Informacijski sustav proizvodnih naloga

Petonjić, Karlo

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

Karlo Petonjić

INFORMACIJSKI SUSTAV PROIZVODNIH NALOGA

ZAVRŠNI RAD

Čakovec, 2023.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PRIJEDIPLOMSKI STUDIJ RAČUNARSTVO

Karlo Petonjić

INFORMACIJSKI SUSTAV PROIZVODNIH NALOGA

INFORMATION SYSTEM FOR PRODUCTION ORDERS

ZAVRŠNI RAD

Mentor: dr. sc. Nađ Josip, pred.

Čakovec, 2023.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
ODBOR ZA ZAVRŠNI RAD

Čakovec, 20. veljače 2023.

država: **Republika Hrvatska**
Predmet: **Programsko inženjerstvo i informacijski sustavi**

ZAVRŠNI ZADATAK br. 2022-RAČ-R-90

Pristupnik: **Karlo Petonjić (0313025695)**
Studij: Redoviti preddiplomski stručni studij Računarstvo
Smjer: Programsko inženjerstvo

Zadatak: **Informacijski sustav proizvodnih naloga**

Opis zadatka:

Cilj Završnog rada je opisati principe rada informacijskih sustava za praćenje proizvodnje, gdje se kao osnovni dokument koristi proizvodni nalog. Kao osnovicu za prikaz poslovanja treba koristiti tzv. osnovni proizvodni model koji bi predstavljao proizvodno poduzeće srednje veličine.
Na osnovi prikazane ideje, potrebno je definirati osnovne zahtjeve za informacijskim sustavom koji bi podržao traženi način rada, te izraditi mali informacijski sustav s mogućnostima kreiranja proizvodnih naloga te upisa utrošenih materijala i njihovih količina, korištenih proizvodnih linija i utrošenog vremena.
Za sve podatke treba biti omogućeno pretraživanje i izvještavanje po izabranim kriterijima.

Rok za predaju rada: 20. rujna 2023.

Mentor:

dr. sc. Josip Nađ, mag.ing.el. pred.

Predsjednik povjerenstva za završni ispit:

SAŽETAK

Zbog potrebe za automatizacijom i poboljšanjem učinkovitosti proizvodnih i poslovnih procesa osmišljeni su mnogi informacijski sustavi koji omogućuju poduzećima da bolje raspoređuju svoje resurse i informacije, odnosno da imaju veću i bolju kontrolu nad svim faktorima koji su uključeni u proizvodne i poslovne procese. Završni rad usredotočen je na informacijski sustav koji je dizajniran posebno za potrebe praćenja proizvodnje gdje se kao osnovni dokument koristi proizvodni nalog. Ovaj informacijski sustav pruža centraliziranu platformu za stvaranje i upravljanje proizvodnim nalozima koji su glavna točka za praćenje i kontrolu zadataka koji se obavljaju unutar nekog poduzeća ili organizacije. Spremanjem i upravljanjem nad bitnim informacijama o radnicima, tvornicama, skladištima i materijalima, informacijski sustav nudi cjeloviti pogled na sve informacije vezane za proces proizvodnje u poduzeću.

Glavne tehnologije korištene za izradu ovog rada u razvoju ovog informacijskog sustava su Node.js, PostgreSQL, React i Bootstrap.

React i Bootstrap su zaslužni za klijentski dio aplikacije. React se koristi zbog svoje sposobnosti kreiranja ponovno iskoristivih komponenti te zbog toga što je kôd napisan u Reactu lakši za održavanje i fleksibilan je zbog svoje modularne strukture. Bootstrap se koristi za brže i lakše dizajniranje odnosno korištenje već kreiranih CSS (engl. *Cascading Style Sheets*) komponenti.

Što se tiče poslužiteljskog dijela aplikacije, koristi se Node.js s mnoštvom različitih npm (engl. *Node Package Manager*) paketa poput Express.jsa, Pga, Cookie-parsera te ostalih paketa zaduženih za rad poslužitelja. Korištenjem Node.jsa i već navedenih paketa vrlo lako se uspostavlja konekcija s PostgreSQL bazom podataka. Korištenjem JavaScripta na klijentskom dijelu aplikacije (React) i na poslužiteljskom dijelu aplikacije (Node.js) omogućuje se ponovno korištenje kôda između klijentskog dijela i poslužiteljskog dijela te se ubrzava razvoj aplikacije zato što je sve napisano u jednom programskom jeziku.

U radu su detaljnije opisane različite značajke i tehnologije koje su korištene u razvoju informacijskog sustava za upravljanjem proizvodnim nalozima, naglašavajući njihovu važnost u kontekstu kreiranja ovog informacijskog sustava.

Ključne riječi: Informacijski sustav, React, NodeJS, PostgreSQL, Bootstrap

Sadržaj

1. UVOD

U današnjem brzo mijenjajućem poslovnom svijetu, učinkovito upravljanje resursima neke tvrtke ili poduzeća ključno je za održavanje konkurentske prednosti. Informacijski sustavi za upravljanje i nadzorom proizvodnje, resursa i ostalih stvari vezanih za uspješno poslovanje poduzeća postali su moćni alati koji omogućuju unapređenje i bolju kontrolu nad različitim poslovnim procesima, pružajući centraliziranu platformu za učinkovito upravljanje resursima. Ovaj završni rad prikazuje informacijski sustav za upravljanje proizvodnim nalozima.

Informacijski sustav za upravljanje proizvodnim nalozima predstavlja sveobuhvatno rješenje koje korisnicima omogućuje stvaranje i upravljanje proizvodnih naloga koji igraju ključnu ulogu u praćenju i kontroliranju zadataka u nekom poduzeću ili organizaciji. Takav sustav pruža mnoštvo informacija o radnicima, tvornicama, skladištima, materijalima, i vremenskim okvirima u kojima se radni nalozi stvaraju i završavaju. Organiziranjem tih podataka informacijski sustav omogućuje poduzećima da steknu vrijedne uvide u svoje proizvodne i poslovne procese.

Ovaj informacijski sustav koristi kombinaciju popularnih web tehnologija kao što su Node.js, PostgreSQL, React i Bootstrap te je korištenjem takvih tehnologija dobiveno robusno i korisnički prijateljsko (engl. *User friendly*) rješenje.

Node.js, sa svojim skalabilnim i učinkovitim okruženjem izvršavanja, čini temelj serverskog (engl. *backend*) dijela aplikacije, osiguravajući optimalne performanse i obradu istovremenih zahtjeva.

PostgreSQL, pouzdan sustav za upravljanje relacijskim bazama podataka, služi za pohranu i dohvat podataka koji su ključni za upravljanje nad proizvodnim nalozima.

Na korisničkom dijelu (engl. *frontend)*, React, popularna JavaScript biblioteka, omogućuje stvaranje dinamičkih i ponovno upotrebljivih korisničkih sučelja (engl. *component*), poboljšavajući upotrebljivost i održavanje aplikacije.

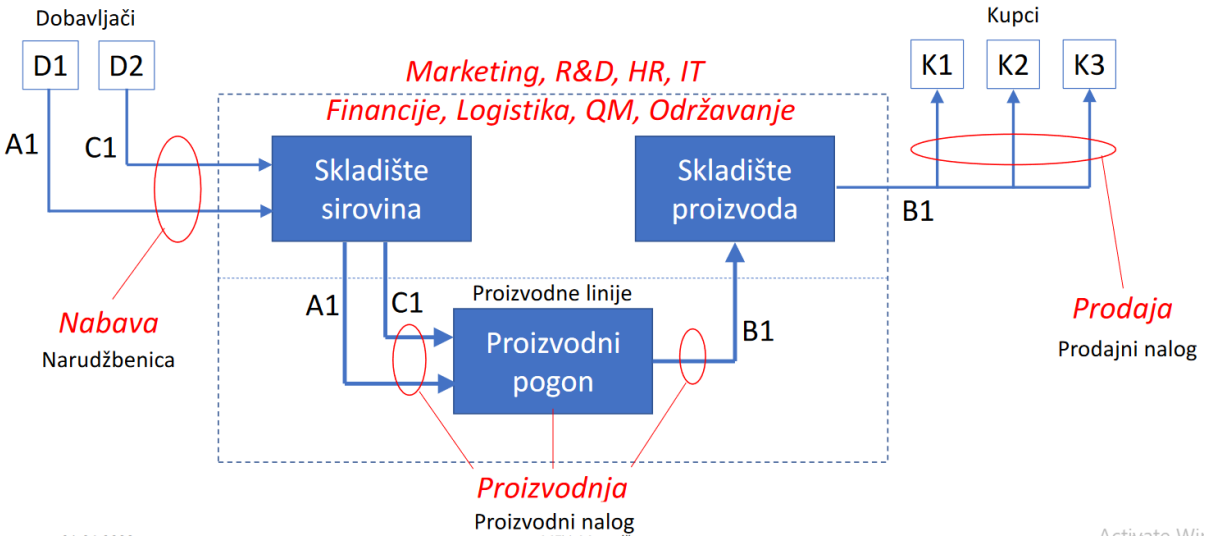
Korištenjem Bootstrapa, popularnog CSS okvira (engl. *framework*), osigurava se vizualno privlačno korisničko iskustvo i responzivnost na različitim uređajima.

2. OSNOVNI PROIZVODNI MODEL

Osnovni proizvodni model je prikaz ili shema poslovanja nekog poduzeća. On omogućuje da vidimo i da imamo kontrolu nad svim informacijama poslovnih procesa.

Postoje tri glavna procesa u osnovnom proizvodnom modelu, a to su:

- Nabava
- Proizvodnja
- Prodaja



Slika 1 Osnovni proizvodni model [1]

Informacije o kupnji, prodaji i proizvodnji igraju ključnu ulogu za uspješno poslovanje nekog poduzeća ili organizacije.

2.1. Kontrola zalihe

Prateći nabavu sirovina, proizvodni proces i prodaju gotovih proizvoda informacijski sustav omogućuje poduzećima optimizirane količine zaliha. To osigurava da je prava količina materijala ili proizvoda dostupna u pravo vrijeme, minimizirajući troškove prijenosa zaliha i izbjegavajući nestašice ili prekomjerne zalihe

2.2. Planiranje potražnje

Podaci o kupnji, prodaji i proizvodnji koje ERP aplikacija bilježi pružaju uvide u trendove potražnje. Analizom ovih informacija organizacije ili poduzeća mogu točnije predvidjeti buduću potražnju i u skladu s tim planirati svoje aktivnosti nabave i proizvodnje.

2.3. Kontrola troškova

Informacije o kupnji, prodaji i proizvodnji ključne su za kontrolu troškova unutar organizacije. Prateći nabavne cijene, prihode od prodaje i troškove proizvodnje, informacijski sustav omogućuje tvrtkama da prate i analiziraju svoje troškove u cijelom opskrbnom lancu. To omogućuje prepoznavanje područja neučinkovitosti, optimiziranje odluka o nabavi, pregovaranje o boljim cijenama s dobavljačima i pojednostavljenje proizvodnih procesa kako bi se smanjili troškovi i povećala profitabilnost.

2.4. Financijsko upravljanje

Podaci o kupnji, prodaji i proizvodnji temelj su za točno financijsko upravljanje nekog poduzeća ili organizacije. Ove podatkovne točke pridonose stvaranju financijskih izvješća, kao što su bilance, izvješća o dobiti i izvješća o novčanom tijeku. Integracijom financijskih podataka s drugim modulima u ERP sustavu, organizacije mogu steći holistički pregled svojih financija i donositi informirane odluke u vezi s proračunom, predviđanjem i financijskim planiranjem.

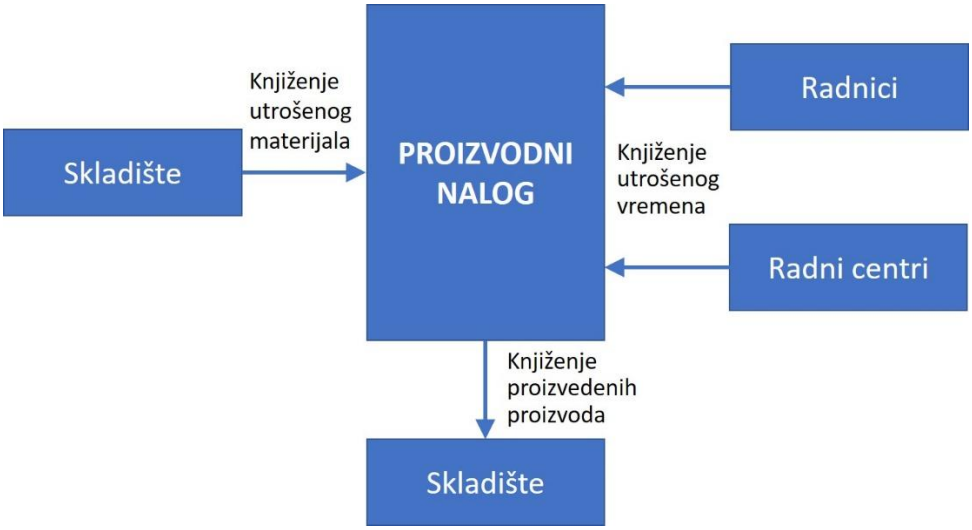
2.5. Optimizacija opskrbnog lanca

Informacije o kupnji, prodaji i proizvodnji omogućuju poduzećima ili organizacijama da optimiziraju svoje procese opskrbnog lanca. Analizom ovih podataka organizacije mogu pojednostaviti načine rada i poboljšati koordinaciju između različitih odjela. To dovodi do poboljšanja učinkovitosti opskrbnog lanca, smanjenja vremena isporuke, poboljšanja zadovoljstva kupaca i povećane ukupne produktivnosti.

3. PROIZVODNI NALOZI

3.1. Osnovna ideja proizvodnih naloga

Glavna ideja proizvodnih naloga je evidencija različitih aspekta proizvodnih procesa. Proizvodni nalog služi kao strukturirani sažetak svih aktivnosti povezanih s procesom proizvodnje kao što su utrošak vremena i radne snage, utrošak materijala i obrada proizvoda kao što je prikazano na Slici 2.



Slika 2 Proizvodni nalog i prateća knjiženja [1]

Svaki proizvodni nalog je vezan na radnika koji ga je izradio te se na taj način osigurava odgovornost osobe koja je izradila proizvodni nalog. Praćenjem utrošenog vremena i količine proizvedenih proizvoda dobivaju se bitne informacije za praćenje napretka proizvodnje i donošenja informiranih odluka. Informacijama dobivenim iz proizvodnih naloga omogućava se uvid u proizvodne aktivnosti, količine utrošenih materijala i resursa, vrijeme potrebno za izradu proizvoda i zadužene zaposlenike. Pristup takvim informacijama osigurava da su proizvodni procesi dobro organizirani, kontrolirani i usklađeni s operativnim ciljevima poduzeća.

3.2. Čimbenici koji utječu na razvoj IS-a proizvodnih naloga

Prilikom odabira alata i tehnologija za razvojem ili kreiranjem informacijskog sustava proizvodnih naloga potrebno je uzeti u obzir nekoliko ključnih čimbenika.

3.2.1. Funkcionalni zahtjevi

Za izradu informacijskog sustava proizvodnih naloga potrebno je definirati specifične funkcije i značajke kojima informacijski sustav treba upravljati. Treba provjeriti mogu li alati i tehnologije adekvatno ispunjavati željene zahtjeve.

3.2.2. Skalabilnost

Tipični informacijski sustavi obično obrađuju velike količine podataka i imaju velika opterećenja korisnika. Potrebno je osigurati da se alati i tehnologije mogu učinkovito skalirati kako bi se prilagodili budućem rastu informacijskog sustava, to jest većim brojevima zahtjeva.

3.2.3. Mogućnost integracije

Informacijski sustavi često se moraju integrirati s različitim sustavima i tehnologijama kao što su baze podataka i API-ji (engl. *Application Programming Interface)*. Potrebno je osigurati da alati i tehnologije koje su odabrane budu robusne te da imaju mogućnosti integracije i da mogu komunicirati i razmjenjivati podatke s drugim sustavima.

3.2.4. Prilagodba i fleksibilnost

Potrebno je procijeniti fleksibilnost alata i tehnologija za proširenje informacijskog sustava prema specifičnim poslovnim zahtjevima. Također potrebno je odrediti može li se postojeći informacijski sustav jednostavno prilagoditi različitim poslovnim procesima.

3.2.5. Dostupnost i podrška

Snažan sustav podrške i aktivna zajednica mogu pružiti vrijedne resurse, dokumentaciju i pomoć kada je potrebno poboljšati postojeći informacijski sustav. Potrebno je procijeniti dostupnost podrške te veličinu i aktivnost zajednice koja koristi takve alate i tehnologije.

3.2.6. Sigurnost

Sigurnost je ključna za svaki informacijski sustav budući da se radi o osjetljivim poslovnim podacima i procesima. Potrebno je osigurati da alati i tehnologije imaju snažne sigurnosne mjere i da su u skladu s relevantnim industrijskim propisima i standardima kao što su enkripcija podataka i kontrola pristupa.

Autor članka [2] govori kako se aplikacija povećava, povećava se i mogućnost grešaka te kako neki informacijski sustavi nude više od čak 800 softverskih prilagodbi koje mogu kontrolirati korisnici.

3.2.7. Cijena

Potrebno je razmotriti ukupne troškove implementacije i održavanja odabranih alata i tehnologija. Također potrebno je procijeniti naknade za licenciranje, troškove hostinga i potencijalne zahtjeve za obuku osoblja. Ako informacijski sustav koristi alate i tehnologije koje se plaćaju, potrebno je procijeniti potencijalni povrat ulaganja u smislu poboljšane učinkovitosti, produktivnosti i boljih poslovnih rezultata.

3.2.8. Održivost

Da informacijski sustav bude održiv i dugovječan potrebno je koristiti tehnologije koje imaju česta ažuriranja i srednji ili velik ekosustav. Također predlaže se izbjegavanje tehnologija koje su zastarjele ili onih koje nemaju dugoročnu podršku.

4. KORIŠTENI ALATI I TEHNOLOGIJE

Korištenjem Node.jsa, PostgreSQLa, Reacta i Bootstrapa, ova web aplikacija pruža sveobuhvatno i učinkovito rješenje za upravljanje radnim nalozima. Te tehnologije pružaju potrebne alate i funkcionalnosti za stvaranje skalabilne, sigurne i korisniku prilagođene aplikacije.

4.1. Web-baziran informacijski sustav

Informacijskom sustavu koji je temeljen na web tehnologijama može se pristupiti s bilo kojeg mjesta s internetskom vezom, što korisnicima omogućuje rad na daljinu ili pristup informacijama dok su u pokretu. Ova pristupačnost poboljšava produktivnost i suradnju jer korisnici mogu pristupiti aplikaciji s različitih uređaja, uključujući prijenosna računala, tablete i pametne telefone.

Web aplikacije su također neovisne o platformi, što znači da im se može pristupiti s različitih operativnih sustava kao što su Windows, macOS ili Linux bez potrebe za posebnim instalacijama ili konfiguracijama. Ova široka kompatibilnost pojednostavljuje implementaciju i smanjuje probleme s kompatibilnošću.

Korištenjem web tehnologija olakšava se pohrana i upravljanje podacima. Svi podaci pohranjuju se na poslužitelju, čime se korisnicima osigurava pristup najnovijim informacijama u stvarnom vremenu. Web aplikacije obično je lakše skalirati u usporedbi s desktop aplikacijama.

Informacijski sustavi temeljeni na webu često zahtijevaju manje ulaganja u komponente i infrastrukturu u usporedbi s desktop aplikacijama. Korisnici mogu pristupiti aplikaciji putem standardnih web preglednika. Centralizirano održavanje i ažuriranja smanjuju troškove i poboljšavaju isplativost.

Općenito, web aplikacije imaju prednost u smislu pristupačnosti, kompatibilnosti s više platformi, jednostavnog ažuriranja i održavanja, centraliziranog upravljanja podacima, skalabilnosti, isplativosti i mogućnosti integracije. Ove prednosti čine web-bazirane informacijske sustave popularnim izborom za poduzeća koje traže fleksibilnost, učinkovitost i održivost.

4.2. Frontend

Prema članku [3] autor govori da se razvoj frontenda odnosi se na proces stvaranja dijela web aplikacije koji je vidljiv korisniku. Uključuje dizajniranje i implementaciju vizualnih i interaktivnih elemenata koje korisnici vide i s kojima komuniciraju izravno u svojim web preglednicima. Razvoj frontenda usredotočen je na izradu korisničkog sučelja (UI) i osiguravanja dobrog korisničkog iskustva (UX).

4.2.1. React

React je popularna JavaScript biblioteka koja se koristi za izradu korisničkih sučelja. Razvijena od strane Facebooka gdje se koristi za stvaranje interaktivnih i dinamičkih web aplikacija. React koristi pristup temeljen na komponentama, gdje je korisničko sučelje podijeljeno na više upotrebljivih i neovisnih komponenti. [4]

React koristi virtualni DOM (engl. *Document Object Model*) za učinkovito ažuriranje i renderiranje korisničkog sučelja. Virtualni DOM je prikaz stvarnog DOM-a u memoriji. Kada dođe do promjena u stanju aplikacije, React uspoređuje virtualni DOM sa stvarnim DOM-om i ažurira samo potrebne dijelove, minimizirajući broj stvarnih DOM manipulacija. Ovaj pristup poboljšava performanse i pruža dobro korisničko iskustvo.

React upravlja stanjem aplikacije koristeći stanje komponenti (engl. *state)* i svojstva (engl. *props*). Kôd 1. prikazuje varijablu koja se koristi pri registraciji nazvanu „user“ gdje kao početno stanje koristi objekt sa svojstvima imena, prezimena, e-maila i lozinke.

```
const [user, setUser] = useState({ first_name: null,
    last_name: null,
    email: null,
    password: null,
  });
```

Kôd 1 React stanje (engl. *state*)

Izvor: Autor

Stanje predstavlja promjenjive podatke koji se mogu mijenjati tijekom vremena unutar komponente. Svojstva su nepromjenjivi podaci koji se prenose od glavne komponente do njenih podređenih komponenti.

React ima golem ekosustav sa širokim rasponom biblioteka i alata. Postoje različite biblioteke i resursi koji proširuju mogućnosti Reacta čime pružaju rješenja za različite slučajeve upotrebe kao što su upravljanje stanjem (Redux, useContext), biblioteke komponenti korisničkog sučelja (Material-UI, Ant Design) i više.

React pojednostavljuje razvoj interaktivnih i dinamičkih korisničkih sučelja, promiče ponovnu upotrebu kôda i nudi robustan ekosustav alata i biblioteka za povećanje produktivnosti i stvaranja modernih web aplikacija.

4.2.2. CSS, Bootstrap i Reactstrap

CSS i Bootstrap su tehnologije koje igraju važnu ulogu u dizajniranju i oblikovanju web stranica i aplikacija.

Bootstrap je popularan CSS (engl. *framework*) otvorenog kôda (engl. *open-source*) koji pojednostavljuje proces izrade responzivnih i vizualno privlačnih web stranica i web aplikacija. Pruža kolekciju unaprijed stiliziranih CSS klasa i JavaScript komponenti koje programeri mogu iskoristiti za stvaranje modernog korisničkog sučelja.

Reactstrap je npm paket koji omogućuje korištenje Bootstrapa poput React komponenti kao što je prikazano na Kôdu 2. Korištenjem Reactstrapa

```
<Button outline color="success"
    style={{ width: "100%" }}
    onClick={() => handleSubmit()}
    disabled={!validEmail || !validPassword}
  >
    Registriraj se
  </Button>
```

Kôd 2 Gumb za registraciju korisnika

Izvor: Autor

Bootstrap je najpopularniji HTML, CSS i JavaScript (engl. *framework*) za razvoj responzivnih i mobilnih web stranica. [5]

4.3. Backend

Backend razvoj odnosi se na proces stvaranja i održavanja komponenti web aplikacije ili softvera na strani poslužitelja. Uključuje upravljanje logikom, operacijama baze podataka i obradom informacija na strani poslužitelja koja su potrebna za podršku i pružanje funkcionalnosti korisnicima.

4.3.1. Node.js

Node.js je open-source server-side JavaScript okruženje na strani poslužitelja koje programerima omogućuje izradu skalabilnih mrežnih aplikacija visokih performansi. Koristi I/O model vođen događajima, što ga čini prikladnim za aplikacije u stvarnom vremenu. [7]

Jedna od ključnih prednosti Node.js je njegov asinkroni ne-blokirajući I/O model. To znači da Node.js može obraditi više istodobnih zahtjeva bez blokiranja izvršavanja drugih zadataka. Kao rezultat toga, može učinkovito rukovati velikom količinom istodobnih veza i isporučiti brzo vrijeme odaziva funkcija.

Node Package Manager je repozitorij otvorenog kôda (engl. *open-source*) alata koji se koristi za razvoj web stranica i aplikacija. Node package manager pruža širok raspon modula i biblioteka za višekratnu upotrebu, omogućujući programerima da jednostavno integriraju funkcionalnost u svoje aplikacije i tako ubrzaju razvoj.

Korištenjem Node.jsa i npm paketa Express moguće je kreirati server i (engl. *application programming interface* - API) čime možemo vrlo lako upravljati HTTP (engl. *HyperText Transfer Protocol*) zahtjevima. Kôd 3. prikazuje način kreiranja i pokretanja servera gdje se kreira Express aplikacija, definira broj priključka (engl. *port-number*) i aktivira (engl. *middleware*).

```
const express = require("express");
const app = express();
const port = 3001;
const cors = require("cors");
const cookieParser = require("cookie-parser");
app.listen(port, ()=>{
    console.log("APP RUNNING ON PORT: " + port);
})
```

Kôd 3 Kreiranje i pokretanje servera

Izvor: Autor

Budući da aplikacija koristi PostgreSQL kao bazu podataka za spajanje na takvu bazu podataka potrebno je koristiti npm paket „pg“ (PostgreSQL). Kôd 4. prikazuje stvaranje konekcije između servera i baze podataka čime je potrebno definirati svojstva poput korisnika, naziva baze podataka, lozinke i broja priključka.

```
const db = new Pool({
    user: "postgres",
    host: "localhost",
    database: "erpis",
    password: "123456",
    port: "5432"
});
```

Kôd 4 Povezivanje na bazu podataka

Izvor: Autor

4.3.2. PostgreSQL

PostgreSQL moćan je sustav za upravljanje relacijskim bazama podataka otvorenog kôda poznat po svojoj robusnosti i pouzdanosti. Pruža skalabilno rješenje visokih performansi za upravljanje strukturiranim podacima, što ga čini popularnim izborom za širok raspon aplikacija. [6]

PostgreSQL slijedi model relacijske baze podataka, omogućavajući korisnicima definiranje tablica, uspostavljanje odnosa između njih i izvođenje složenih upita pomoću SQL-a (Structured Query Language). Podržava različite vrste podataka, uključujući numeričke, tekstualne, datum/vrijeme, JSON i još mnogo toga.

Nudi širok raspon snažnih mogućnosti postavljanja upita (engl. *query*). Podržava složene SQL upite, uključujući podupite, spojeve i agregacije.

Pruža sigurnosne značajke za zaštitu podataka i osiguravanje privatnosti. Podržava autentifikaciju korisnika i kontrolu pristupa temeljenu na ulogama. Uz to, PostgreSQL nudi opcije šifriranja za podatke u prijenosu i mirovanju.

5. APLIKACIJA

U ovome poglavlju objasnit ću kako pripremiti radno okruženje za pokretanje aplikacije, način pokretanja aplikacije i funkcionalnosti same aplikacije.

5.1. Priprema radnog okruženja

Kako bi pokrenuli aplikaciju potrebno je na početku instalirati alate i tehnologije koje su korištene za izradu ovog informacijskog sustava.

Za pokretanje aplikacije potrebno je instalirati Node.js i npm (*engl. Node Package Manager*). Node Package Manager je moćan alat koji dolazi s Node.jsom i pruža nekoliko pogodnosti za programere kao što su:

- Upravljanje različitim bibliotekama (*engl. dependency*) u projektima
- Dijeljenje vlastitog kôda kao paketa, čime taj isti kôd možemo upotrebiti u više različitih projekata
- Alati za provjeru zastarjelih paketa i sigurnosnih ranjivosti u projektu, taj npm omogućava da ostanemo u tijeku s najnovijim verzijama i da osigurava sigurnost našeg projekta

Node.js je moguće preuzeti sa službene stranice Node.jsa: https://nodejs.org/en. Nakon instalacije Node.jsa preporuča se da kroz terminal ili command promt izvršimo naredbu „node -v“. Trebao bi se prikazati važeći broj verzije, što pokazuje da je Node.js sada dostupan za upotrebu.

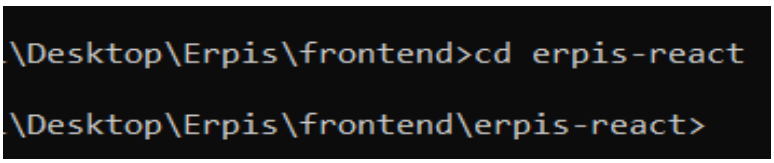
Nakon instalacije NodeJSa i npm-a potrebno je još samo instalirati PostgreSQL. PostgreSQL je moguće preuzeti sa službene stranice PostgreSQL-a: https://www.postgresql.org/.

5.2. Pokretanje aplikacije

Budući da ovaj informacijski sustav koristi programski jezik Javascript na backendu i frontendu, pojam često zvan kao „full-stack Javascript development“ vrlo je lakša komunikacija između klijentskog i korisničkog dijela aplikacije.

5.2.1. Klijentski dio aplikacije

Koristeći terminal ili command prompt potrebno se pozicionirati u direktorij react-erpis. To se može postići pomoću naredbe cd, nakon koje je potrebno napisati put do direktorija. Na primjer, ako se datoteka nalazi na radnoj površini, naredba bi bila „cd Desktop“.



Slika 3 Pozicioniranje u direktorij erpis-react

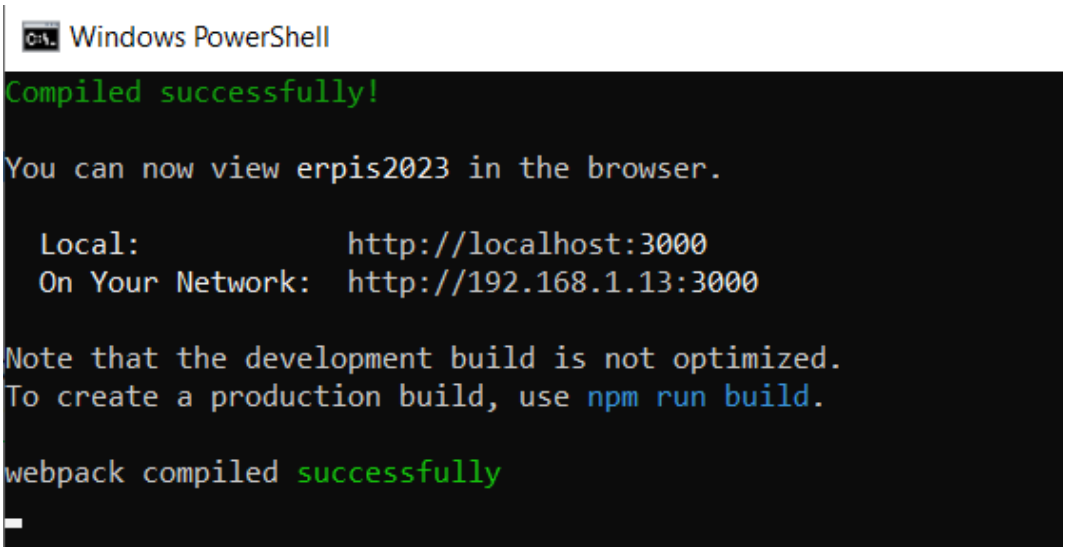
Izvor: Autor

Nakon pozicioniranja u točan direktorij potrebno je instalirati sve pakete i biblioteke koje su korištene u frontend dijelu aplikacije. To se može vrlo jednostavno učiniti pomoću node package managera. Kako bi instalirali npm pakete potrebno je u command promptu ili terminalu izvršiti naredbu „npm“ i ili „npm install“.

Slika 4 Node Package Manager komanda koja instalira pakate

Izvor: Autor

Ukoliko su se instalirali svi paketi potrebno je upotrijebiti naredbu „npm start“ za pokretanje aplikacije. Naredba „npm start“ izvršit će naredbu navedenu u skripti „start“ u ovom slučaju skriptu: „react-scripts start“. „react-scripts“ je skup skripti iz paketa „create-react-app“ koji postavlja razvojno okruženje i pokreće poslužitelj.



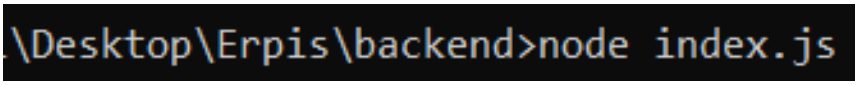
Slika 5 Izvršenje naredbe „npm start“

Izvor: Autor

Nakon izvršenja naredbe „npm start“ command prompt će ispisati informaciju da su se skripte uspješno kompajlirale te će se otvoriti prozor u vašem web pregledniku s početnom stranicom aplikacije.

5.2.2. Poslužiteljski dio aplikacije

Istim načinom kao i kod klijentskog dijela koristeći terminal ili command prompt potrebno se je pozicionirati u direktorij backend. To se postiže pomoću naredbe cd, nakon koje je potrebno napisati put do direktorija. Nakon pozicioniranja u točan direktorij potrebno je izvršiti naredbu „node index.js“. Naredba „node index.js“ koristi se za izvršavanje JavaScript datoteke pod nazivom „index.js“ pomoću Node.js runtime okruženja.



Slika 6 Izvršavanje naredbe „node index.js“

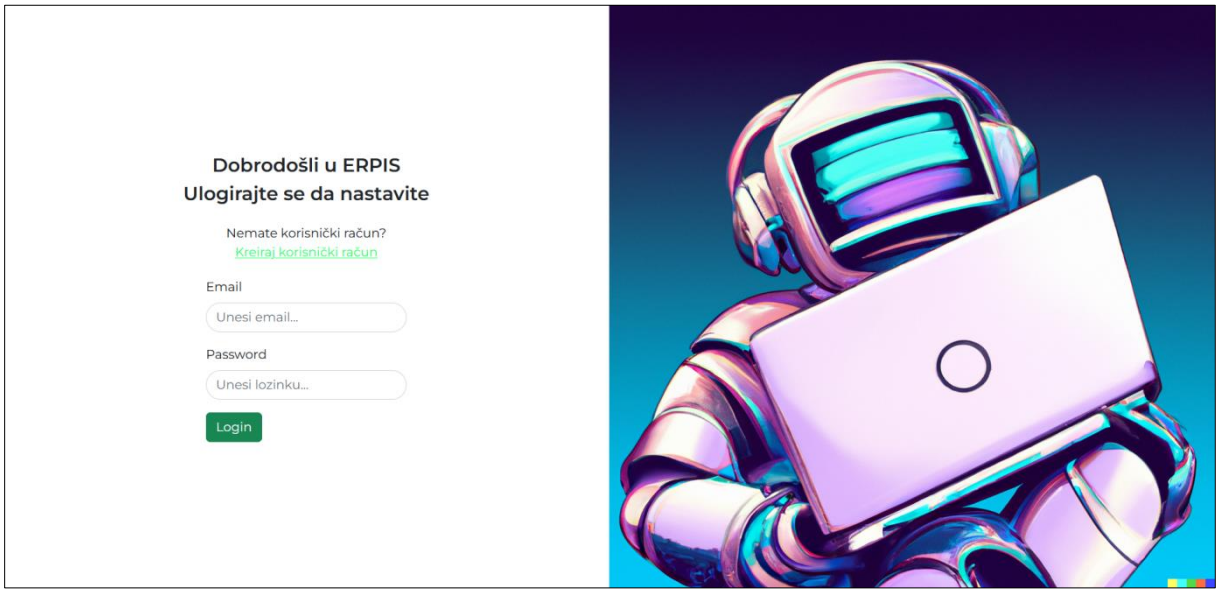
Izvor: Autor

Nakon što se naredba node „index.js“ izvršila, u command promptu će se ispisati poruka da je aplikacija pokrenuta na priključku (engl. *port*) 3001 što znači da je server uspješno pokrenut.

5.3. Funkcionalnosti aplikacija

5.3.1. Registracija i prijava

Pokretanjem aplikacije otvara se početna stranica na kojoj se korisnik treba prijaviti ukoliko ima korisnički račun te nakon uspješne prijave korisnik ima pristup svim funkcionalnostima aplikacije.



Slika 7 Početna stranica aplikacije

Izvor: Autor

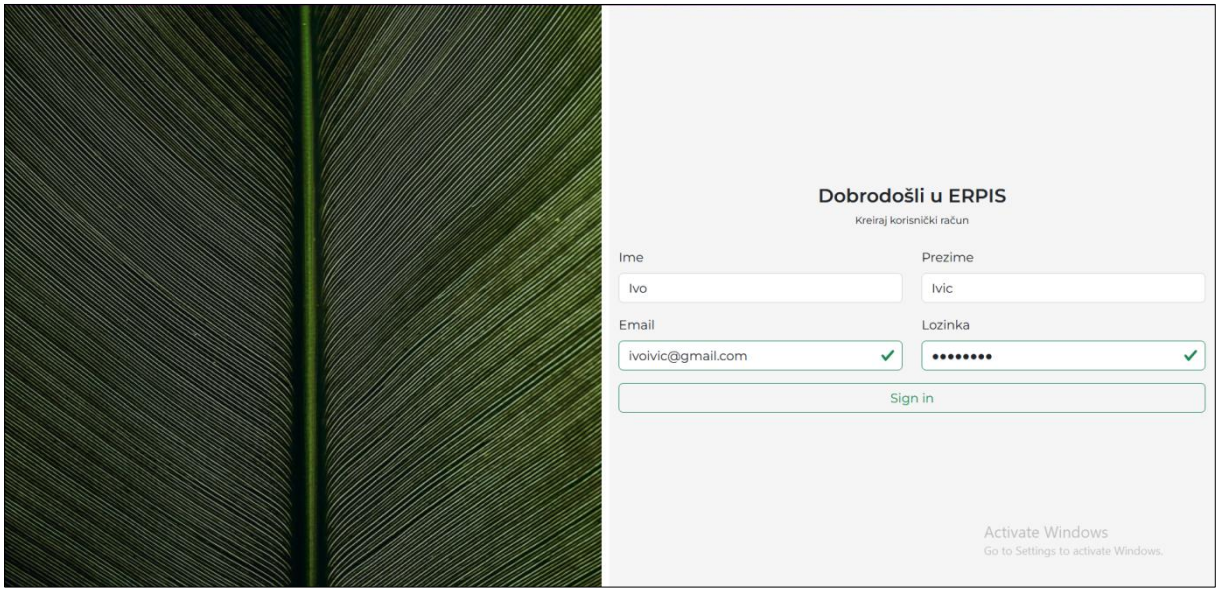
Da bi se korisnik uspješno prijavio potrebno je u polje email upisati svoju email adresu koja je dogovorena s adminom prilikom registracije te isto tako u polje password upisati lozinku koja mu je dodijelio admin. Slika 8 prikazuje formu za upis emaila i lozinke. Nakon upisanih podataka potrebno je pritisnuti gumb „Login“ koji šalje podatke o korisniku prema poslužitelju. Podaci se šalju pomoću HTTP (engl. *HyperText Transfer Protocol*) POST zahtjeva, čime, ukoliko su podaci točni, kao odgovor od poslužitelja dobivamo JavaScript web token koji nas time preusmjerava na stranicu za izradu proizvodnih naloga.

```
<Form>
  <FormGroup>
    <Label>Email</Label>
    <Input
      placeholder="Unesi email..." type="email" style={{borderRadius: "50px"}} value={email}
      onChange={(e)=>setEmail(e.target.value)}
    />
  </FormGroup>
  <FormGroup>
    <Label>Password</Label>
    <Input
      placeholder="Unesi lozinku..." type="password" style={{borderRadius: "50px"}} value={password}
      onChange={(e)=>setPassword(e.target.value)}
    />
  </FormGroup>
  <Button color="success" onClick={()=> handleLogin()}>Login</Button>
</Form>
```

Slika 8 Forma za upis podataka

Izvor: Autor

Pristup informacijskom sustavu korisnicima daje isključivo admin. Klikom na gumb „Novi Korisnik“ koji se nalazi na navigacijskoj traci koja je vidljiva isključivo adminu preusmjerava admina na stranicu gdje može registrirati novog korisnika. Registracija se vrši tako da admin upiše ime i prezime, email i lozinku novog korisnika kao što je prikazano na Slici 9.



Slika 9 Registracija korisnika

Izvor: Autor

Aplikacija koristi (engl. regular expressions „regex“) čime provjerava jesu li email i lozinka pravilno upisani to jest zadovoljavaju li posebne kriterije formatiranja navedene u regex funkciji. Za validaciju lozinke koristimo regex funkciju prikazanu na Kôdu 5. koja zahtijeva sljedeće kriterije:

- Lozinka mora sadržati između 6 i 16 slova
- Lozinka mora imati barem jedan broj
- Lozinka ne smije sadržati niti jedan razmak između slova (*engl. whitespace characters*)

```
const isValid = new String(txt)
      .toLowerCase()
      .match(
        /^(?=.*\d)(?!.*\s).{6,16}$/
      );
```

Kôd 5 Regex za validaciju lozinke

Izvor: Autor

Što se tiče validacije emaila koristi se regex funkcija prikazana na Kôdu 6. koja zahtijeva sljedeće kriterije:

- Lokalni dio (ispred simbola "@") može sadržavati slova, znamenke, točke, povlake, znakove plus i crtice
- Naziv domene (iza simbola "@" i prije (engl. *top level domain*)) može sadržavati slova, znamenke, točke i crtice.

```
const isValid = new String(email)
      .toLowerCase()
      .match(
        /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/
      );
```

Kôd 6 Regex za validaciju email-a

Izvor: Autor

Nakon upisa svih podataka i ukoliko su podaci validirani pritiskom na gumb „Sign in“ šalju se podaci prema poslužitelju. Podaci se šalju pomoću HTTP POST zahtjeva gdje poslužitelj provjerava postoji li u bazi podataka korisnik s istim e-mailom kao što prikazano na Kôdu 7.

```
const data = await db.query(`SELECT * FROM "users" WHERE "email"= $1;`, [email]);
        const arr = data.rows;
        if(arr.length > 0) {
            return res.status(400).json({
                err: "Email s istim nazivom već postoji"
            })
        }
```

Kôd 7 Provjera emaila

Izvor: Autor

Ukoliko u bazi podataka nije pronađen korisnik s istim e-mailom nastavlja se s procesom registracije korisnika tako što se lozinka hešira (engl. *hashing*). Heširanje je proces pretvaranja podataka - teksta, brojeva, datoteka - u niz slova i brojeva fiksne duljine. Podaci se pretvaraju u nizove fiksne duljine ili hash vrijednosti pomoću posebnog algoritma koji se zove hash funkcija. [8]

Korištenjem bcrypt biblioteke moguće je heširati lozinku na različite načine ovisno o potrebama aplikacije, u ovom slučaju koristi se bcrypt funkcija koja koristi (engl. *Blowfish encryption algorithm).*

```
const hashPassword = async(password) => {
    return new Promise((resolve, reject)=>{
        bcrypt.hash(password, 10, (err, hash)=> { //bcrypt funkcija koja kao parametre uzima lozinku i broj soli
            if(err) {
                reject(res.status(400).json({
                    erorr: "Nešto je pošlo po krivu" //Ukoliko se dogodi greška zaustavlja se funkcija i vraća error
                }));
            }
            resolve(hash); //Lozinka uspješno heširana
        });
    })
}
```

Slika 10 Heširanje lozinke pomoću bcrypt biblioteke

Izvor: Autor

Nakon što je lozinka uspješno heširana novog korisnika se sprema u bazu podataka. Novog korisnika se sprema u bazu podataka pomoću SQL upita prikazanog na Kôdu 8.

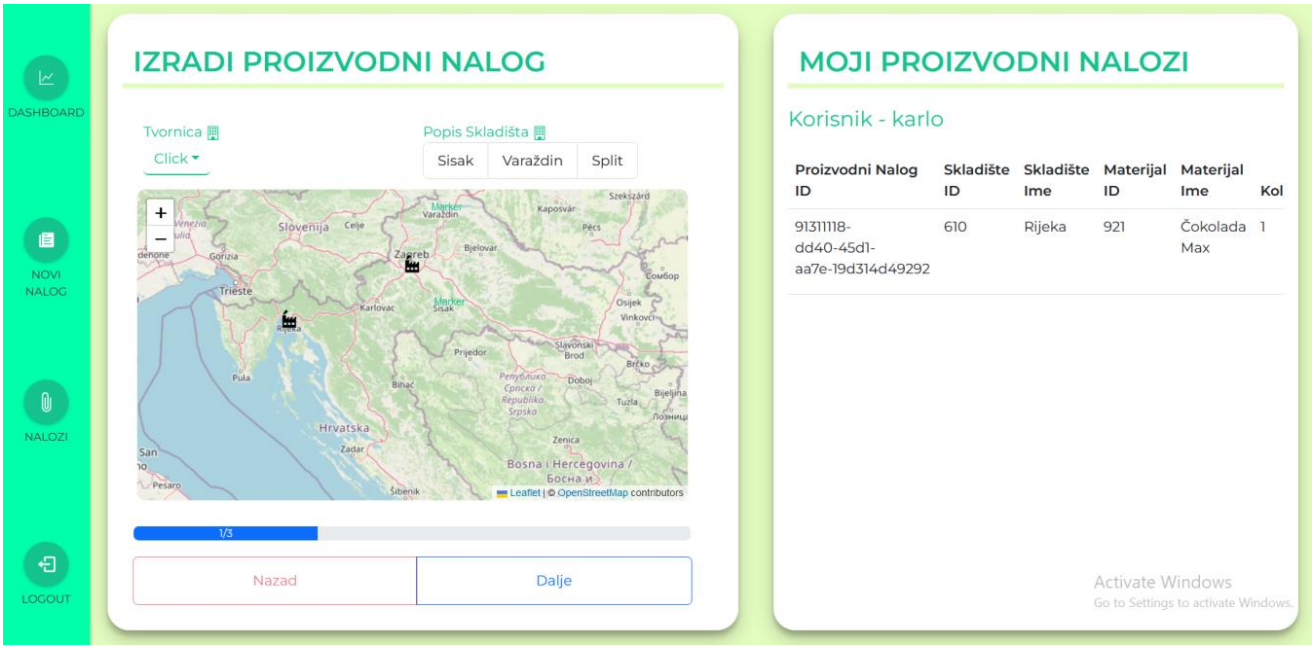
```
INSERT INTO users (first_name, last_name, email, password) VALUES ($1, $2, $3, $4);
```

Kôd 8 SQL upit za registraciju korisnika

Izvor: Autor

5.3.2. Izrada proizvodnih naloga

Kao što je već spomenuto u poglavlju 5.3.1. Registracija i prijava, uspješnom prijavom otvara se stranica za izradu proizvodnih naloga kao što je prikazano na Slici 11.

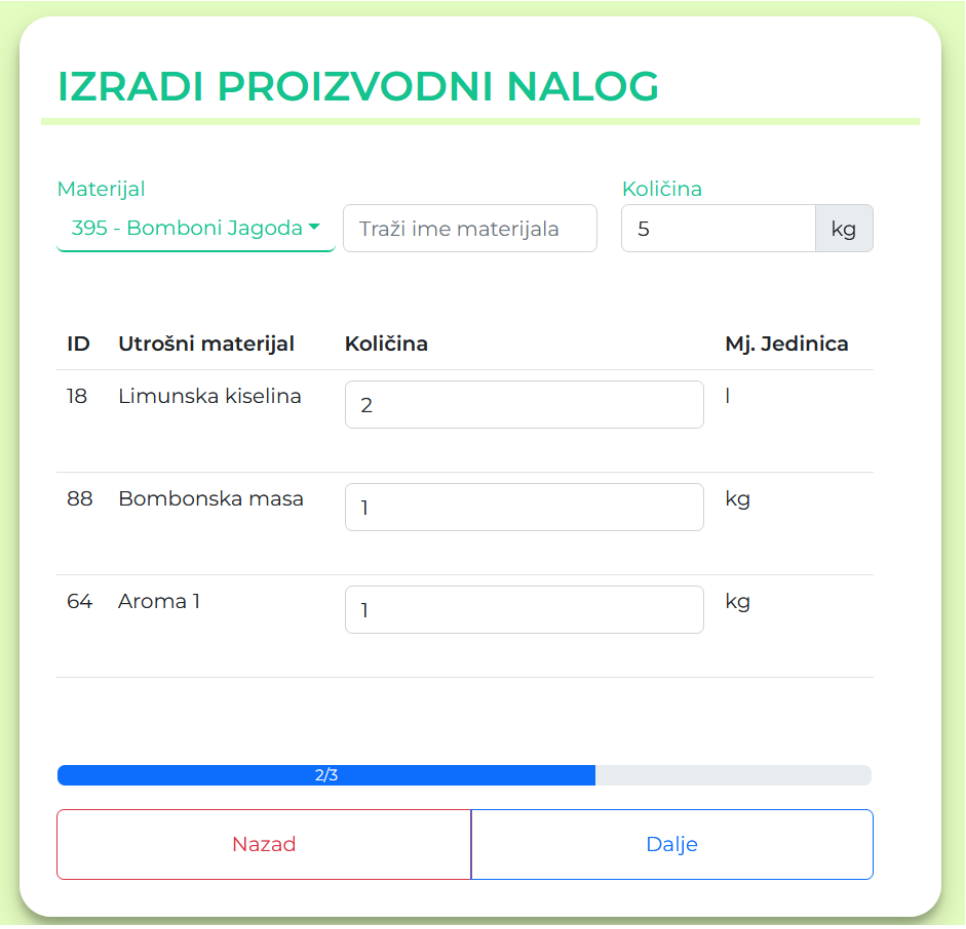


Slika 11 Stranica za izradu proizvodnih naloga

Izvor: Autor

Za izradu proizvodnog naloga potrebno je zadovoljiti sljedeće kriterije:

- Odabrati tvornicu u kojoj se vrši izrada proizvoda, na primjer Zagreb
- Odabrati proizvod koji će se proizvesti, na primjer Bomboni Jagoda
- Upisati količinu proizvoda koji će se proizvesti
- Upisati količinu materijala koji se koriste za izradu proizvoda



Slika 12 Upis proizvoda i materijala

Izvor: Autor

Nakon što su upisani svi podaci klikom na gumb „Dalje“ otvara se prozor u kojem su vidljive sve informacije vezane za izradu proizvodnog naloga kao što je prikazano na Slici 13.

Slika 13 Informacije o izradi proizvodnog naloga

Izvor: Autor

Pritiskom na gumb „Izvrši proizvodni nalog“ podaci o proizvodnom nalog se šalju prema poslužitelju. Poslužitelj provjerava informacije o količini materijala koji se koriste za izradu proizvoda te ukoliko je količina potrebnog materijala navedena u proizvodnom nalogu veća od količine materijala koji je raspoloživ u skladištu, proizvodni nalog se ne izvršava. Slika 14. prikazuje funkciju koja dohvaća materijale iz skladišta koji imaju količinu veću ili jednaku broju upisanom na proizvodnom nalogu.

```
const checkItemsInStash = async (mat, quantity) => {
  try {
    const { rows } = await db.query(
      "SELECT num_of_items FROM quantity WHERE material_id = $1 AND num_of_items >= $2;",
      [mat, quantity]
    );

    if (rows.length <= 0) return false;

    return parseInt(rows[0].num_of_items);
  } catch (error) {
    if (error) {
      throw error;
    }
  }
};
```

Slika 14 Funkcija za provjeru količine materijala

Izvor: Autor

Ukoliko se u skladištu nalazi dovoljan broj materijala za izradu proizvoda, proizvodnom nalogu se dodaje (engl. *Unique User IDentifier*) koristeći funkciju randomUUID iz crypto paketa.

Funkcija randomUUID koristi se za generiranje UUID-a pomoću kriptografski sigurnog generatora slučajnih brojeva [9].

Nakon dodjeljivanja UUID-a, proizvodni nalog se sprema u bazu podataka s napomenom da je status proizvodnog naloga u tijeku. Proizvodni nalog sa statusom „U tijeku“ označava da je proizvodni nalog kreiran ali ne i završen.

5.3.3. Pregled i uređivanje proizvodnih naloga

Koristeći navigacijsku traku klikom na gumb „Nalozi“ otvara se stranica na kojoj su vidljivi svi proizvodni nalozi koje je korisnik kreirao. Proizvodni nalozi sadrže sljedeće informacije:

- ID Proizvodnog naloga
- Ime i ID skladišta
- Ime i ID proizvoda
- Količinu i mjernu jedinicu proizvoda
- Datum i vrijeme izrade i završetka proizvodnog naloga
- Status proizvodnog naloga

Svaki proizvodni nalog također sadrži dva gumba koji omogućuju uređivanje ili završavanje (promjena statusa) proizvodnog naloga. Slika 15. prikazuje tablicu u kojoj se nalaze proizvodni nalozi gdje je status prvog proizvodnog naloga završen dok je status drugog proizvodnog naloga u tijeku.

Slika 15 Prikaz proizvodnih naloga

Izvor: Autor

Klikom na ikonicu kvačice proizvodni nalog se završava dok klikom na ikonicu olovke otvara se prozor koji omogućuje uređivanje proizvodnog naloga – promjenu količine proizvoda i materijala. Završavanjem proizvodnog naloga poslužitelj ažurira količinu proizvoda i materijala u bazi podataka te mijenja status proizvodnog naloga iz „U tijeku“ u „Završen“.

Kôd 9. prikazuje funkciju koja kreira SQL upit koji mijenja status proizvodnog naloga koristeći ID proizvodnog naloga kao parametar.

```
const data = await db.query("UPDATE warrants SET status = true,
date_end_of_warrant = CURRENT_DATE, time_end_of_warrant = cast(now()
AS TIME) WHERE id = $1;", [id]);
```

Kôd 9 Promjena statusa proizvodnog naloga

Izvor: Autor

Ukoliko se izvrše promjene na proizvodnom nalogu šalje se PUT zahtjev s informacijama o proizvodnom nalogu u (engl. *request body-u*) kao što je prikazano na Kôdu 10.

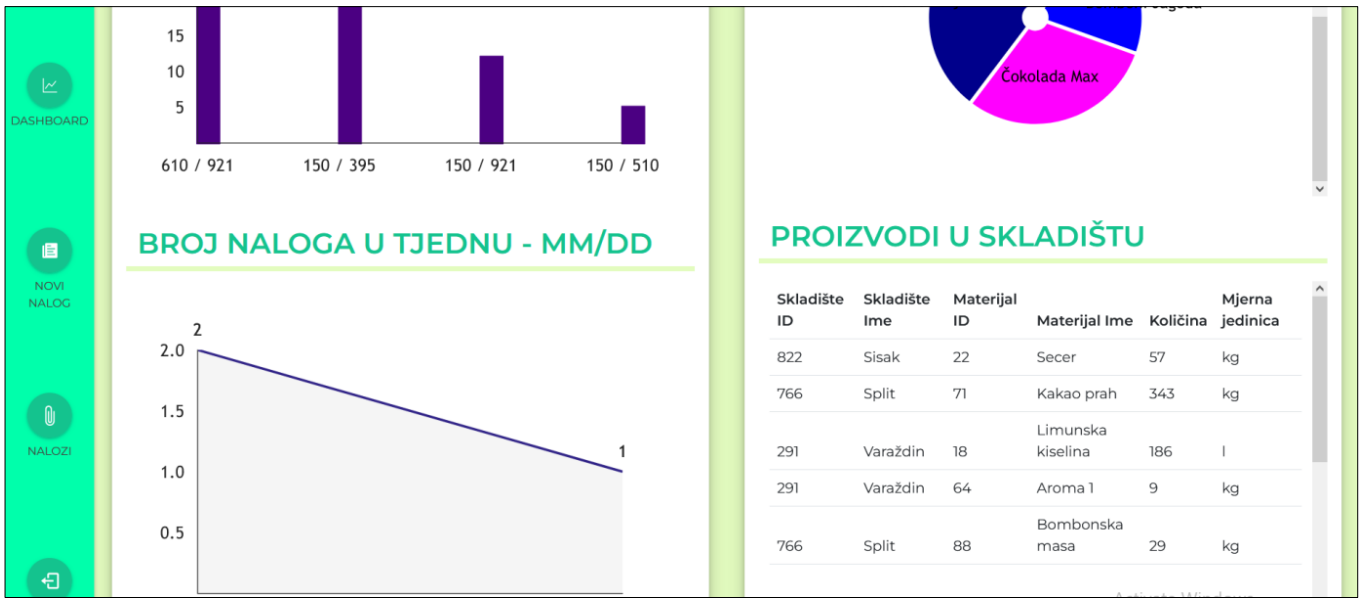
```
const handleUpdateWarrant = async () => {
    const warrant = { id: materials[findMaterialById(warrantId)].id,
      quantity: materials[findMaterialById(warrantId)].quantity,
      submaterials};
    try {
      const { data } = await axios.put(
        process.env.REACT_APP_WARRANTS_UPDATE_WARRANT,{ warrant }
      );
    } catch (error) {
      if (error) throw error;
    }
  };
```

Kôd 10 Promjena količine proizvoda i materijala

Izvor: Autor

5.3.4. Nadzorna ploča

Nadzorna ploča (engl. *Dashboard)* je dio aplikacije koji pruža uvid u informacije vezane za proizvodne naloge, proizvode i materijale kao što je prikazano na Slici 16.



Slika 16 Dashboard

Izvor: Autor

Za prikaz statističkih podataka korišten je npm paket „Victory“. Victory je biblioteka koja se koristi za jednostavno crtanje grafikona i vizualizaciju podataka. Informacije je moguće prikazati mnoštvom različitih Victory grafikona, neki od najkorištenijih su: stupičasti grafikon, tortni i prstenasti grafikoni, površinski grafikoni, linijski grafikoni...

Da bi izradili grafikon pomoću Victory-a potrebno je odabrati željeni grafikon te u svojstvo „data“ staviti željene podatke. Osim što je moguće kreirati grafikone također je i moguće uređivati svojstva kao što su: boja grafikona, visina, radius, tekst ...

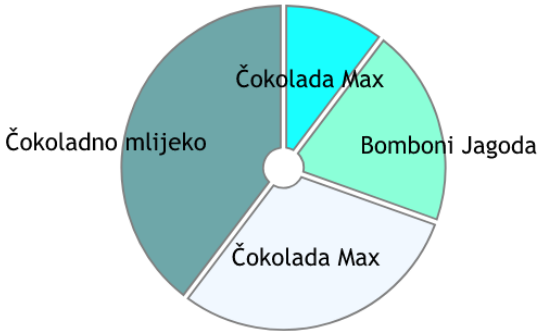
Kôd 11. prikazuje React komponentu koja služi za izradu tortnog grafikona gdje se u svojstvo „data“ postavljaju podaci vezani o količini proizvoda dohvaćeni iz baze podataka dok su ostala svojstva vezana za izgled grafikona.

```
<VictoryPie
        data={graphData}
        innerRadius={10}
        labelRadius={({ innerRadius }) => innerRadius + 30}
        radius={80}
        colorScale={[ "indigo","blue", "magenta",}
    />
```

Kôd 11 Victory tortni grafikon

Izvor: Autor

Slika 17. prikazuje tortni graf gdje se kao prikaz koristi količina proizvoda u svim skladištima.



Slika 17 Tortni graf – proizvodi

Izvor: Autor

Za prikaz količine proizvoda i materijala osim korištenja grafikona također je korištena tablica koja sadrži sve informacije o proizvodima i materijalima kao što je prikazano na Slici 17.

MATERIJALI U SKLADIŠTU

Skladište ID	Skladište Ime	Materijal ID	Materijal Ime	Količina	Mjerna jedinica
822	Sisak	22	Secer	57	kg
766	Split	71	Kakao prah	343	kg
291	Varaždin	18	Limunska kiselina	186	l
291	Varaždin	64	Aroma 1	9	kg
766	Split	88	Bombonska masa	29	kg

Slika 18 Materijali u skladištu

Izvor: Autor

Za dohvaćanje podataka vezanih o količini materijala u skladištu potrebno je na tablici „quantity“ spojiti (engl. *join*) tablicu „sub_materials“ i „warehouses“ kao što je prikazano na Kôdu 12.

```
SELECT DISTINCT ON (quantity.id) whs.id AS whs_id, whs.name AS whs_name, submat.id AS mat_id, submat.name AS mat_name, num_of_items, submat.measure FROM quantity INNER JOIN sub_materials AS submat ON submat.id = material_id INNER JOIN warehouses AS whs ON whs.id = warehouse_id;
```

Kôd 12 SQL upit za količinu materijala u skladištu

Izvor: Autor

6. ZAKLJUČAK

Informacijski sustav u kojem je moguće pratiti i imati pod kontrolom različite poslovne procese može biti izuzetno bitan i važan poduzećima. Dobar informacijski sustav dovodi do poboljšanja produktivnosti, smanjenih pogrešaka i bolje koordinacije među odjelima. Nudi objedinjeni prikaz poslovnih aktivnost i procesa, financijskih transakcija i još mnogo toga. Takav pristup podacima omogućuje informirano donošenje odluka, jer tako menadžeri mogu analizirati trendove podataka kako bi donijeli strateške odluke koje pokreću rast i profitabilnost tvrtke. Da bi informacijski sustav bio dobar i koristan potrebno je pažljivo razmotriti alate i tehnologije prilikom izrade istog.

Aplikacija ili informacijski sustav prikazan u završnom radu podijeljena je na klijentski i poslužiteljski dio. Za klijentski dio zadužene su tehnologije poput Reacta i Bootstrapa dok su za poslužiteljski dio korištene tehnologije Node.js i PostgreSQL. Aplikacija omogućuje stvaranje proizvodnih naloga pomoću kojih je moguće pratiti različite poslovne procese.

Informacijski sustavi sveobuhvatna su rješenja koja integriraju i optimiziraju različite aspekte poslovanja, odnosno u ovom slučaju proizvodnje. Oni tvrtkama pružaju jedinstvenu platformu za upravljanje nad procesima, poboljšanjem učinkovitosti, donošenjem informiranih odluka i postizanjem održivog rasta.

7. LITERATURA

1. **Nađ, Josip.** *Programsko inženjerstvo i informacijski sustavi.* Čakovec, Međimursko Veleučilište u Čakovcu, 2020.

2. **Bernot, Matt.** ERP Safety and Cybersecurity: What You Need To Know. *www.striven.com.* [Mrežno] [Citirano: 1. 6 2023.] https://www.striven.com/blog/erp-safety-and-cybersecurity-what-you-need-to-know.

3. **Lemonaki, Dionysia.** Frontend VS Backend – What's the Difference? *FreeCodeCamp.* [Mrežno] [Citirano: 1. 6 2023.] https://www.freecodecamp.org/news/frontend-vs-backend-whats-the-difference/.

4. **Herbert, David.** What is React.js? (Uses, Examples, & More). *Hubspot.* [Mrežno] 6. 27 2022. [Citirano: 1. 6 2023.] https://blog.hubspot.com/website/react-js.

5. Bootstrap 3 Tutorial. *W3schools.* [Mrežno] [Citirano: 1. 6 2023.] https://www.w3schools.com/bootstrap/.

6. About PostgreSQL. *postgresql.org.* [Mrežno] [Citirano: 1. 6 2023.] https://www.postgresql.org/about/.

7. About Nodjes. *www.nodejs.org.* [Mrežno] [Citirano: 1. 6 2023.] https://nodejs.org/en/about.

8. What Is Hashing, and How Does It Work? www.codecademy.com. [Mrežno] [Citirano: 23.8.2023.] https://www.codecademy.com/resources/blog/what-is-hashing/

9. Crypto: randomUUID() method. developer.mozilla.org. [Mrežno] [Citirano: 28.3.2023.] https://developer.mozilla.org/en-US/docs/Web/API/Crypto/randomUUID

8. POPIS SLIKA

9. POPIS KÔDOVA