.

Osim funkcionalnih dodataka mogu se poboljšati i vizualni vizualnom optimizacijom pomoću CSS-a.

8. ZAKLJUČAK

Veliku ulogu ove aplikacije igra to što je ona zapravo web aplikacija. Web aplikacije ne treba instalirati i ako se nalazi na internetu možemo je pristupiti od bilo gdje imamo pristup internetu. Osim pristupa iz bilo koje lokacije, dodatak je i pristup sa bilo kojeg uređaja. Web aplikacije puno su prilagodljivije s velikog ekrana na mali nego neke računalne/mobilne aplikacije, što znači mogućnost korištenja istih proširuje se sa računala i na mobitele, te tablete ili pak na bilo koji uređaj koji može pristupiti web pregledniku i internetu. Današnje Web tehnologije omogućuju interaktivne Web aplikacije koje se brzo ažuriraju sa novim stanjima i na taj način dodatno zadovoljava korisničke potrebe s pravovremenim informiranjem o stanju primjene aplikacije.

Aplikacija izvršava sve osnovne funkcije sa potpuno uređenom bazom za njen daljnji razvoj. Kako idejama nema granica tako nema ni mogućnostima ove aplikacije, od implementacije sigurnosti do raznih novih značajaka i komunikacijskih protokola. Korištenje dobro poznatih i razvijanih tehnologija poput React-a, JavaScript-a, HTML-a i CSS-a omogućuje aplikaciji laku i jednostavnu prilagodljivost na neke specifične zahtjeve organizacija.

Model vladanja projektima ima veliku ulogu u poboljšanju komunikacije između različitih uloga u nekoj organizaciji a time i samoj uspješnosti neke organizacije, a pomoću web tehnologije mogućnost njegova pristupa kroz ovakvu aplikaciju postaje široka i jednostavna. Dostupnost ovakvom alatu donosi samoj aplikaciji jako veliku prednost i potencijal za daljnji razvoj. Nadalje, pravovremena informiranost svih sudionika putem zajedničkog sučelja doprinosi transparentnoj komunikaciju zaposlenika u nekoj organizaciji. Neke ideje ka budućem razvoju ove aplikacije idu prema ugrađivanju simulacijskih mogućnosti u rad aplikacije te rane identifikacije mogućih problema i rizika za neku organizaciju. Rano otkrivanje rizika jedan je od ključnih preduvjeta opstanka na dinamičnom konkurentom tržištu softverskih proizvoda.

9. LITERATURA

[1] Shaikh, T. H., Shah, H. N., Khan, F. L., Shaikh, N. A., & Pirani, D. Z. (2018.*). Survey of Web-Based Project Management System*. In *International Conference on Smart Systems and Inventive Technology (ICSSIT 2018.)*. IEEE Xplore Part Number: CFPI8P17-ART. ISBN: 978-1-5386-5873-4.

[2] Müller, R. (2009.). *Project governance* (Fundamentals of project management). *Gower Publishing Limited*.

[3] Pressman, R.S. and Maxim, B.R. (2014) Software Engineering: A Practitioner's Approach. 8th Edition. McGraw-Hill Inc., New York, NY, USA.

[4] Robbins Niederst, J. (2012.). *Learning Web Design* (JavaSctipt/DOM scripting) (str. 13). *O'Reilly Media*

[5] Galinac Grbac, T. (2022.). *Planiranje projekata razvoja programskog proizvoda*, PowerPoint prezentacija sa predavanja.

[6] Fran. (2023, Lipanj 3). *What are HTML and CSS used for? The basics of coding for the web,* Future Learn. https://www.futurelearn.com/info/blog/what-are-html-css-basics-of-coding, pristupljeno: 10.9.2023.

[7] (n.d.). *What is JavaScript?,* Mdn web docs. https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript ,pristupljeno: 10.9.2023

[8] Yakubu, V. (2022, Prosinac 16). *Complete Guide to DOM and ReactDOM in ReactJS,* CopyCat. https://www.copycat.dev/blog/reactdom/, pristupljeno: 10.9.2023

[9] Smith, J. (2023). .). *React Reference*., React: https://react.dev/reference/react, pristupljeno: 10.9.2023

[10] (n.d.). *Chart.js,* Chart.js. https://www.chartjs.org/ ,pristupljeno: 10.9.2023

[11] (n.d.). *About Node.js,* Node.js. https://nodejs.org/en/about ,pristupljeno: 11.9.2023

[12] (n.d.). *4.x API,* Express.js. https://expressjs.com/en/4x/api.html

,pristupljeno: 11.9.2023

[13] Gillis, A. S. (n.d.). *REST API (RESTful API),* Tech Target Network. https://www.techtarget.com/searchapparchitecture/definition/RESTful-API , pristupljeno: 11.9.2023

[14] (n.d.). *MongoDB Documentation, MongoDB*. https://www.mongodb.com/docs/, pristupljeno: 11.9.2023

[15] Alie, S. S. (2015, Listopad 10). *Project governance,* Project Managment Institute. https://www.pmi.org/learning/library/project-governance-critical-success-9945 , pristupljeno: 11.9.2023

[16] Galinac, Tihana, Focused Competence Planning for Large Scale Software Development, European Systems and Software Process Improvement and Innovation, EuroSPI 2007.

[17] (n.d.). *What is project governance?,* projectmanagementqualification.com. https://www.projectmanagementqualification.com/blog/2019/07/23/project-governance-explained/ , pristupljeno: 12.9.2023

[18] (2023, Srpanj 6). *What is Project Governance?, ADEACA*. https://www.adeaca.com/blog/faq-items/what-is-project-governance/ , pristupljeno: 12.9.2023

[19] Slate A. (2022, Travanj 20). *Critical Path Method for Project Management,* Wrike. https://www.wrike.com/blog/critical-path-is-easy-as-123/ , pristupljeno: 12.9.2023

[20] Levy F. K., Thompson G. L., and Wiest J. D. (1963, September). *The ABCs of the Critical Path Method,* Harvard BUsiness Review. https://hbr.org/1963/09/the-abcs-of-the-critical-path-method , pristupljeno: 12.9.2023

10. SLIKE

1. Prikaz strukture aplikacije – https://www.evertop.pl/wp-content/uploads/2021/03/Front_backend_Obszar-roboczy-1-kopia-24-768x547.png
2. Prikaz komponente sa komponentom unutar nje
3. Komunikacije klijent-poslužitelj - https://darvishdarab.github.io/cs421_f20/docs/readings/client_server/
4. Prikaz tablice trajanja i ovisnosti
5. Prikaz grafa ovisnosti
6. Prikaz kritičnog puta
7. Prikaz strukture baze podataka
8. Prikaz HR - Zaposlenici
9. Prikaz HR – Grupe
10. Prikaz Linijski upravitelj - zahtjevi
11. Prikaz Linijski upravitelj – zaposlenici grupe
12. Prikaz Linijski upravitelj – grafički prikaz odnosa zahtjeva i dostupnosti zaposlenika
13. Prikaz Upravitelj projekatima – lista projekata
14. Prikaz Upravitelj projekatima – graf značajki
15. Prikaz Upravitelj projekatima – uređivanje značajke
16. Prikaz Upravitelj projekatima – prikaz upravljanja značajkama
17. Prikaz radar dijagrama
18. Prikaz kritičnosti
19. Prikaz PERT dijagrama

11. GITHUB REPOZITORIJ WEB APLIKACIJE

Front-end: https://github.com/romeodombaj/project_governance_fe.git

Back-end: https://github.com/romeodombaj/project_governance_be.git

12. SAŽETAK

Završni rad opisuje web aplikaciju na temu *„Vladanje projektima“*. Web aplikacija sastoji se od tri različita upravljačka panela koje koriste upravitelj ljudskih resursa, linijski upravitelj, te projektni upravitelj. Svaki od tih panela koristi osnovne CRUD funkcionalnosti kao što su dohvaćanje podataka, uređivanje podataka, spremanje podataka, te brisanje istih. Podaci su međusobno povezani te to daje aplikaciji veliku prednost u smislu preglednosti, te automatizacije izbora dostupnih podataka. U zadatku su korištene tehnike poput PERT i računanje kritičnog puta kako bi se odredilo završno vrijeme, te vrijeme kašnjenja projekata.

Ključne riječi: web aplikacija, vladanje projektima, PERT, Kritični put, CRUD, upravljanje projektima, linijsko upravljanje, upravljanje ljudskim resursima

13. SUMMARY

The final thesis describes a web application on the topic of "Project Governance." The web application consists of three different control panels used by the Human resources manager, Line manager, and Projectmanager. Each of these panels utilizes basic CRUD functionalities such as data retrieval, data editing, data saving, and data deletion. The data is interconnected, which gives the application a significant advantage in terms of visibility and automation of available data selections. Techniques such as PERT and critical path analysis were used in the task to determine the project's completion time and project delay time.

Key words: web application, project governance, PERT, Critical path, CRUD, project managment, line managment, human resources managment