Dokumentacija za Zadaću broj 2

ZALIHE

Radila: Amila Huzbašić

Index:228

Predmet: Web programiranje

Overview:

Backend za aplikaciju ZALIHE izgrađen je pomoću Node.js, popularne platforme na server strani koja se temelji na JavaScriptu. Odgovorana je za upravljanje razmjenom podataka između sučelja i baze podataka, pružajući sigurne i brze API-je za pristup aplikaciji. Node.js je odabran zbog svoje sposobnosti rukovanja s više zahtjeva istovremeno, učinkovitog upravljanja memorijom i velike biblioteke modula. Backend je dizajniran za učinkovito i sigurno pohranjivanje, obradu i dohvaćanje podataka te upravljanje procesom autentifikacije korisnika. Sve u svemu, backend na Node.js pruža stabilnu i skalabilnu osnovu za aplikaciju ZALIHE.

Projektni zadatak:

Sistem za upravljanje zalihama je sistem čija je osnovna zadaća ubrzati poslovanje organizacije. Organizacije čije se poslovanje zasniva na sirovinama čine ogromne napore kako bi što efikasnije koristili materijale u skladu sa njihovim potrebama i zahtjevima. U takvim prilikama organizacija mora obaviti niz zadataka i operacija kako bi uspješno vodila svoje poslovanje bez automatizacije sistema (manuelno).

Zbog toga je vaš zadatak napraviti web aplikaciju koja će automatizirati upravljanje zalihama neke organizacije. Takvu jedna aplikacija može se koristiti u bilo kojem sektoru industrije, mada sistem može biti korišten i u drugim sektorima (bankarski, zdravstveni i slično), ali ovaj projekat primarnot treba da bude namjenjen maloj i/ili velikoj industriji.

Backend web aplikacije potrebno je izgraditi u Node/Express web tehnologijama. Svi detalji moraju se čuvati u SQL bazi podataka (mysql za primjer). Opciono, frontend web aplikacije izgraditi u Reactu ili Angularu.

Sistem treba da omogući lako upravljanje i održavanje zaliha. Upravljanje zalihama treba biti razvijeno na način da korisnicima omogući dodavanje inventara, izmjena i brisanje inventara, unesu količine zaliha i druge detalje, ažuriraju status zaliha i još mnogo toga.

Vaše rješenje treba da sadrži sljedeće elemente:

• Omogućiti upravljanje zaposlenicima organizacije

• Omogućiti upravljanje zalihama u skladištu

• Omogućiti upravljanje dobavljačima

• Omoguićiti upravljanje troškovima

• Upravljane proizvodima

• Upravljanje proizvodnim procesom

Aplikaciju trebate podjeliti u dva modulu: Admin modul i modul za zaposlenika.

Zadaci zaposlenika su sljedeći:

• Upravljanje zalihama u skladištu (dodavanje, ažuriranje i brisanje)

• Upravljanje dobavljačima (dodavanja i ažuriranje podataka)

• Omogućiti upravljanje troškovima (omogućiti izvještaje o troškovima)

• Omogućiti upravljanje proizvodima (dodavanje, ažuriranje podataka)

Zadaci admina su sljedeći:

• Sve što je definisno za zaposlenika

• Upravljanje zaposlenicima (dodavanje i ažuriranje)

Detalji sistema:

U bazi podataka potrebno je da čuvate sljedeće informacije:

Korisnici: zaposlenik_id, korisnicko_ime, sifra, uloga

Zaposlenici: id, ime, prezime, broj_telefona, adresa, emai_adresa, datum_zaposlenja, datum_otkaza

Sirovine: id, naziv, kolicina, min_kolicina, cijena, jedinica_mjere, da_li_se_koristi, dobavljac_id

Dobavljaci: id, naziv, jib (jedinstveni identifikacioni broj), pdv (pdv broj), broj_telefona, kontakt_osoba, email_adresa, datum_pocetka, datum_zavrsetka

Proizvodi: id, naziv, slika_proizvoda (čuvati url slike), proizvodni_proces_id, marza, cijena

Proizvodni_proces: id, naziv, datum_pocetka, datum_završetka, cijena (Napomena, proizvodni proces su aktivni određeni vremenski period. U svom poslovanju organizacija može proglasiti proizvodni proces zastarjelim, i poreban je novi. To će definisati početnim i krajnim datumom. Ako proizvodni proces ima postavljen datum_zavrsetka, on se više ne koristi, a ako ne, onda je on u upotrebi. U datom trenutku samo jedan proizvodni proces može biti aktivan).

Proizvodi_proces_stavka: id, sirovina_id, kolicina

Aplikacija treba da sadrži sljedeće module (dijelove):

Prijava na sistem

Svaki zaposlenik se mora prijaviti na sistem za korisničkim imenom i šifrom koja mu je dodjeljena. Nijednom dijelu aplikacije se ne može pristupiti ako zaposlenik nije prijavljen. Svaki zaposlenik kojem je datum otkaza prazno polje u bazi podatak je trenutno zaposlen u organizaciji i može pristupiti sistemu. Ako je dobio otkaz (ili dao otkaz) nije mu više omogućen pristup sistemu.
Veza između tabele Korisnici i Zaposlenici je jedan na jedan (1:1)
Zaposlenici se ne mogu registrovati na sistem, njih u sistem unosi admin organizacije.

Promjena šifre

Svaki zaposlenik ima mogućnost da promjeni svoju šifru koju je dobio prilikom zaposlenja. Poželjno je da zaposlenik promjeni svoju prvu šifru, ali aplikacija ga ne treba na to podsjećati.

Upravljanje zaposlenicima

Admin organizacije ima mogućnost da upravlja podacima o zaposlenicima. Admin ima mogućnost da doda podatke o zaposlenicima u bazi podataka. Admin može promjeniti podatke o zaposleniku,

ali ne može ga izbrisati iz sistema. Ako zaposlenik više nije dio organizacije, promjeni se njegov datum otkaza (postavi se na neki datum). Svaki zaposlenik kojem je datum otkaza prazno polje u bazi podataka je aktivni zaposlenik i on se može prijaviti na sistem.
Svaki zaposlenik može biti u jednoj od dvije uloge: Zaposlenik ili Admin.

Upravljanje dobavljačima

Svaki zaposlenik ima mogućnost da upravlja podacima o dobaljačima sirovina i da ima pregled svih podataka o njima. Potrebno je prikazati listu svih dobavljača, kao i mogućnosti da se podaci

izmjene.

Upravljanje sirovinama

Svaki zaposlenik ima mogućnost da upravlja sirovinama koje su potrebne za poslovanje organizacije. Za svaku sirovinu definiše se dobavljač koji je dobavlja za organizaciju.
Potrebno je prikazati listu svih sirovina, stranicu za dodavanje sirovina, stranicu za izmjenu sirovina. Minimalna količina za sirovinu je količina koju organizacija mora imati kako bi uspješno obavila svoje poslovanje. U slučaju da količina sirovina padne ispod minimalne količina potrebno je napraviti automatsku narudžbu prema dobavljaču (ovaj dio nije potrebno implementirati, ali lijepo ga je imati).

Upravljanje proizvodima

Svaki uposlenik ima mogućnost da upravlja podacima o proizvodu.
Prilikom unosa podataka o proizvodu potebno je definisati i proizvodni proces za taj proizvod. Cijena proizvoda se definiše na osnovu cijena sirovina i njigovih količina koje su potrebne da se proizvod napravi pomnože na sa maržom za taj proizvod. Marža je nešto što unosi zaposlenik i definisana je u procentima. Cijena koja se dobije na gore navedeni način je cijena bez pdv-a, ali na stranici o proizvodu je potrebno prikazati i cijenu sa pdv-om, kao i iznos pdv-a. U bazi se čuva cijena bez PDV-a.

Upravljanje proizvodnim procesima

Svaki uposlenik može dodavati proizvodne procese i mjenajti njihove podatke. Proizvodni procesi se definišu odvojeno od proizvoda, ali se dodjeljuju određenom proizvodu kako je to i navedeno ranije. Za svaki proizvodni proces definišu se stavke tog proizvodnog procesa, a koje se sastoje od

podataka o sirovini koja će se koristiti te u kojoj količini. Cijenu ne unosi zaposlenik, već se onda definiše na osnovu cijene sirovine pomnožene sa količinom koja je potrebna za taj proizvodni

proces.

Sadržaj

Uvod

Projekat ZALIHE ima za cilj pružiti organizacijama rješenje za učinkovito upravljanje svojim inventarnim operacijama. Sistem je dizajniran za automatizaciju ručnih procesa, pojednostavljivanje zadataka i operacija uključenih u upravljanje sirovinama i zalihama. Web aplikacija izgrađena je korištenjem Node.js i Express web tehnologija, te koristi SQL bazu podataka, kao što je MySQL, za pohranu svih potrebnih detalja. Cilj je stvoriti user-friendly i jednostavnu platformu koja će pojednostaviti upravljanje i održavanje zaliha. Sustav će korisnicima omogućiti izvršavanje zadataka kao što su dodavanje, izmjena i brisanje zaliha, unos količina zaliha i drugih detalja te ažuriranje statusa zaliha. Ovaj je projekt namijenjen za korištenje u industrijskom sektoru, iako se može prilagoditi za korištenje u drugim sektorima kao što su bankarstvo i zdravstvo. Projekt ZALIHE ključan je alat organizacijama za optimizaciju poslovanja i poboljšanje ukupne učinkovitosti.

Baza podataka

Ova baza podataka pod nazivom dbzalihe_228 ima više tabela:

user tabela sadržava informacije o useru kao id, name, contact number, email, password, status I role. Email column ima jedinstveno ograničenje.

sirovine tabela sadržava informacije o sirovinama kao id i name.

proizvodi sadržava informacije o proizvodima uključujući id, name, sirovineId, description, price, i status.

dobavljaci sadržava podatke o dobavljačima uključujući id, name, jib, VAT, phone number, contact person, email address, start date, i end date.

Proizvodni_procesi tabela sadržava informacije o proizvodnim procesima uključujući id, name, start date, end date, and price. Stupac name ima jedinstveno ograničenje i end date mora biti veći od start date.

Proizvodi_proces_stavka sadržava informacije kao što su id, sirovina_id, i quantity.

Dodan je primjer korisnika sa nazivom '**Admin**', e-poštom **'admin@gmail.com'** i ulogom **'admin'** u tabelu korisnika.

.ENV

Datoteka .env koristi se za pohranjivanje varijabli okoline za projekt Node.js. U ovoj .env datoteci postoji nekoliko parova ključ-vrijednost koji su definirani:

PORT: Ova se varijabla koristi za postavljanje broja priključka na kojem će server slušati. U ovom slučaju, postavljeno je na 8080.

DB_PORT, DB_HOST, DB_USERNAME, DB_PASSWORD i DB_NAME: Ove se varijable koriste za pohranjivanje detalja veze za MySQL bazu podataka. Broj porta baze podataka postavljen je na 3306, host je postavljen na localhost, korisničko ime je postavljeno na root, te je lozinka postavljena a naziv baze podataka je postavljen na "dbzalihe_228".

ACCESS_TOKEN: Ova varijabla je vrijednost niza koja se koristi kao pristupni token.

USER: Ova varijabla je postavljena na "user".

Ove varijable okruženja mogu se koristiti u kodu pozivanjem na objekt process.env. Na primjer, process.env.PORT bi vratio vrijednost "8080".

Server.js

Server.js datoteka je ulazna tačka web aplikacije. Odgovoran je za pokretanje servera i dopuštanje aplikaciji da sluša na određenom portu.

Prvi redak datoteke zahtijeva biblioteku dotenv, koja učitava varijable okoline iz .env datoteke.

Zatim zahtijeva "http" modul i glavnu aplikacijsku datoteku "index". Metoda "http.createServer" stvara instancu HTTP servera i prosljeđuje joj glavnu aplikacijsku datoteku "index" kao argument.

Na kraju, server sluša port naveden u .env datoteci pomoću varijable okruženja "process.env.PORT". To omogućuje aplikaciji primanje dolaznih HTTP zahtjeva na navedenom priključku i odgovaranje na njih u skladu s tim.

Connection.js

connection.js datoteka koristi se za stvaranje veze s MySQL bazom podataka pomoću mysql biblioteke. Parametri veze kao što su port, host, korisničko ime, lozinka i naziv baze podataka dohvaćaju se iz varijabli okruženja definiranih u .env datoteci. Stanje veze bilježi se na konzoli. Ako je veza uspješna, bilježi se poruka "Connected", inače se bilježi poruka o pogrešci. Objekt veze se izvozi kao modul za korištenje u drugim dijelovima aplikacije.

/services/

/services/authentication.js

Ovaj kod je Node.js modul koji izvozi međuversku funkciju pod nazivom "authenticateToken" za provjeru autentičnosti JSON Web Tokena (JWT). Funkcija provjerava prisustvo zaglavlja "Authorization" u zahtjevu i izdvaja token. Zatim provjerava token sa tajnim ključem pohranjenim u varijabli okruženja "ACCESS_TOKEN". Ako je token valjan, dekodirani korisni teret se pohranjuje u svojstvu "locals" objekta odgovora i poziva se sljedeća funkcija međuopreme. Ako token nije pronađen ili je nevažeći, funkcija šalje status odgovora 401 Neovlašteno ili 403 Zabranjeno.

/services/checkRole.js

Ovaj kod izvozi funkciju "checkRole" koja implementira međuopremu za provjeru uloge korisnika u aplikaciji.

On preuzima ulogu iz objekta "res.locals" i uspoređuje je sa vrijednošću pohranjenom u varijabli okruženja "process.env.USER".

Ako se uloga ne podudara s očekivanom vrijednošću, funkcija klijentu šalje HTTP 401 Neovlašteno odgovor. U suprotnom, poziva sljedeći međuverski softver u lancu koristeći "next()".

Izvezeni objekat ima par ključ/vrijednost gdje je ključ "checkRole", a vrijednost je funkcija "checkRole".

/routes/

/routes/user.js

Datoteka user.js u mapi ruta implementira logiku za korisničke krajnje tačke web aplikacije, kao što su prijava, prijava i zaboravljena lozinka. Kod koristi sljedeće pakete/biblioteke: express, body-parser, connection, jsonwebtoken, nodemailer, dotenv i autentikacija.

router.post('/signup', async (req, res) => {...: Ova se krajnja tačka koristi za registraciju korisnika. Ako korisnikova e-pošta ne postoji, kreira se novi korisnički račun s navedenim detaljima.

router.post('/login', (req, res) => {...: Ova se krajnja tačka koristi za prijavu korisnika. Ako se e-pošta i lozinka podudaraju s bazom podataka, JSON web token se generira i šalje kao odgovor .

router.post('/forgotPassword', (req, res) => {...: Ova se krajnja tačka koristi za poništavanje korisničke lozinke. E-pošta se šalje na e-poštu korisnika s detaljima za prijavu.

/routes/sirovine

Kod definira skup krajnjih tačaka REST API-ja za upravljanje "sirovine" resursima koristeći Express.js okvir. Rute koriste sljedeće karakteristike:

Međuverzija za autentifikaciju i provjeru uloge korisnika: auth.authenticateToken i checkRole.checkRole.

Povezivanje na bazu podataka pomoću objekta veze.

4 krajnje tačke:

POST /add za dodavanje novog "sirovine" resursa.

GET /get za preuzimanje svih "sirovine" resursa.

PATCH /ažuriranje za ažuriranje postojećeg "sirovine" resursa.

DELETE /delete/:id za brisanje postojećeg "sirovine" resursa po njegovom ID-u.

Krajnje tačke koriste parametre upita ili URL parametre da dohvate ili ažuriraju "sirovine" resurs i vrate statusne kodove na osnovu ishoda operacije (npr. 200 za uspjeh, 404 za resurs koji nije pronađen, 500 za grešku servera).

/routes/proizvodi.js

proizvodi.js je dio Node.js Express.js web aplikacije, koja izlaže RESTful API-je za CRUD operacije na podacima o proizvodu. Podaci o proizvodu pohranjeni su u bazi podataka (SQL), a modul za povezivanje se koristi za interakciju s bazom podataka. Modul ruta koristi Express.js Router() za rukovanje HTTP zahtjevima i odgovorima. Modul također koristi module autentifikacije i checkRole za provjeru autentičnosti zahtjeva i provjeru korisničkih uloga.

Rute definirane u ovom modulu uključuju:

/add: Post zahtjev za dodavanje novog proizvoda u bazu podataka.

/get: Get zahtjev za preuzimanje svih proizvoda iz baze podataka.

/getbySirovine/:id: Zahtjev za dobivanje proizvoda na osnovu sirovineId-a u URL putanji.

/getbyId/:id: zahtjev za preuzimanje proizvoda na osnovu ID-a u URL putanji.

/update/:id: Zahtjev za update za ažuriranje postojećeg proizvoda u bazi podataka.

/delete: Zahtjev za brisanje za brisanje proizvoda iz baze podataka.

/updateStatus: Zahtjev za update za ažuriranje statusa proizvoda u bazi podataka.

/routes/dobavljaci.js

Ova skripta izvozi skup API krajnjih tačaka za CRUD (Create, Read, Update, Delete) operacije na "dobavljacima" koristeći Express.js framework i MySQL bazu podataka.

Svaka krajnja tačka koristi sintaksu "async/await" za rukovanje asinkronim operacijama baze podataka.

Krajnje tačke su:

GET "/": Preuzmi sve dobavljače iz tabele "dobavljaci" u bazi podataka i vrati rezultat kao JSON odgovor.

GET "/:id": Preuzmi jednog dobavljača po njegovom ID-u iz tabele "dobavljaci" u bazi podataka i vrati rezultat kao JSON odgovor. Ako nije pronađen dobavljač sa datim ID-om, vratite odgovor "400 Bad Request" sa JSON porukom "Nemoguce pronaci dobavljaca" (Ne mogu pronaći dobavljača).

POST "/": Kreirajte novog dobavljača u tabeli "dobavljaci" na osnovu podataka u telu zahteva, koji treba da bude u JSON formatu. Vratite novokreiranog dobavljača kao JSON odgovor sa statusnim kodom "201 Created".

ZAKRPA "/:id": Ažurirajte postojećeg dobavljača u tabeli "dobavljaci" na osnovu podataka u telu zahteva i ID-a dobavljača navedenog u URL-u. Vratite ažuriranog dobavljača kao JSON odgovor.

DELETE "/:id": Izbrišite dobavljača iz tabele "dobavljaci" na osnovu ID-a navedenog u URL-u. Vratite JSON odgovor sa porukom "Obrisi dobavljaca" (izbrisan dobavljač). Ako dobavljač sa navedenim ID-om nije pronađen, vratite odgovor "404 Not Found" sa JSON porukom "Nemoguce pronaci dobavljaca" (Nije moguće pronaći dobavljača).

/routes/upravljanjeDobavljacima.js

Ovaj kod je Node.js Express.js web aplikacija koja implementira RESTful API za upravljanje dobavljačima (koji se u kodu nazivaju "dobavljaci"). Koristi Express.js okvir za rukovanje dolaznim HTTP zahtjevima i odgovorima na te zahtjeve. Zahtijeva module express i veze, kao i prilagođeni modul checkRole, koji se koristi za provjeru uloge korisnika. Kod definira krajnje točke za rukovanje sljedećim radnjama:

Nabavite sve dobavljače

Nabavite određenog dobavljača

Ažurirajte dobavljača

Kod definira funkcije međuopreme za rukovanje zahtjevima za određenog dobavljača (getdobavljaci), koji preuzima dobavljača po njegovom ID-u i sprema dobavljača u objekt odgovora za naknadnu obradu od strane funkcija rukovatelja rute. Kod izvozi Express.js objekt rutera kao modul, koji drugi dijelovi aplikacije mogu koristiti za rukovanje dolaznim zahtjevima.

/routes/upravljanjesirovinama.js

Ovaj kod izvozi objekt rutera izgrađen koristeći Express.js framework. Sadrži tri REST API krajnje tačke koje su u interakciji sa bazom podataka:

GET /sirovine: Odabire sve redove iz tabele sirovine i šalje JSON odgovor rezultata.

POST /sirovine: Dodaje novi red sirovine tabeli sa imenom proslijeđenim u tijelo zahtjeva. Šalje JSON odgovor s porukom o uspjehu.

PUT /sirovine/:id: Ažurira red u sirovine tabeli sa navedenim ID-om u URL-u, postavljajući njegovo ime na vrijednost proslijeđenu u tijelu zahtjeva. Šalje JSON odgovor s porukom o uspjehu.

Kod uvozi dva modula, ekspresni i konekcioni, i zahtijeva provjeru modula međuverskog modula za autorizaciju korisnika. Koristi nereferencirani klijentski objekt za izvršavanje SQL upita.

/routes/upravljanjeProizvodima.js

Ovo je Node.js skripta koja izvozi Express.js ruter sa 4 krajnje tačke za upravljanje proizvodima:

GET krajnja točka koja preuzima sve proizvode iz baze podataka i vraća ih u JSON formatu.

POST krajnja točka koja dodaje novi proizvod u bazu podataka uzimajući informacije o proizvodu iz tijela zahtjeva. Također izračunava cijenu proizvoda, PDV i sprema ih u bazu podataka.

PUT krajnja točka koja ažurira proizvod u bazi podataka na osnovu ID-a proizvoda navedenog u URL-u zahtjeva. Također izračunava ažuriranu cijenu proizvoda, PDV i sprema je u bazu podataka.

Krajnja točka DELETE koja briše proizvod iz baze podataka na osnovu ID-a proizvoda navedenog u URL-u zahtjeva.

Skripta zahtijeva Express.js i Sequelize biblioteke za pokretanje i pristup podacima o proizvodima i sirovinama iz baze podataka. Također koristi "checkRole" modul za kontrolu pristupa zasnovanu na ulozi.

/routes/proizvodniproces.js

Ovaj kod izvozi ruter u Express framework koji pruža CRUD (kreiranje, čitanje, ažuriranje, brisanje) operacije na tablici "Proizvodni_procesi" pohranjenoj u bazi podataka (predstavljenoj db objektom). API prihvaća HTTP zahtjeve s odgovarajućim metodama (GET, POST, PUT, DELETE) za izvođenje operacija.

GET / vraća JSON niz sa statusom, greškom i objektima odgovora, koji predstavlja sve redove u tabeli "Proizvodni_procesi".

POST / ubacuje novi red u tablicu "Proizvodni_procesi" koristeći podatke iz tijela zahtjeva sa svojstvima name, start_date i end_date.

PUT / ažurira postojeći red u tabeli "Proizvodni_procesi" koristeći podatke iz tijela zahtjeva sa svojstvima name, start_date, end_date i id.

DELETE / briše red u tabeli "Proizvodni_procesi" koristeći id iz tijela zahtjeva.

Zaključak:

ZALIHE sistem za upravljanje zalihama je web aplikacija koja automatizira upravljanje zalihama organizacija. Izgrađen je korištenjem Node i Express web tehnologija, a detalji se pohranjuju u SQL bazu podataka (npr. MySQL). Frontend aplikacije može se napraviti pomoću Reacta ili Angulara. Osnovni cilj sistema je poboljšanje efikasnosti poslovanja organizacije omogućavanjem lakog upravljanja i održavanja zaliha. Sistem uključuje nekoliko ključnih karakteristika kao što su upravljanje zaposlenima, upravljanje zalihama, upravljanje dobavljačima, upravljanje troškovima, upravljanje proizvodima i upravljanje proizvodnim procesom. Ove karakteristike imaju za cilj da pojednostave poslovanje organizacije i osiguraju da su potrebe organizacije zadovoljene.