Razvoj web aplikacije za praćenje najma sportske opreme

Muller, Matej

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

Matej Muller

RAZVOJ WEB APLIKACIJE ZA PRAĆENJE NAJMA SPORTSKE OPREME

Završni rad

Pula, rujan 2024.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

Matej Muller

RAZVOJ WEB APLIKACIJE ZA PRAĆENJE NAJMA SPORTSKE OPREME

Završni rad

JMBAG: 0303095014, redovni student
Studijski smjer: Informatika
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Kolegij: Strukture podataka i algoritmi
Mentor: Izv. prof. dc. sc. Tihomir Orehovački

Pula, rujan 2024.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani MATEJ MULLER, kandidat za prvostupnika INFORMATIKE 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, 04.09.2024

IZJAVA O KORIŠTENJU AUTORSKOGA DJELA

Ja, MATEJ MULLER, dajem odobrenje Sveučilištu Jurja Dobrile u Puli, nositelju prava korištenja, da moj završni rad pod nazivom „RAZVOJ WEB APLIKACIJE ZA PRAĆENJE NAJMA SPORTSKE OPREME“ 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, 04.09.2024

SADRŽAJ

1. UVOD

U današnjem digitalnom dobu, posjedovanje funkcionalne i intuitivne web aplikacije može značajno poboljšati poslovanje, omogućujući obrtima da dosegnu širu publiku, povećaju efikasnost poslovanja i povećaju prihode. Web aplikacije omogućavaju obrtima da svoje usluge učine dostupnima 24/7, što znači da korisnici mogu pregledavati, rezervirati i plaćati za usluge u bilo koje vrijeme, bez obzira na radno vrijeme obrta. Navedeno značajno povećava dostupnost usluga i može privući više klijenata koji preferiraju online rezervacije. Osim toga, digitalizacija poslovanja omogućava bolje upravljanje inventarom, optimizaciju rasporeda i učinkovitije upravljanje resursima. Razvoj web aplikacije u svrhu unaprijeđenja obrta pruža značajnu priliku obrtnicima da unaprijede svoje poslovanje i iskoriste prednosti digitalizacije. Mogu koristiti povratne informacije za poboljšanje kvalitete i prilagodbu radnog okruženja, pa i ponude. Obrti mogu prikupljati podatke o ponašanju korisnika, popularnosti određenih artikala i vremenskim razdobljima najveće potražnje (Sharp et al., 2019). Ove informacije omogućuju vlasnicima da donose informirane odluke o nabavi nove opreme, prilagodbi cijena i promocija, te optimizaciji poslovnih procesa. Jedna od najvećih prednosti web aplikacija je njihova fleksibilnost i skalabilnost. Kako obrt raste aplikacija se može prilagoditi novim potrebama i zahtjevima tržišta. Dodavanje novih funkcionalnosti, prilagodba postojećih značajki i širenje poslovanja na nova tržišta postaje puno jednostavnije uz digitalnu platformu koja podržava takve promjene.

Cilj izrade ovog završnog rada je detaljno opisati proces razvoja web aplikacije za najam opreme za vodene sportove. Namjera je pružiti sveobuhvatan pregled svih faza projekta, od inicijalne ideje do implementacije i testiranja. Biti će objašnjen koncept u obliku use case dijagrama i ciljevi aplikacije, uključujući unapređenje dostupnosti usluga, povećanje efikasnosti poslovanja i poboljšanje korisničkog iskustva koji uključuju: planiranje, dizajn, implementaciju, testiranje i održavanje. Poseban naglasak je na prikazu i objašnjenju koda koji je pisan tijekom razvoja aplikacije. Biti će uključeni primjeri koda i tehnički detalji kako bi se ilustrirali razvojni procesi i rješenja koja su primijenjena. Opisan je i veliki broj tehnologija koje su korištene u razvoju web aplikacije. Objašnjeni su razlozi za odabir specifičnih tehnologija i alata te njihove prednosti i izazovi, a posebno su obrađeni dizajn korisničkog sučelja (UI) i korisničkog iskustva (UX).

Poveznica na projekt: https://github.com/mullermatej/rentrack-web-app

2. PRIPREMA RADNOG OKRUŽENJA

2.1 Instalacija VS Code-a

Prvi korak u pripremi okruženja za rad na web aplikaciji za najam opreme za vodene sportove jest instalacija Visual Studio Code-a. VS Code je popularan kod editor poznat po svojoj prilagodljivosti i širokom spektru ekstenzija koje olakšavaju razvoj. Jedna od ključnih prednosti korištenja VS Code-a je njegova prilagodljivost putem ekstenzija. Postoji veliki broj ekstenzija koje mogu značajno poboljšati produktivnost i efikasnost rada. Ekstenzije kao što su ESLint i Prettier pomažu u održavanju konzistentnog stila koda i automatskom formatiranju, dok Tailwind CSS IntelliSense pruža korisne savjete za korištenje Tailwind CSS klasa. Također, Material UI Snippets ekstenzija omogućuje brzo umetanje komponenti iz Material UI biblioteke, čime se ubrzava proces razvoja korisničkog sučelja.

2.2 Postavljanje Node.js i npm-a

Za korištenje programskog jezika JavaScript, potrebno je instalirati Node.js i npm (node package manager). Node.js omogućuje pokretanje JavaScript koda izvan preglednika, dok npm olakšava upravljanje paketima (Node.js, 2024). Programski kod 1 prikazuje naredbe u konzoli pomoću kojih se provjerava jesu li Node.js i npm uspješno instalirani.

```
node -v
npm -v


v20.11.1
10.2.4
```

Programski kod 1. Node.js i npm naredbe za provjeru verzije

Ove naredbe prikazati će verzije instaliranih alata, što potvrđuje da su Node.js i npm uspješno instalirani. Ako su verzije prikazane onda su alati spremni za korištenje. Korištenjem npm-a, jednostavno se mogu dodavati novi paketi, ažurirati postojeći i upravljati potrebnim paketima projekta, što značajno olakšava razvojni proces.

2.3 Kreiranje React aplikacije

Za kreiranje nove React aplikacije korišten je Vite. Vite je brz i lagan alat za izgradnju modernih web aplikacija. Razvio ga je Evan You, tvorac Vue.js -a. Vite je dizajniran kako bi poboljšao performanse razvoja i pružio bržu i efikasniju alternativu alatima kao što su Webpack i Create React App. Njegova glavna svrha je omogućiti brže pokretanje i brži razvoj koristeći ESM (ECMAScript module) za brže učitavanje modula tijekom razvoja. Vite nudi trenutno pokretanje servera, optimizirani build proces i bogat ekosustav dodataka, što ga čini idealnim izborom za razvoj moderne web aplikacije (React, 2024). Programski kod 2 predstavlja početak kreiranja aplikacije naredbom u terminalu.

```
npm create vite@latest rentrack  --template react
cd rentrack
npm install
```

Programski kod 2. Kreiranje početnog predloška React aplikacije koristeći Vite.

2.4 Konfiguracija Tailwind CSS-a

Tailwind CSS je "utility-first" CSS framework koji omogućuje brzo i efikasno stiliziranje komponenti. Umjesto pisanja prilagođenih CSS klasa za svaki element, Tailwind koristi skup predefiniranih klasa koje se mogu kombinirati za stvaranje složenih dizajna direktno u HTML ili JSX datotekama. Ovaj pristup omogućuje brže stiliziranje i bolju kontrolu nad izgledom aplikacije bez potrebe za pisanjem puno prilagođenog CSS-a (Tailwind, 2020). Programski kod 3 prikazuje jedan od načina za instalirati tailwind koristeći npm.

```
npm install tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

Programski kod 3. Početne naredbe za inicijalizaciju tailwind css-a

Nakon instalacije potrebno je konfigurirati tailwind.config.js dodavanjem putanja do React datoteka kako bi Tailwind mogao pravilno procesirati CSS klase. Programski kod 4 sadrži kod tailwind.config.js datoteke u kojoj je potrebno izmijeniti content polje.

```
module.exports = {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Programski kod 4. Početno postavljanje tailwind.config.js datoteke

Zatim je potrebno dodati Tailwind direktive u src/index.css datoteku. Programski kod 5 sadrži potrebne direktive.

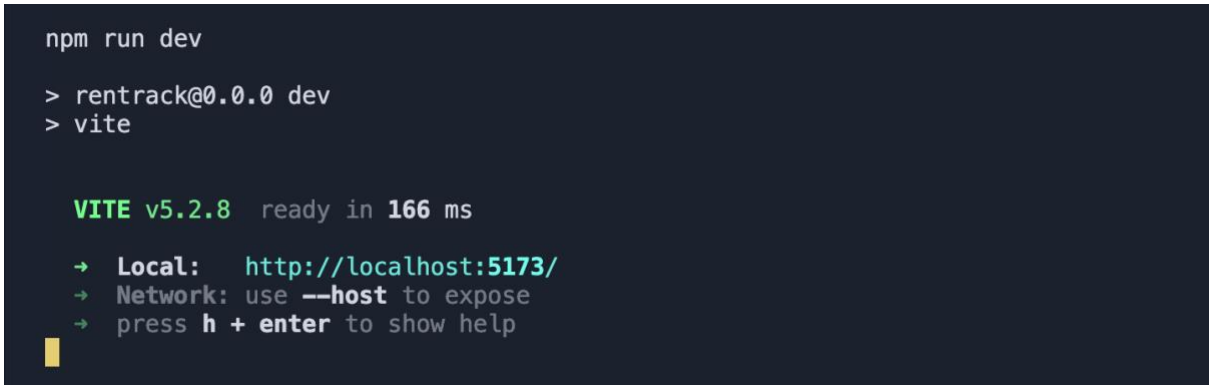
```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Programski kod 5. Uključivanje osnovnih tailwind biblioteka

Dodavanjem ovih direktiva, osigurava se da su svi osnovni stilovi i klase dostupni unutar cijelog projekta. Tailwind CSS značajno pojednostavljuje proces stiliziranja i omogućuje brzo eksperimentiranje s dizajnom. Kombiniranjem utility klasa moguće je brže testirati različite stilove i prilagoditi izgled komponenti bez potrebe za pisanjem dodatnog CSS koda. Ovaj pristup ne samo da ubrzava razvoj, već također održava kod čistim i lako održivim.

2.5 Pokretanje projekta

Nakon što su svi potrebni alati i biblioteke instalirani potrebno je pokrenuti projekt kako bi se provjerilo da je sve ispravno postavljeno. Ova naredba će pokrenuti lokalni razvojni server i otvoriti projekt u pregledniku. U terminalu unutar direktorija potrebno je pokrenuti naredbu za pokretanje aplikacije, kao što prikazuje programski kod 6.

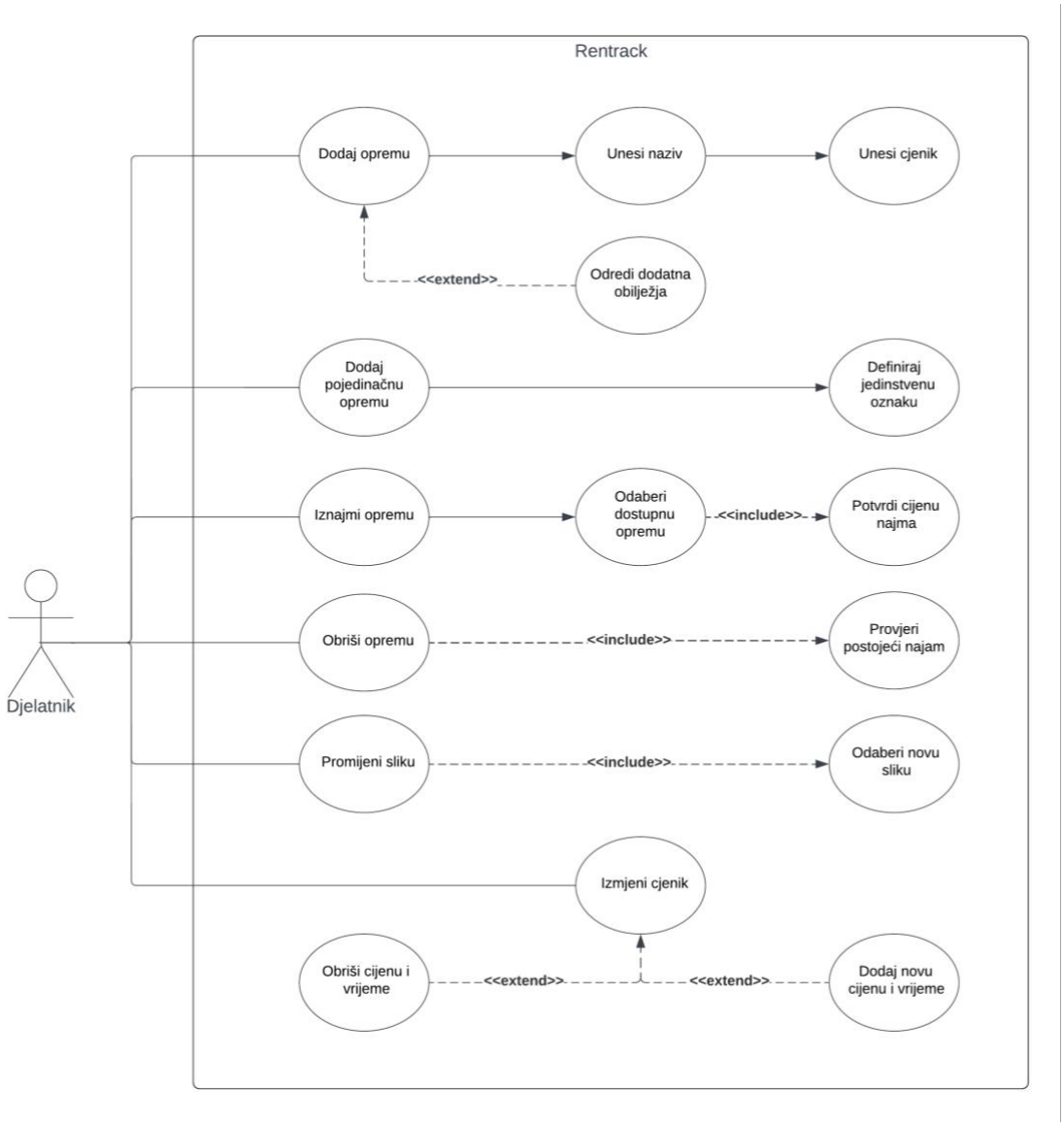


Programski kod 6. Pokretanje aplikacije sa npm run dev naredbom

3. PROCES RAZVOJA WEB APLIKACIJE "RENTRACK"

3.1 Use Case dijagram

Kako bi se pojednostavio smisao i slika aplikacije napravljen je use case dijagram. Pomoću ovog dijagrama jasno se određuju mogućnosti i funkcionalnosti kojima djelatnik ima pristup i što je sve moguće unutar aplikacije. Slika 1 prikazuje use case dijagram aplikacije.



Slika 1. Use Case dijagram

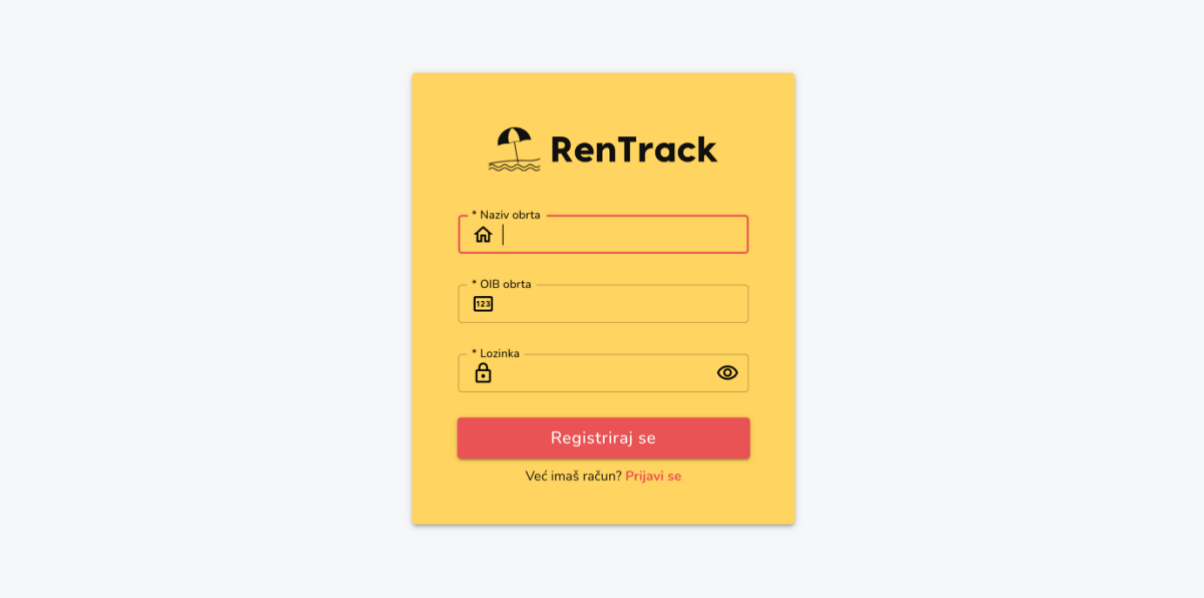
Use case dijagram prikazuje djelatnikov hod kroz aplikaciju. Djelatnik ima dvije osnovne mogućnosti, dodavanje potpuno nove kategorije opreme i dodavanje pojedinačne opreme. Za dodavanje potpuno nove opreme djelatnik je dužan unijeti njen naziv i cjenik, dok su dodatna obilježja neobavezna. Što se tiče pojedinačne opreme djelatnik je dužan dodati samo

jedinstvenu oznaku opreme. Kod najma opreme potrebno je odabrati jednu od pojedinačne opreme koja ima oznaku "dostupno", te jednu od određenih opcija cijena i trajanja najma. Brisanje povlači obaveznu provjeru gdje se pretražuju sve pojedinačne opreme i da li je koja u najmu, sprječavajući brisanje opreme koja je zauzeta. Djelatnik može promijeniti sliku opreme i taj proces obavezno zahtjeva novu sliku u jednom od podržanih formata. Posljednje, moguće je i uređivanje postavki cjenika opreme gdje djelatnik može obrisati postojeću ili dodati novu cijenu i vrijeme najma.

3.2 Autentifikacija

3.2.1 Registracija

Za ulaz u aplikaciju potrebno je registrirati obrt. Slika 2 prikazuje sučelje sa potrebnim informacijama za uspješnu registraciju.



Slika 2. Stranica za registraciju

OIB je jedinstvena identifikacijska oznaka za svaki obrt i pri upisu OIB-a koji već postoji aplikacija javlja grešku. Programski kod 7 prikazuje asinkronu funkciju koja se poziva pri svakom pokretanju poslužitelja i ona ima dvije svrhe: povezati se sa bazom podataka i kreirati indeks na polju "oib" u kolekciji "users".

```
(async () => {
    const db = await connect();
    await db.collection('users').createIndex({ oib: 1 }, { unique: true });
})();
```

Programski kod 7. Asinkrona callback funkcija na poslužitelju

3.2.2 Kriptiranje podataka

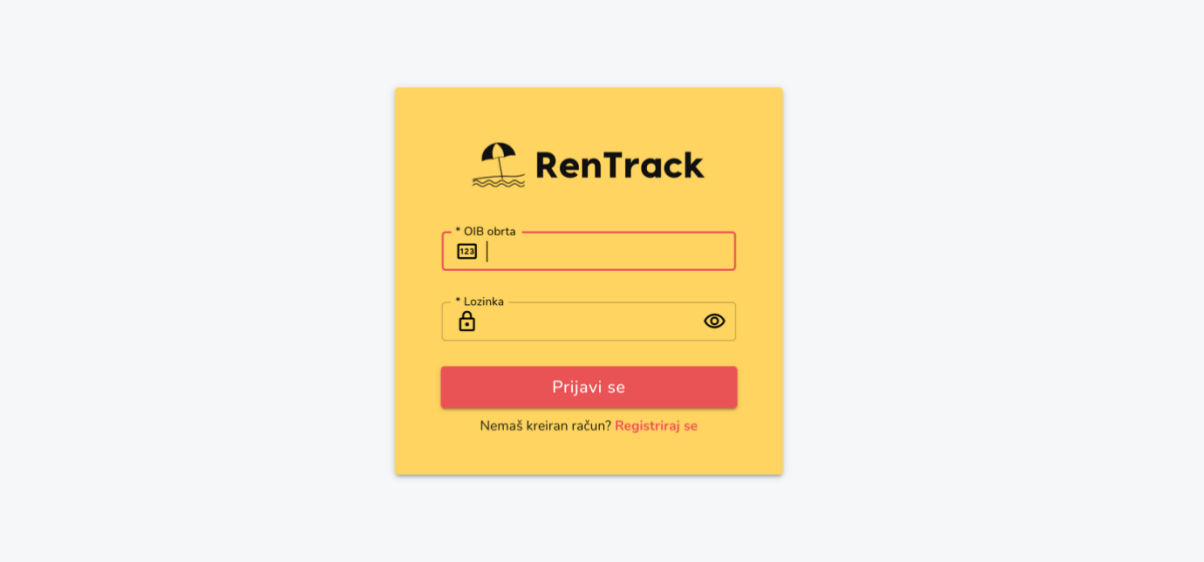
Kako bi se izbjeglo direktno spremanje unesene lozinke u bazu podataka poziva se asinkrona funkcija za heširanje iz biblioteke bcrypt (Bcrypt biblioteka za kriptiranje, 2023). Ovim putem nastoji se zaštiti osobne podatke korisnika aplikacije i tretirati ih s poštovanjem i dostojanstvom (Lazar et al., 2017). Programski kod 8 predstavlja funkciju koja vraća hash tj. niz karaktera koji se spremaju u bazu umjesto direktnog niza karaktera unesenih od djelatnika.

```
export default {
    async registerUser(userData) {
        const db = await connect();

        let doc = {
            name: userData.name,
            oib: userData.oib,
            password: await bcrypt.hash(userData.password, 8),
        };
        try {
            const result = await db.collection('users').insertOne(doc);

            if (result && result.insertedId) {
                return result.insertedId;
            }
        } catch (e) {
            if (e.code === 11000) {
                throw new Error(e);
            }
        }
    },
```

Programski kod 8. Funkcija za registraciju korisnika

3.2.3 Prijava

Da bi pristupio aplikaciji djelatnik mora proći kroz proces prijave. U slučaju da nema izrađen korisnički račun potrebno je pritisnuti "Registriraj se" dugme na dnu prozora koje vodi na stranicu za registraciju obrta. Za prijavu potrebno je imati OIB broj obrta i lozinku. Slika 3 prikazuje sučelje za prijavu.



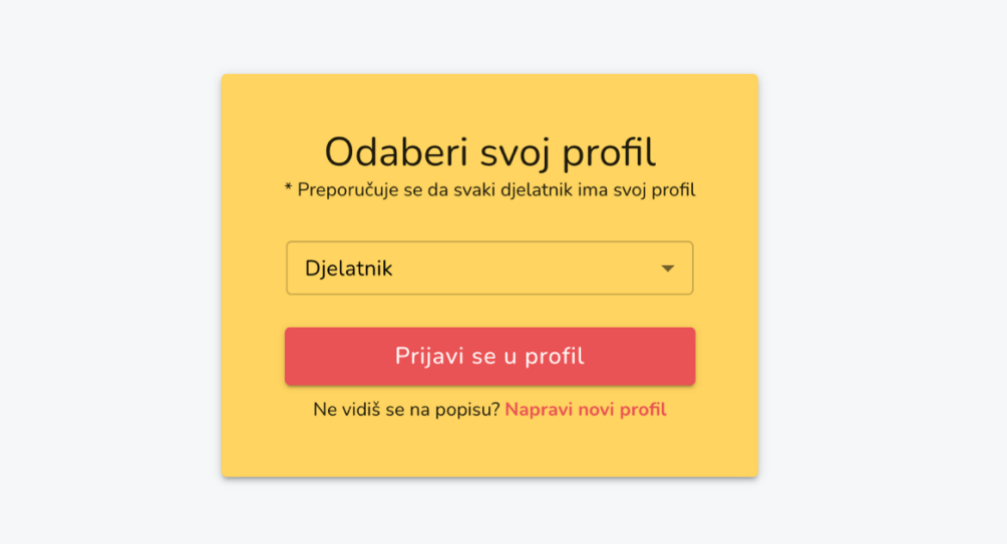
Slika 3. Stranica za prijavu obrta

Nakon što su podaci uneseni radi se zahtjev na poslužitelja koji prima objekt/dokument koji sadrži vrijednosti OIB-a i lozinke.

3.3 Odabir djelatnika

3.3.1 Dohvaćanje djelatnika

Radi organiziranijeg rasporeda s podacima svaki djelatnik bi trebao imati svoj profil. Na taj način pruža se detaljniji uvid u rad svih djelatnika. Slika 4 prikazuje odabir djelatnika i po potrebi opciju za kreiranje novog djelatnika.



Slika 4. Stranica za prijavu djelatnika

Programski kod 9 prikazuje strukturu popisa djelatnika koja se čuva se u mongo bazi podataka.

```
_id: ObjectId('6664b1b414ce448e5c3be53c')
businessId : "6664b0ff14ce448e5c3be53a"
name : "Matej"
surname : "Muller"
income : 768

_id: ObjectId('66672d98b817c52938eb0e45')
businessId : "6664b0ff14ce448e5c3be53a"
name : "Ivo"
surname : "Ivić"
income : 0

_id: ObjectId('66672da1b817c52938eb0e46')
businessId : "6664b0ff14ce448e5c3be53a"
name : "Marko"
surname : "Marić"
income : 0
```

Programski kod 9. Struktura podataka u mongo bazi

Poslužitelj je odgovoran za dohvaćanje svih profila djelatnika. Programski kod 10 opisuje proces koji pomoću GET zahtjeva spaja poslužitelja s bazom podataka i traži kolekciju users, koju zatim svrstava u polje i šalje kao odgovor.

```
app.get('/users', async (req, res) => {
    const users = await db.collection('users').find().toArray();

    if (!users) {
        res.status(400).send('No users found');
    }

    res.json(users);
});
```

Programski kod 10. REST zahtjev za dohvaćanje djelatnika

Rezultat je polje objekata gdje svaki objekt predstavlja jednog djelatnika, a svaki djelatnik ima točno pet ključ-vrijednost parova. Glavna vrijednost je "businessId" koja veže obrt sa svojim djelatnicima. Jedan djelatnik može biti vezan samo sa jednim objektom, dok jedan obrt može imati više djelatnika. Nakon što je polje djelatnika uspješno dohvaćeno ono se obrađuje na strani klijenta. Na klijentu se koristi javascript biblioteka axios za slanje GET zahtjeva na poslužitelj. Polje vrijednosti koje se dobiva kao odgovor se postavlja u "data" varijablu deklariranu kao react state. S obzirom da je to state varijabla ona ima i svoju odgovarajuću

funkciju setData, za promjenu vrijednosti, koja je potrebna da se postavi novo-dohvaćeno polje unutar data varijable. Za prikaz podataka o djelatnicima korišteni su material UI elementi: Box, FormControl, InputLabel, Select i MenuItem (Material UI komponente, 2021).

3.3.2 Izrada djelatnika

Za izradu djelatnika potrebno je unijeti ime i prezime. Nakon toga klijent šalje POST zahtjev za kreiranje novog klijenta, koji sadrži jedan dokument tj. objekt sa unesenim informacijama. Dohvaćanje korisnikovog unosa podataka vrši se pomoću state varijabli i handleChange funkcija koje na izmjenu karaktera unutar InputField komponente pozivaju state funkcije i mijenjaju na novi unos. Programski kod 11 prikazuje InputField komponentu i način na koji se funkcija poziva pri svakoj izmjeni.

```
export default function InputField({ value, type = 'text', setNewUser, field }) {
    const handleChange = (event) => {
        setNewUser((prevState) => ({
            ...prevState,
            [field]: event.target.value,
        }));
    };
```

Programski kod 11. Funkcija za ažuriranje stringa unutar state varijable

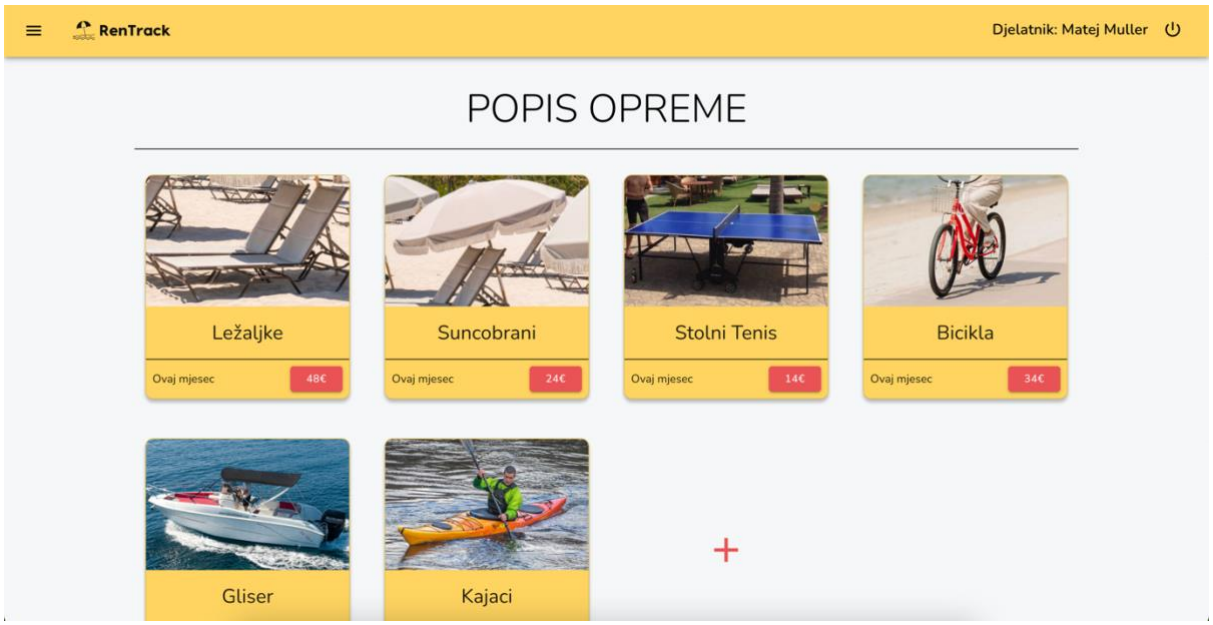
Poslužitelj obrađuje zahtjev sa klijenta u posebnoj datoteci "auth.js" unutar koje se nalazi i funkcija "createProfile" koja kao parametar prima objekt sa klijenta. Zatim se korigira finalna struktura koja će se slati u mongo bazu i izvodi se try-catch blok sa mongo insertOne funkcijom koja ubacuje taj objekt u kolekciju "profiles". Programski kod 12 sadrži cijelu createProfile funkciju.

```
async createProfile(profileData) {
    const db = await connect();

    let doc = {
        businessId: profileData.businessId,
        name: profileData.name,
        surname: profileData.surname,
        income: 0,
    };
    try {
        const result = await db.collection('profiles').insertOne(doc);

        if (result && result.insertedId) {
            return result.insertedId;
        }
    } catch (e) {
        console.log(e);
    }
},
```

Programski kod 12. CreateProfile funkcija za dodavanje novog djelatnika

3.4 Popis opreme

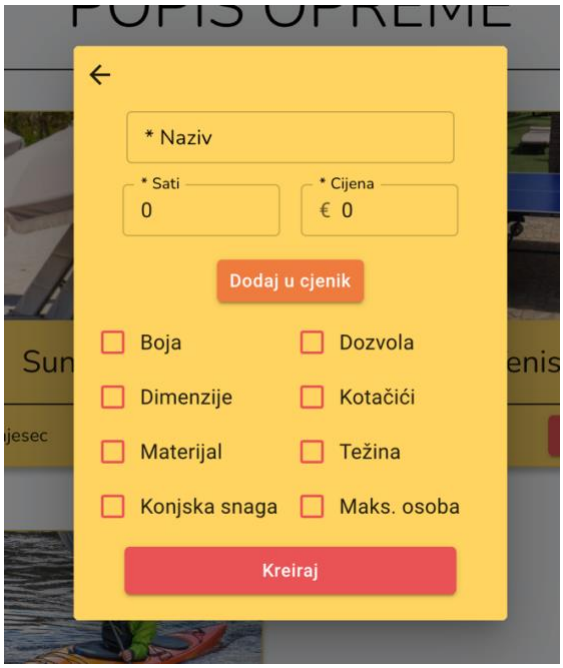
Popis opreme je glavna stranica aplikacije. Ovdje se nalazi popis svih proizvoda/usluga koje obrt pruža. Podijeljeni su u kartice sa slikom, imenom i ovomjesečnim prihodom. Na kraju stranice nalazi se dugme za dodavanje nove opreme. Slika 5 prikazuje glavnu stranicu sa popisom sve opreme obrta.



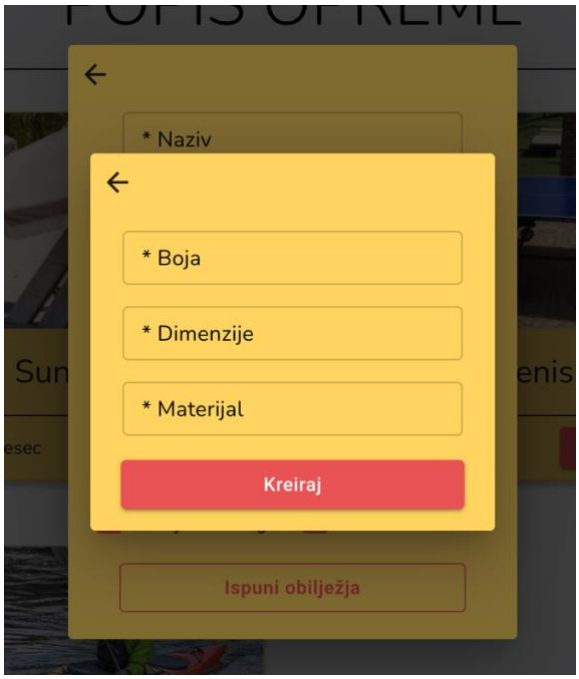
Slika 5. Stranica sa popisom sve opreme

Potrebne vrijednosti za novu opremu su: naziv, sati i cijena (cjenik) i opcionalna obilježja kao što su boja i materijal. Pri odabiru jednog obilježja "kreiraj" dugme pretvara se u "ispuni

obilježja" koje otvara prozor sa onoliko InputField komponenti koliko je obilježja djelatnik odabrao. Oba prozora, prije i poslije odabira, prikazani su na slikama 6 i 7, respektivno.



Slika 6. Dodavanje nove opreme



Slika 7. Dodatna obilježja opreme

3.4.1 Slike opreme

Svaka oprema ima svoju odgovarajuću sliku koju je moguće izmjenjivati. Slika opreme predstavlja javni link na sliku koji je pohranjen unutar imageUrl varijable kao niz karaktera u mongo bazi, kao što i prikazuje programski kod 13.

```
_id: ObjectId('6664b2ce14ce448e5c3be53d')
businessId : "6664b0ff14ce448e5c3be53a"
name : "ležaljke"
▸ prices : Object
▸ addedEquipment : Array (5)
▸ features : Object
imageUrl : "https://firebasestorage.googleapis.com/v0/b/rentrack-b7327.appspot.com…"
```

Programski kod 13. Struktura objekta opreme unutar mongo baze podataka

Za pohranjivanje slike korišten je firebase storage. Za manipulaciju slikama importane su funkcije: ref, getDownloadUrl i uploadBytes iz firebase/storage biblioteke (Firebase, 2024). Programski kod 14 prikazuje proces komunikacije sa firebase-om.

```
const handleChangeImage = async (event) => {
    const selectedImage = event.target.files[0];
    setImage(selectedImage);

    console.log('Equipment name', imageEquipmentName);
    setBackgroundColor('gray');
    setSnackbarMessage('Pričekajte trenutak...');
    setSnackbarOpen(true);

    if (selectedImage) {
        const imageRef = ref(storage, `images/${selectedImage.name}`);
        try {
            const snapshot = await uploadBytes(imageRef, selectedImage);
            console.log('Uploaded a blob or file!', snapshot);
            const url = await getDownloadURL(snapshot.ref);
            setImageUrl((prev) => [...prev, url]);
            addNewImageUrl(url, imageEquipmentName);
            console.log('URL', imageUrl);
            setBackgroundColor('forestGreen');
            setSnackbarMessage('Slika uspješno promijenjena.');
            setSnackbarOpen(true);
        } catch (error) {
            setSnackbarMessage('Greška! Pokušaj ponovno.');
            setSnackbarOpen(true);
            console.error('Error uploading image: ', error);
        }
    }
};
```

Programski kod 14. Izmjena slike opreme

Odabrana slika se sprema u "image" state. Ona se pomoću funkcije "uploadBytes" šalje na firebase i sprema na definiranu putanju, images/imeSlike u ovom slučaju. Nakon što je slika

uspješno spremljena dohvaća se njen javni lik (običan niz karaktera) koji se sprema u mongo bazu pod imageUrl ključ.

3.5 Pojedinačna oprema

3.5.1 Funkcionalnosti pojedine opreme

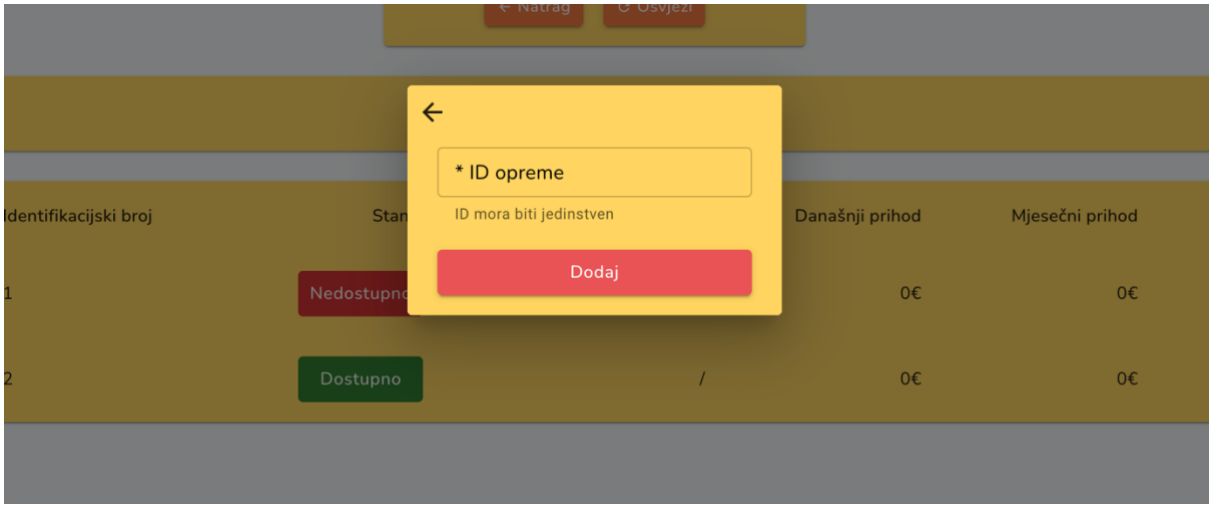
Slika 8 prikazuje osnovne funkcionalnosti koje stoje uz ime svake opreme.

Slika 8. Sučelje s funkcionalnostima pojedine opreme

Funkcionalnosti opreme su: dugme za dodavanje nove pojedinačne opreme, cjenik, zarada, dostupnost, navigacijska dugma za povratak i osvježavanje stranice.

3.5.2 Dodavanje nove pojedinačne opreme

Za dodavanje nove pojedinačne opreme potreban je jedinstveni identifikacijski broj, odnosno broj koji se pridodaje opremi i koji se već ne koristi kod te opreme. Slika 9 prikazuje način unošenja identifikacijskog broja.



Slika 9. Sučelje za upis identifikacijske oznake nove pojedinačne opreme

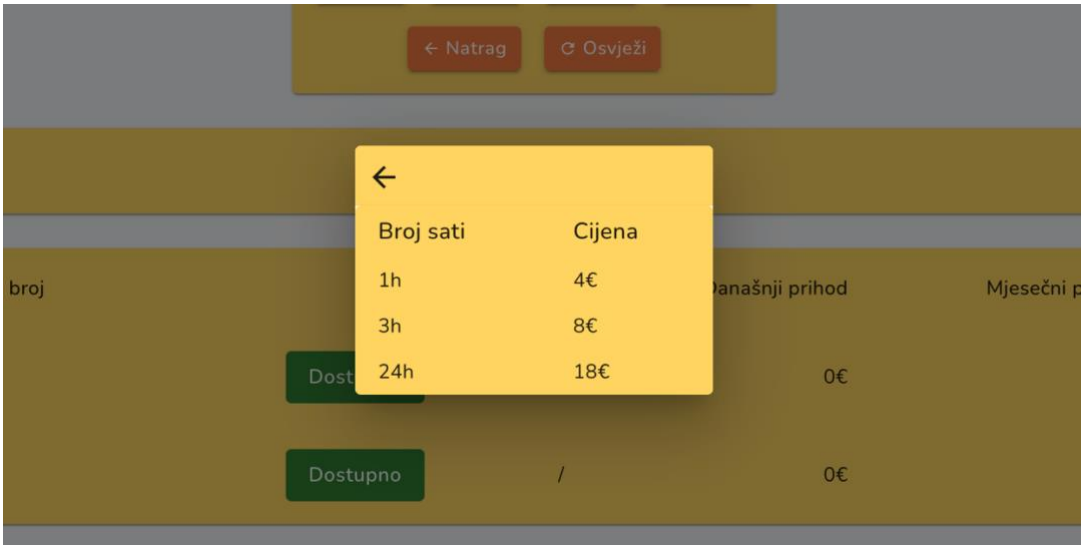
Nakon što je djelatnik unio broj, radi se provjera kroz svu opremu toga tipa i određuje se da li će oprema biti uspješno kreirana ili će javiti grešku. Cijeli proces se radi kroz asinkronu funkciju "handleAddEquipment()" koja sadrži dva try-catch bloka. Prvi blok šalje zahtjev na poslužitelja koji dohvaća svu pojedinačnu opremu prema imenu opreme. Oprema se potom sprema u "addedEquipment" varijablu (polje objekata). Posljednje vrti for petlju u kojoj provjerava da li postoji barem jedan ključ u cijelom polju objekata "addedEquipment" varijable čiji je id jednak onom koji je unio djelatnik i ako postoji baci grešku. U slučaju da ne postoji, prolazi se u drugi try-catch blok koji služi za dodavanje nove opreme u bazu pomoću POST zahtjeva na poslužitelj. Informacije koje su potrebne za dodavanje nove pojedinačne opreme su: businessId (varijabla spremljena u lokalnom prostoru preglednika), equipmentName (ime opreme za koju dodajemo novu pojedinačnu opremu) i doc (dokument ili objekt koji sadrži osnovne ključeve id, availability, endTime, profitDay, profitMonth, history). Programski kod 15 sadrži dva try-catch bloka zajedno sa upitima na poslužitelja.

```
try {
    const response = await axios.get(`${baseUrl}/equipment/${businessId}/${equipmentName}`);
    const addedEquipment = response.data[0].addedEquipment;
    for (const equipment of addedEquipment) {
        if (equipment.id === parseInt(newEquipmentId)) {
            throw new Error('Equipment already exists');
        }
    }
    try {
        const response = await axios.post(`${baseUrl}/equipment/${businessId}/${equipmentName}`, doc);
        console.log(response);
        setBackgroundColor('forestGreen');
        setSnackbarMessage('Oprema uspješno dodana.');
        setSnackbarOpen(true);
        setTimeout(() => {
            window.location.reload();
        }, 1200);
    } catch (error) {
        console.error(error);
    }
} catch (error) {
    console.error(error);
    setBackgroundColor('fireBrick');
    setSnackbarMessage('Greška! ID opreme već postoji.');
    setSnackbarOpen(true);
}
```

Programski kod 15. Provjere i zahtjevi za dodavanje nove pojedinačne opreme

3.5.3 Cjenik

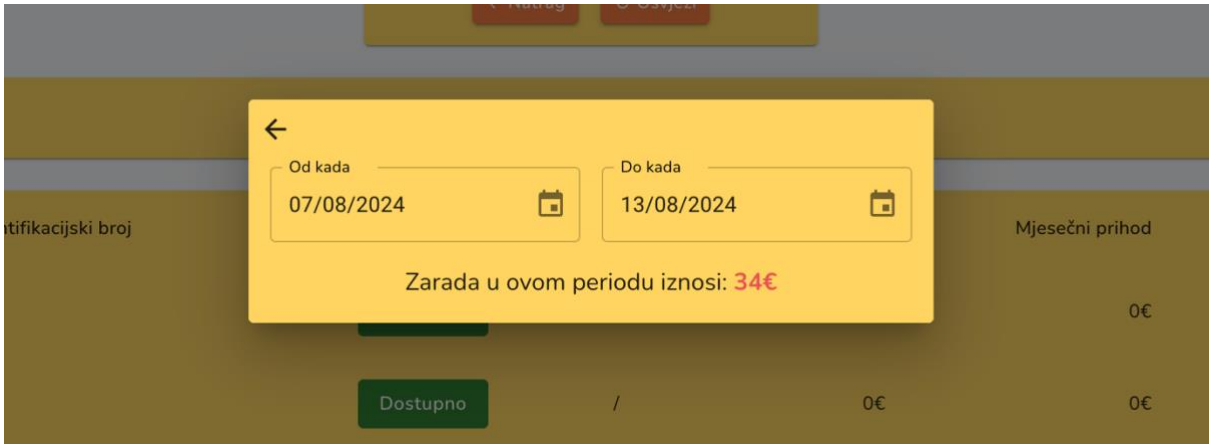
Cjenik se izrađuje pri kreiranju potpuno nove opreme. Cijena se dodaje na način da se upisuje broj sati najma i cijena. U cjenik je moguće dodavati više različitih ponuda. Svaki cjenik vidljiv je na pregledavanju pojedinačne opreme. Moguće je naknadno brisanje i dodavanje cijena/sati u postavkama. Slika 10 prikazuje opcije cjenika koje definiramo za opremu.



Slika 10. Cjenik pojedine opreme

3.5.4 Zarada

Opcijom "zarada" možemo definirati određeni raspon datuma za koji nas zanima prihod opreme. Slika 11 prikazuje zaradu u određenom vremenskom periodu.



Slika 11. Prikaz zarade u određenom vremenskom rasponu

Za izračun prihoda koristi se funkcija calculateProfit() koja je u sklopu useEffect() hook-a. UseEffect prima jednu callback funkciju koja će se izvršiti pri prvom pokretanju komponente u kojoj se nalazi, u SimpleDialog() funkciji Profit komponente. Na kraju su navedene varijable kao drugi parametar useEffect hook-a koje ako dođe do njihove izmjene rade ponovno pokretanje ove komponente. Kako bi izračunali profit potrebni su svi zabilježeni trenuci davanja opreme u najam koji se nalaze unutar history polja i koji su objektne strukture. Programski kod 16 sadrži UseEffect i calculateProfit funkciju.

```
useEffect(() => {
    const calculateProfit = async () => {
        const startingDate = dateRange[0];
        const endingDate = dateRange[1];

        try {
            const response = await axios.get(`${BASE_URL}/equipment/${businessId}/${equipmentName}`);
            const addedEquipment = response.data[0].addedEquipment;

            for (const object of addedEquipment) {
                for (const historyObject of object.history) {
                    if (historyObject.date >= startingDate && historyObject.date <= endingDate) {
                        setProfit((prevProfit) => prevProfit + parseInt(historyObject.price));
                    }
                }
            }
        } catch (error) {
            console.error(error);
        }
    };
    {
        dateRange[1] && calculateProfit();
    }
}, [dateRange, businessId, equipmentName]);
```

Programski kod 16. Izračun profita po rasponu datuma

AddedEquipment sadrži svu pojedinačnu opremu, a svaka pojedinačna oprema ima svoje history polje objekata. Iterira se kroz dvije for petlje dok se ne dođe do tih vrijednosti. Jednom kada su pronađene te vrijednosti ostatak se lako uspoređuje pomoću jednog if uvjeta s kojim se postavlja state varijable Profit koristeći njenu setProfit funkciju na novu vrijednost, ali se čuva prethodna vrijednost tako što se navede kao parametar callback funkcije. Vrijednosti su pretvarene u brojčane sa parseInt kako ne bi došlo do zbrajanja karaktera i brojeva.

3.5.5 Dostupnost

Dostupnost prikazuje koliko je pojedine opreme trenutno u najmu, kako bi u bilokojem trenutku odmah na vrhu stranice bilo vidljivo da li je ikoja oprema slobodna.

3.5.6 Obilježja opreme

Svaka oprema može imati i obilježja. Kao i cijena, obilježja se određuju pri kreiranju opreme i moguće ih je odabrati više. Obilježja su karakteristike opreme koje služe za smanjivanje nesporazuma i lakšeg određivanja koja je oprema čija u slučaju da dođe do miješanja sa tuđom. Također su i informacija u slučaju da djelatnik možda ne zna ili je zaboravio neke od specifikacija kao npr. kada bi kupac pitao koliko gliser ima konjskih snaga ili da li ga je moguće uzeti sa ili bez dozvole. Obilježja su prikazana na vrhu stranice ispod naziva opreme, kao što je i vidljivo na slici 12.

Slika 12. Obilježja glisera

3.5.7 Tablica opreme

Unutar ove tablice nalazi se popis sve pojedinačne opreme zajedno sa njihovim najbitnijim informacijama sortiranim po stupcima. Svaki stupac sadrži: identifikacijski broj, stanje najma, vrijeme isteka najma, današnji prihod, mjesečni prihod, povijest i dugme za uklanjanje. Identifikacijski broj je jedinstveni broj kojim se oprema razlikuje. Prilikom unosa postojećeg aplikacija javlja grešku. Stanje opreme je ovisno o najmu crveno ili zeleno dugme koje se mijenja na Nedostupno/Dostupno. Pri kliku na Dostupno otvara se cjenik sa svim satima i cijenama određene na cjeniku i djelatnik može odabrati jednu od ponuđenih opcija najma. Dok je oprema u najmu prikazati će se datum u stupcu "Ističe" nakon kojega će se oprema sama postaviti na "Dostupno". Današnji i mjesečni prihod su brz pregled pojedinačne opreme i koliko koja zarađuje. U slučaju da na primjer neka ležaljka ne donosi prihod možemo ju pomaknuti na drugu lokaciju ili ne kupovati nove ležaljke za mjesto gdje je prihod manji. Slika 13 prikazuje tablicu i sve njene stupce.

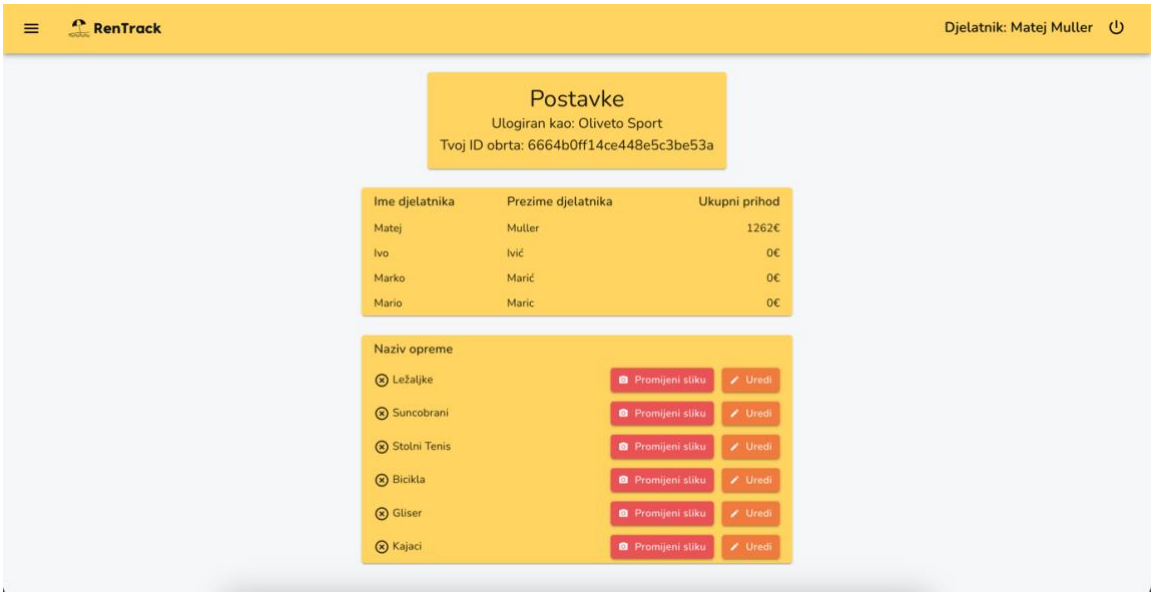
Slika 13. Tablica za praćenje opreme

Povijest opreme je dostupna na pritisak dugma sa strelicom na lijevoj strani. Otvara se jedan popis u kojem se bilježi svaki najam opreme, kada je izdana, tko ju je izdao i na koliko dugo. Svi podaci spremljeni su direktno unutar objekta u mongo bazi nakon što se najam opreme potvrdi. Slika 14 prikazuje "povijest" dio specifične opreme i informacije o toj zabilješci.

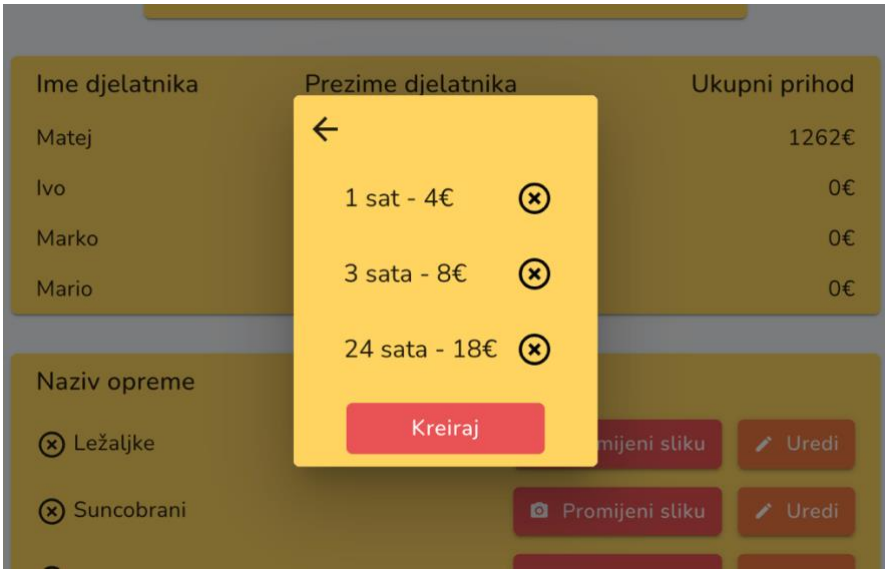
Slika 14. Tablica povijesti pojedine opreme

3.6 Postavke

Na vrhu stranice sa postavkama obrta nalazi se naziv obrta i jedinstveni ID tog obrta. Ispod toga nalaze se dva prostora koji sadrže informacije o djelatnicima i opremi. Dio za djelatnike je popis svih djelatnika obrta i njihov ukupan prihod od najma opreme. Drugi dio je popis sve opreme. Ovdje je moguće brisati opremu, ali isključivo ako niti jedna pojedinačna oprema već nije u nekom najmu. U slučaju da jest aplikacija javlja grešku i link koji vodi na stranicu te opreme. Ovdje je moguće i promijeniti sliku opreme tako što djelatnik učita novu. Aplikacija prihvaća samo datoteke tipa .jpeg, .jpg i .png. Posljednje, svaka oprema ima svoje "uredi" dugme. Ovo je mjesto za uređivanje cjenika. Vidljive su sve cijene i sati te ih je moguće uklanjati sa cjenika ili dodavati nove. Stranica sa svim postavkama obrta i način izmjene cijene sa cjenika vidljivi su na slikama 15 i 16, respektivno.



Slika 15. Sučelje stranice postavke



Slika 16. Prikaz cjenika sa mogućnosti uređivanja cijena i broja sati

3.7 Mongo baza podataka

Za spremanje podataka korištena je mongo baza (MongoDB, 2024). MongoDB je NoSQL baza podataka koja koristi dokumentno orijentiran model za spremanje podataka omogućavajući fleksibilnu i dinamičnu strukturu podataka u JSON formatu. MongoDB podržava složene upite, indeksiranje i agregacije, što olakšava rukovanje velikim i složenim dataset-ovima. Za najlakšu navigaciju korišten je i mongoDB atlas. Potrebno je izraditi novi projekt i za taj projekt kreirati jedan cluster. Cluster je skupina kolekcija gdje je svaka kolekcija skup JSON dokumenata.

3.7.1 Kreiranje novih kolekcija

Za kreiranje kolekcija MongoDB ne zahtjeva nikakav dodatni kod. Potrebno je navesti ime kolekcije i ako je dodana mongo komanda insertOne() koja ubacuje JSON dokument u neku kolekciju, mongo će automatski tu kolekciju kreirati ako ona već ne postoji. Programski kod 17 prikazuje proces korištenja insertOne mongo funkcije koja je dovoljna za kreiranje kolekcije profiles.

```
    try {
        const result = await db.collection('profiles').insertOne(doc);

        if (result && result.insertedId) {
            return result.insertedId;
        }
    } catch (e) {
        console.log(e);
    }
},
```

Programski kod 17. Dodavanje novog djelatnika u kolekciju profiles

3.7.2 Spremanje podataka u JSON formatu

Podaci su spremljeni u JSON formatu. JSON (JavaScript Object Notation) je lagan format za razmjenu podataka koji je lako čitljiv i jednostavan za shvaćanje. Osnovan je na podskupu JavaScript jezika, ali je jezično neovisan i podržan je u mnogim programskim jezicima kroz razne biblioteke i alate. Kreirana je baza podataka Rentrack i unutar nje nalaze se tri clustera: equipment, profiles i users.

3.7.3 Equipment cluster

Equipment cluster sadrži podatke o svoj opremi i pojedinačnoj opremi, kao što je i prikazano u programskom kodu 18.

```
_id: ObjectId('6664b2ce14ce448e5c3be53d')
businessId : "6664b0ff14ce448e5c3be53a"
name : "ležaljke"
▸ prices : Object
▸ addedEquipment : Array (5)
▸ features : Object
imageUrl : "https://firebasestorage.googleapis.com/v0/b/rentrack-b7327.appspot.com…"

▸
_id: ObjectId('6664b5eb3b20fef422d94a0a')
businessId : "6664b0ff14ce448e5c3be53a"
name : "suncobrani"
▸ prices : Object
▸ addedEquipment : Array (2)
▸ features : Object
imageUrl : "https://firebasestorage.googleapis.com/v0/b/rentrack-b7327.appspot.com…"
```

Programski kod 18. Struktura Equipment clustera u mongo bazi

Ovdje je izuzetno bitan businessId koji je vrijednost pohranjena za svakog djelatnika ovisno o obrtu u kojem je prijavljen. Pomoću te vrijednosti vezani su obrt, djelatnik i oprema koju daje u najam. Equipment cluster pohranjuje svu opremu od svih obrta i svih djelatnika aplikacije.

Ovdje se nalaze i istoimene opreme različitih obrta. Da bi ih se moglo razlikovati koristi se upravo taj businessId koji je potreban za svako brisanje, dodavanje ili izmjenjivanje opreme ili djelatnika. Prices je objekt koji u sebi sadrži ključeve i vrijednosti koji predstavljaju sate i cijene gdje su ključevi sati, a vrijednosti su cijene. AddedEquipment je polje objekata u kojem je pohranjena sva pojedinačna oprema. Ako obrt ima dvadeset ležaljki ovdje će biti dvadeset objekata. Svaki taj objekt ima svoje vrijednosti kao što su ID i availability koji samo poprimaju string/broj. Također svaki objekt ima i history ključ koji je još jedan objekt. History objekt sadrži timestamp informacije o najmu opreme, kada je dana u najam, tko ju je izdao i koliko je taj najam naplaćen. Programski kod 19 prikazuje detaljniju strukturu dodane opreme.

```
_id: ObjectId('6664bb3e3b20fef422d94a0c')
businessId : "6664b0ff14ce448e5c3be53a"
name : "bicikla"
▾ prices : Object
    1 : 4
    3 : 8
    24 : 18
▾ addedEquipment : Array (2)
  ▾ 0: Object
      id : 1
      availability : "available"
      endTime : "/"
      profitDay : "0"
      profitMonth : "0"
    ▾ history : Array (2)
      ▾ 0: Object
          date : "08/06/2024, 22:15:02"
          worker : "Matej Muller"
          hours : "24"
          price : "18"
      ▸ 1: Object
  ▸ 1: Object
▸ features : Object
imageUrl : "https://firebasestorage.googleapis.com/v0/b/rentrack-b7327.appspot.com…"
```

Programski kod 19. Detaljniji prikaz strukture equipment clustera

Features predstavlja objekt u kojem su pohranjena obilježja, odnosno samo ključ-vrijednost parovi poput "konjska snaga" : 100. ImageUrl je string koji se ovdje zapisuje preko POST zahtjeva kada djelatnik učita novu fotografiju koja se sprema u firebase i pri spremanju uzima URL koji sprema ga kao string. Taj string se zatim dohvaća na klijentu i koristi kao src parametar img elementa.

3.7.4 Profiles cluster

Profili predstavljaju djelatnike obrta. Svaki djelatnik ima svoj zasebni profil preko kojeg se prati njegova aktivnost na poslu i zapisuje ukupni prihod, kao što je i navedeno u programskom kodu 20.

```
_id: ObjectId('6664b1b414ce448e5c3be53c')
businessId : "6664b0ff14ce448e5c3be53a"
name : "Matej"
surname : "Muller"
income : 968

_id: ObjectId('66672d98b817c52938eb0e45')
businessId : "6664b0ff14ce448e5c3be53a"
name : "Ivo"
surname : "Ivić"
income : 0

_id: ObjectId('66672da1b817c52938eb0e46')
businessId : "6664b0ff14ce448e5c3be53a"
name : "Marko"
surname : "Marić"
income : 0
```

Programski kod 20. Struktura profiles clustera

3.7.5 Users cluster

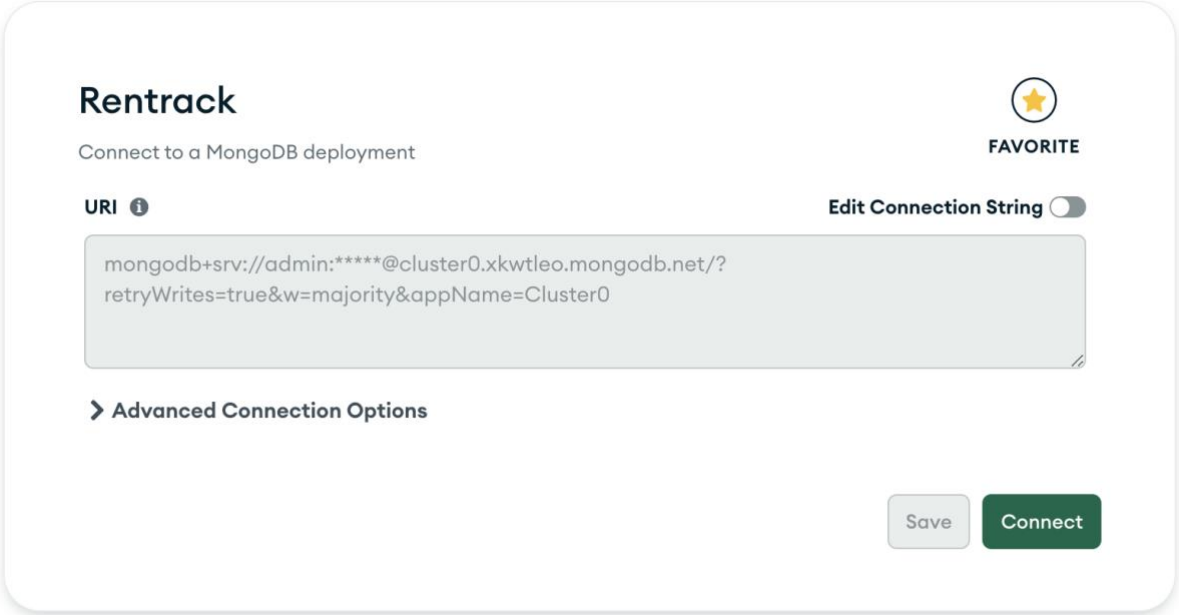
Users cluster su svi kreirani obrti. Svaki obrt ima svoj ID, naziv, oib i lozinku. Lozinka nije spremljena direktno u bazi nego je kriptirana koristeći bcrypt biblioteku tako da je spremljen samo nasumično generirani niz karaktera povećavajući razinu sigurnosti aplikacije. Programski kod 21 prikazuje kriptiranu lozinku pohranjenu u bazi podataka.

```
_id: ObjectId('6664b0ff14ce448e5c3be53a')
name : "Oliveto Sport"
oib : "46533160065"
password : "$2b$08$OxBadxXRAkjMprip6Xf1p.xPPAwcaDL4hBcKxA57rTSH1Aa8pGWyG"
```

Programski kod 21. Struktura users clustera

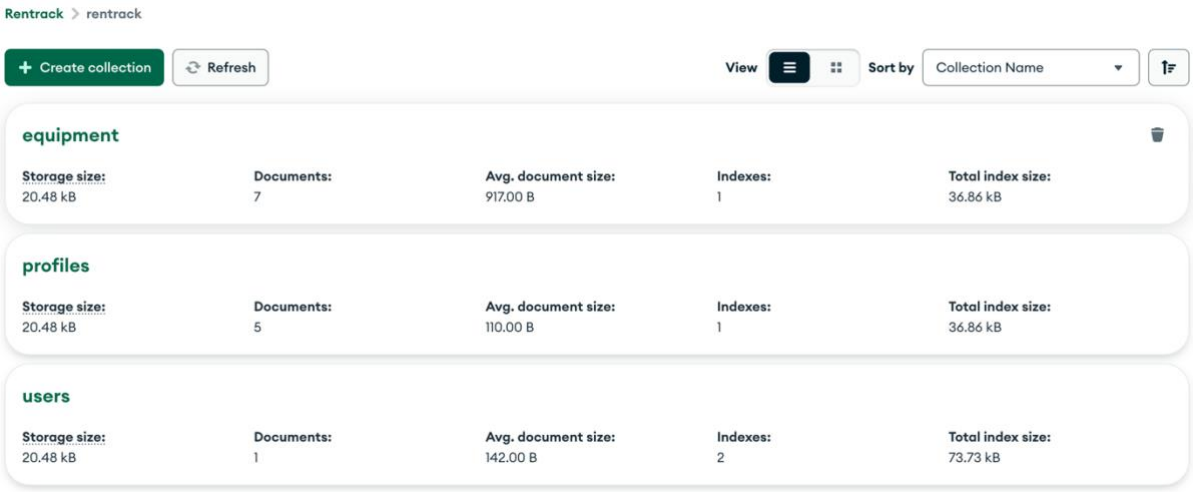
3.7.6 MongoDB Compass

MongoDB compass je aplikacija koja omogućuje lakše korištenje baze podataka. Pri pokretanju aplikacije potrebno je unijeti connection string. Connection string je poveznica sa bazom podataka dobivena pri kreiranju projekta. Slika 17 prikazuje spajanje na mongo projekt putem jedinstvene poveznice.



Slika 17. Spajanje na bazu preko MongoDB Compass desktop aplikacije

Nakon što je connection string unesen aplikacija vodi do pregleda svih baza podataka. Ovdje se može vidjeti broj dokumenata, indexa i njihove prosječne veličine, kao što je i vidljivo na slici 18.



Slika 18. Pregled kolekcija rentrack baze podataka

4. ZAKLJUČAK

Aplikacija Rentrack kreirana je s namjerom olakšavanja i unaprijeđenja poslovnog procesa rada na obrtima za najam opreme za vodene sportove. Skoro svaki oblik posla pokušava se modernizirati najčešće putem web aplikacija koje će pružati interaktivno i jednostavno izvršavanje posla. Djelatnici mogu vrlo jednostavno i pregledno koristiti aplikaciju putem mobitela zbog njenog responzivnog dizajna. Zbog toga je i aplikacija korisna za naplatu u pokretu, kao na primjer naplatu ležaljki. Uz intuitivno sučelje lako se odabire ležaljka i zabilježuje njen najam, a sam proces traje nekoliko sekundi. Aplikaciju može koristiti više djelatnika istovremeno, a vlasnik ima pregled posla u realnom vremenu preko svog mobitela ili računala. Korišten skup tehnologija MERN (mongo, express, react, node.js) je jedan od najčešćih korištenih skupova za izrade modernih web aplikacija. Ova tehnologija popularna je zbog svoje sposobnosti da omogući razvoj brzih i skalabilnih aplikacija s jednostavnom integracijom između klijenta i poslužitelja.

Uočljiva su i poneka ograničenja. Jedno od njih je skalabilnost. MongoDB je noSQL baza podataka i iz tog razloga možda nije najbolji izbor za složene transakcijske operacije koje zahtijevaju ACID (atomnost, konzistentnost, izolacija i trajnost) usklađenost. Misli se na baratanje transakcijama i konzistentnošću podataka u usporedbi s tradicionalnim relacijskim bazama podataka kao što su MySQL ili PostgreSQL. Zbog korištenja Node.js -a teže operacije mogu poprilično usporiti aplikaciju tako da bi trebalo izbjegavati intenzivne zadatke za procesor. Autorizacija i autentifikacija uvijek mogu biti strože i kompleksnije. U ovoj aplikaciji korištena je jednostavna biblioteka za kriptiranje lozinke. Pošto je korišten React koji je klijentsko-orijentirana biblioteka, može doći do povećanog inicijalnog vremena za pokretanje i lošije SEO optimizacije, što znači da će aplikacija manje iskakati u pretraživanju tražilica. Moguće je koristiti Next.js kako bi se taj problem otklonio, ali izričito dodaje na kompleksnosti aplikacije.

Web aplikacija Rentrack predstavlja inteligentan korak naprijed prema modernizaciji i optimizaciji poslovanja u sektoru najma opreme za vodene sportove. Unatoč određenim ograničenjima aplikacija pruža brojne korisne funkcionalnosti i efikasnost tijekom poslovanja koja može značajno unaprijediti izvršavanje zadataka i iskustvo djelatnika.

5. LITERATURA

1. React. (2024) Dostupno na: https://react.dev/learn [Pristupljeno: 23.03.2024]

2. Node.js. (2024) Dostupno na: https://nodejs.org/docs/latest/api/ [Pristupljeno: 15.03.2024]

3. Firebase. (2024) Dostupno na: https://firebase.google.com/docs/web/setup [Pristupljeno: 05.06.2024]

4. Material UI komponente. (2021) Dostupno na: https://mui.com/material-ui/all-components/ [Pristupljeno: 02.04.2024]

5. Tailwind. (2020) Dostupno na: https://tailwindcss.com/docs/installation [Pristupljeno: 24.03.2024]

6. Bcrypt biblioteka za kriptiranje. (2023) Dostupno na: https://www.npmjs.com/package/bcrypt [Pristupljeno: 09.06.2024]

7. MongoDB. (2024) Dostupno na: https://www.mongodb.com/docs/manual/reference/operator/ [Pristupljeno: 30.04.2024]

8. Sharp, H., Rogers, Y., & Preece, J. (2019). Interaction Design: Beyond Human-Computer Interaction (5th ed.). John Wiley & Sons, Inc.

9. Lazar, J., Feng, J. H., & Hochheiser, H. (2017). Research Methods in Human-Computer Interaction (2nd ed.). Morgan Kaufmann Publishers.

POPIS SLIKA

POPIS PROGRAMSKIH KODOVA

SAŽETAK

Aplikacija Rentrack razvijena je za olakšavanje i unaprijeđenje poslovanja obrta za najam opreme vodenih sportova, omogućujući djelatnicima brz i pregledan rad putem mobilnih i desktop uređaja zahvaljujući responzivnom dizajnu. Temelji se na MERN tehnologiji, popularnoj zbog svoje brzine i skalabilnosti, omogućujući vlasnicima pregled posla u realnom vremenu. Iako postoje ograničenja poput skalabilnosti MongoDB-a i potencijalnih performansnih problema s Node.js-om, aplikacija pruža korisne funkcionalnosti za učinkovito izvršavanje zadataka. Aplikacija Rentrack značajno doprinosi modernizaciji i optimizaciji poslovanja u području praćenja rada obrta i najma opreme.

Ključne riječi: MERN stack, najam opreme, web aplikacija, vodeni sportovi.

ABSTRACT

The Rentrack application is developed to facilitate and enhance the business operations of water sports equipment rental shops, allowing staff to work quickly and efficiently via mobile and desktop devices thanks to its responsive design. It is based on MERN technology, known for its speed and scalability, enabling owners to monitor operations in real-time. Although there are limitations such as MongoDB's scalability and potential performance issues with Node.js, the application provides useful functionalities for effective task execution. Rentrack significantly contributes to the modernization and optimization of business operations in the area of tracking shop work and equipment rental.

Keywords: MERN stack, equipment rental, web application, water sports.