Razvoj aplikacije za upravljanje zgradama na temelju stope zauzetosti prostora

Smolić, Filip

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:209134

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-24**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1485

RAZVOJ APLIKACIJE ZA UPRAVLJANJE ZGRADAMA NA TEMELJU STOPE ZAUZETOSTI PROSTORA

Filip Smolić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1485

RAZVOJ APLIKACIJE ZA UPRAVLJANJE ZGRADAMA NA TEMELJU STOPE ZAUZETOSTI PROSTORA

Filip Smolić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1485

Pristupnik:	**Filip Smolić (0036539669)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentorica:	doc. dr. sc. Anita Banjac

Zadatak: **Razvoj aplikacije za upravljanje zgradama na temelju stope zauzetosti prostora**

Opis zadatka:

Informacija o zauzetosti i načinu korištenja prostora ključna je za optimizaciju operativnih troškova rada zgrade. U sklopu rada potrebno je razviti aplikaciju za upravljanje zgradama koja korisniku pruža detaljnu analitiku korištenja prostora, proračunatu na temelju podataka iz poslovnog sustava, sustava rezervacija te sustava za registraciju prisutnosti korisnika. Poslovni sustav sadrži informacije o bolovanjima, godišnjim odmorima i poslovnim putovanjima korisnika zgrade. Grafičko sučelje aplikacije treba biti organizirano na pregledan način, s mogućnošću grafičkog odabira pojedinog kata zgrade, pojedinih tipova zona ili individualnih zona. Aplikaciju je potrebno primijeniti nad povijesnim podacima iz sustava za gospodarenje energijom neboderske zgrade Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

Rok za predaju rada: 14. lipnja 2024.

Sadržaj

Uvod 1
1. Slična programska rješenja 2
1.1. Energy Elephant 2
1.2. WatchWire 2
1.3. Usporedba postojećih i razvijenog rješenja 3
2. Funkcionalne specifikacije 4
2.1. Funkcionalni zahtjevi 4
2.2. Dijagrami korištenja 5
2.2.1. Glavna nadzorna ploča 5
2.2.2. Nadzorna ploča za statistiku zauzetosti 6
2.2.3. Nadzorna ploča za statistiku energetske potrošnje 7
2.2.4. Zaslon trenutne zauzetosti 7
2.2.5. Zaslon za ispis 8
3. Implementacija rješenja 9
3.1. Korištene tehnologije 9
3.1.1. React 9
3.1.2. ASP.NET Core 9
3.1.3. Python 9
3.2. Arhitektura aplikacije 10
3.2.1. Klijentska strana 10
3.2.2. Serverska strana 13
4. Obrada podataka 20
5. Daljnji razvoj i nadogradnje aplikacije 25
6. Korisničke upute 26
Zaključak 33
Literatura 34
Sažetak 35
Summary 36

Uvod

Klimatske promjene kao posljedica povećanja emisija stakleničkih plinova, trenutno predstavljaju jedan od vodećih problema s kojima se suočava čovječanstvo. U tom kontekstu, okretanje svijeta u smjeru održivog razvoja i energetske osviještenosti predstavlja nužan korak kako bi se smanjili negativni utjecaji na okoliš te osigurali gospodarski rast, resursi i kvalitetan život za buduće generacije.

S obzirom da operativna potrošnja energije u zgradama predstavlja oko 30% ukupne globalne potrošnje [1], metode i sustavi efikasnog upravljanja zgradama su od sve većeg značaja.

S time na umu, u sklopu ovog završnog rada, razvijena je web aplikacija za upravljanje zgradama na temelju stope zauzetosti prostora. Konkretnije, radi se o neboderskoj zgradi Fakulteta elektrotehnike i računarstva u Zagrebu (FER-a), a aplikacija je osmišljena da na jednostavan i intuitivan način omogući praćenje i analizu podataka o korištenju prostora i potrošnji energije. Generirani grafovi, liste i tablice pružaju jasan uvid u stanje korištenja zgrade te olakšavaju identifikaciju problema i pružaju solidnu bazu za odabir rješenja istog.

U idućim poglavljima rada, detaljno će se opisati proces razvoja i organizacije aplikacije kao i korištene tehnologije. Detaljnije, prvo poglavlje sadrži kratki pregled sličnih postojećih programskih rješenja i njihove usporedbe s razvijenim modelom. Drugo poglavlje se fokusira na funkcionalne specifikacije programske potpore sa navedenim funkcionalnim zahtjevima i dijagramima korištenja svake komponente. Treće poglavlje opisuje arhitekturu aplikacije, a četvrto poglavlje proces obrade podataka u aplikaciji. Peto poglavlje predstavlja prijedloge za poboljšanje i nadogradnju dok šesto poglavlje pruža jednostavne upute za korištenje aplikacije.

1. Slična programska rješenja

U sljedećim podnaslovima 1.1 i 1.2 su navedene dvije već postojeće aplikacije za obradu i analizu podataka energetske potrošnje ustanove. Iako nude još dodatnih mogućnosti poput obrade podataka potrošnje vode i emisije ugljikovog dioksida, služe kao dobra referentna točka za usporedbu korisničkih sučelja energetske potrošnje.

1.1. Energy Elephant

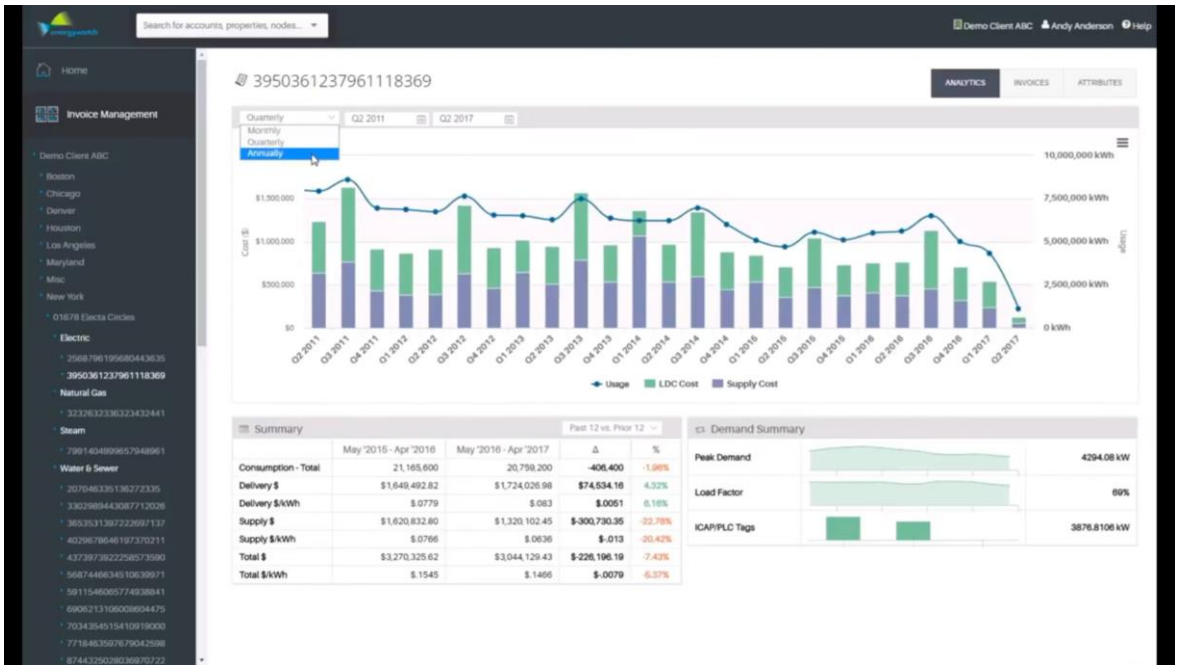
Energy Elephant je aplikacija za upravljanje energijom s primarnim ciljem optimizacije potrošnje i poboljšanjem održivosti [2]. Podatke učitava iz energetskih računa te pruža detaljne analize potrošnje energije kroz pametno korisničko sučelje (Slika 1.1). Osim energetske potrošnje, aplikacija prikuplja i analizira podatke o potrošnji vode i generiranju otpada.

Slika 1.1 Energy Elephant korisničko sučelje (preuzeto s [2]).

1.2. WatchWire

Slično kao EnergyElephant, platforma WatchWire služi kao snažan alat za optimizaciju potrošnje energije u velikim organizacijama. Pruža detaljne analize, izvještaje i alate za

efikasniju potrošnju energije [3]. Za razliku od EnergyElephanta, WatchWire može podatke prikupljati i iz energetskih mjerača, senzora i drugih uređaja. Također, ima korisničko sučelje koje omogućuje korisnicima prilagođeni pregled podataka i funkcionalnosti (Slika 1.2).



Slika 1.2 WatchWire korisničko sučelje (preuzeto s [4]).

1.3. Usporedba postojećih i razvijenog rješenja

Aplikacije navedene u prethodnom poglavlju su napravljene kao univerzalni alati za praćenje potrošnje energije na širokom spektru različitih zgrada. Iako prilagodljive za razne slučajeve, kompatibilnost s lokalnim sustavima za upravljanje neboderskom zgradom FER-a je upitna. Aplikacija za upravljanje zgradama, razvijena u sklopu ovog završnog rada, prilagođena je za rad s podatcima o zauzetosti prostorija u neboderskoj zgradi FER-a. Podatci koji se koriste preuzeti su iz FER-ovog poslovnog sustava i sustava za gospodarenje energijom zgrade FER-a razvijenog od strane Laboratorija za sustave obnovljivih izvora energije (LARES-a). Na taj način, aplikacija isporučuje točnu i detaljnu analitiku obrađenih podataka i korisnicima omogućava dublje razumijevanje energetskih karakteristika FER-ovog nebodera.

2. Funkcionalne specifikacije

Cilj aplikacije je pružiti detaljnu analitiku korištenja prostora i energetske potrošnje. Grafičko sučelje aplikacije treba biti organizirano na pregledan način, s mogućnošću odabira pojedinog kata zgrade, pojedinih tipova zona ili individualnih zona.

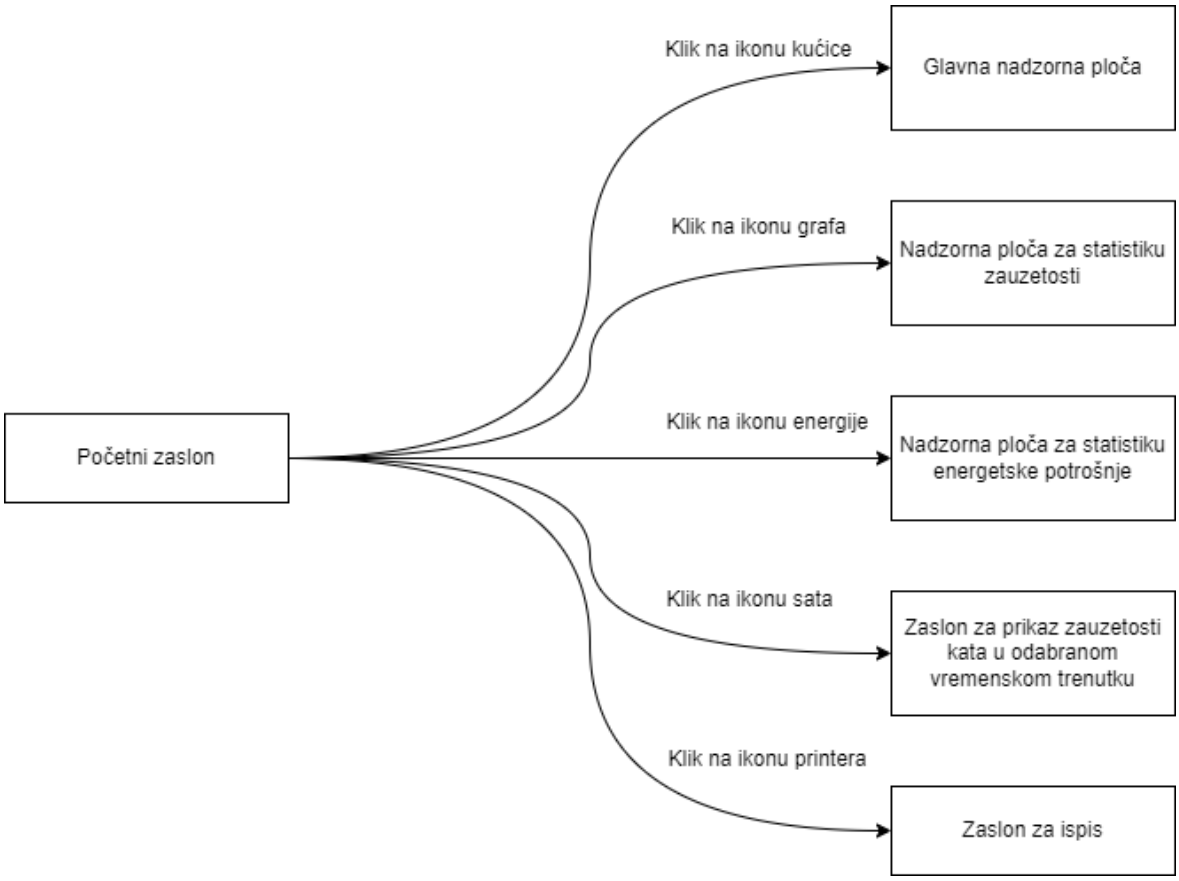
2.1. Funkcionalni zahtjevi

Aplikacija mora zadovoljiti sljedeće funkcionalne zahtjeve:

- sveobuhvatan kontrolni panel koji omogućuje korisnicima ručno odabiranje katova zgrade, zona i tipova zona, kao i specifičnih vremenskih razdoblja
- prikaz proračuna nepotrebno utrošene energije: energija koja je utrošena za grijanje/hlađenje praznog prostora za vrijeme kada u prostoru nije bilo nikoga i kada je evidentirana potrošnja energije za grijanje/hlađenje prostora
- prikaz dnevnih/tjednih/mjesečnih izvještaja i statistika o potrošnji energije u odabranoj prostoriji
- prikaz dnevnih/tjednih/mjesečnih izvještaja i statistika o potrošnji energije na cijelom katu
- glavni izbornik s prikazom ukupne godišnje nepotrebno potrošene energije, listom pet učionica koje se najmanje koriste, listom pet prostorija s najmanje/najviše nepotrebno potrošene energije, listom pet prostorija s najmanje/najviše potrošene energije općenito
- izbornik za pregled stanja zauzetosti prostorija u konkretnom odabranom vremenskom trenutku
- izbornik sa prikazom dnevnih/tjednih statistika o korištenju odabrane prostorije i korištenju prostora cijelog kata
- izbornik za ispis koji omogućuje korisniku ispis godišnjeg izvještaja i opciju sortiranja izvještaja na temelju jedne odabrane kategorije – zauzetost, potrošnja energije ili nepotrebno potrošena energija

2.2. Dijagrami korištenja

Kako bi se lakše prikazala interakcija između korisnika i sustava koriste se dijagrami korištenja aplikacije. Pri pokretanju aplikacije, otvara se početni zaslon s postavljenom glavnom nadzornom pločom, a daljnja navigacija kroz aplikaciju se ostvaruje pomoću izbornika s lijeve strane. Klikom na jednu od ikonica izbornika, otvara se željena nadzorna ploča kao što je prikazano na shemi (Slika 2.1).

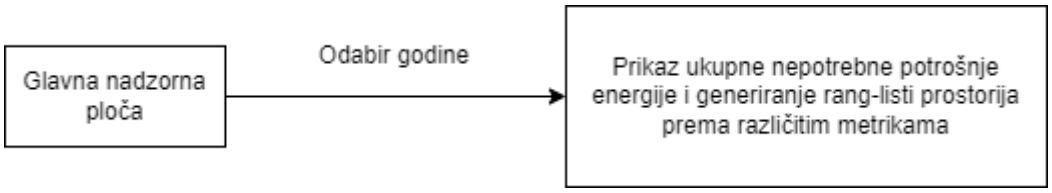


Slika 2.1 Shema navigacije sa početnog zaslona.

2.2.1. Glavna nadzorna ploča

Glavna nadzorna ploča je u izborniku predstavljena ikonom kućice. Zamišljena je da služi kao početni zaslon te sadrži najbitnije podatke za nekog tko upravlja zgradom. Omogućava korisniku unos godine, na temelju čega prikazuje sljedeće:

- horizontalan graf usporedbe ukupne nepotrebno potrošene toplinske/električne energije u cijeloj zgradi
- lista pet najmanje zauzetih učionica
- lista pet prostorija sa najmanjom/najvećom ukupnom energetskom potrošnjom
- lista pet prostorija s najmanje/najviše nepotrebno potrošene energije

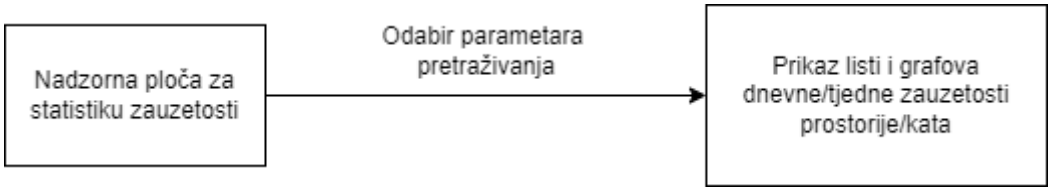


Slika 2.2 Dijagram korištenja glavne nadzorne ploče.

2.2.2. Nadzorna ploča za statistiku zauzetosti

Nadzorna ploča za statistiku zauzetosti je u izborniku predstavljena ikonom grafa. Forma za unos parametara pretraživanja omogućava izbor kata, prostorije i datuma. Nakon klika na gumb „Pretraži“, prikazuju se četiri različita grafa i dvije liste (Slika 2.3). Liste su mjesečne, a daju informacije o pet najviše i pet najmanje zauzetih prostorija na odabranom katu. Grafovi su sljedećeg oblika:

- stupčasti graf zauzetosti prostorije kroz dan
- stupčasti graf postotka zauzetosti prostorije u proteklih sedam dana
- stupčasti graf postotka zauzetosti kata kroz dan
- stupčasti graf postotka zauzetosti kata u proteklih sedam dana



Slika 2.3 Dijagram korištenja nadzorne ploče za statistiku zauzetosti.

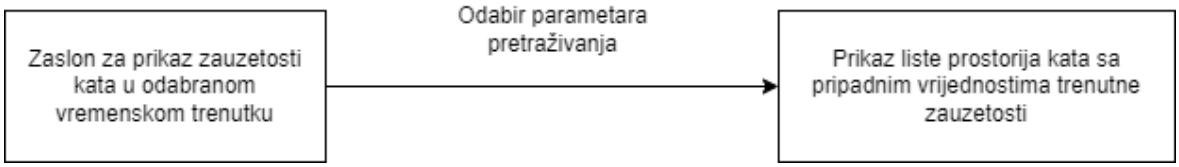
2.2.3. Nadzorna ploča za statistiku energetske potrošnje

Nadzorna ploča za statistiku energetske potrošnje je u izborniku predstavljena ikonom energije. Podatkovno je najteži dio aplikacije, u smislu da obrađuje i prikazuje najveću količinu informacija. Pretraživanje po parametrima je identično kao i kod nadzorne ploče za zauzetost, s time da korisnik ima dodatnu mogućnost navigacije između različitih prikaza statistika (Slika 2.4). Dnevna, tjedna ili mjesečna opcija u kombinaciji s opcijom prostorije ili kata, daje ukupno šest različitih setova grafova i listi koji prikazuju relevantne podatke.

Slika 2.4 Dijagram korištenja nadzorne ploče za energetsku potrošnju.

2.2.4. Zaslon trenutne zauzetosti

Zaslon za prikaz zauzetosti kata u odabranom vremenskom trenutku je u izborniku predstavljen ikonom sata. Omogućava korisniku odabir kata, datuma i vremenskog trenutka, tj. sata. Nakon postavljanja željenih parametara, prikazuje se lista svih prostorija zajedno s jednom od odgovarajućih vrijednosti zauzetosti – zauzeto, slobodno ili nema podatka. Služi kao zgodan alat za provjeru općeg stanja zauzetosti kata u nekom vremenskom trenutku. Na taj način, korisnik se može lakše orijentirati u daljnjem pretraživanju zanimljivih podataka na nekoj od preostalih nadzornih ploča (Slika 2.5).



Slika 2.5 Dijagram korištenja zaslona trenutne zauzetosti.

2.2.5. Zaslon za ispis

Zaslon za ispis je u izborniku predstavljen ikonom printera. Omogućava korisniku odabir godine i jedne od opcija sortiranja. Sortiranje može biti na temelju zauzetosti, ukupne potrošnje energije ili ukupne nepotrebno potrošene energije. Klikom na gumb „Prikaži“, generira se i prikazuje predložak godišnjeg izvještaja cijele zgrade, sortiran po jednoj od gore navedenih opcija. Ako je korisnik zadovoljan predloškom, može ga ispisati klikom na gumb „Ispis“ (Slika 2.6).

Slika 2.6 Dijagram korištenja zaslona za ispis.

3. Implementacija rješenja

3.1. Korištene tehnologije

Za izradu aplikacije korištene su sljedeće tehnologije. React knjižnica [5] za razvoj klijentske strane aplikacije, ASP.NET Core okvir [6] za izradu serverske strane i Python [7] za pisanje kratkih skripti za pred obradu .csv datoteka.

3.1.1. React

React je JavaScript knjižnica koja se koristi za izradu korisničkih sučelja web aplikacija. Glavna prednost korištenja ove tehnologije se temelji na uvođenju komponenti. Strukturira korisničko sučelje kao cjelinu sastavljenu od manjih, ponovno iskoristivih jedinica, čime se osigurava modularan i skalabilan razvoj. Te jedinice se nazivaju komponente. Dodatno, React koristi virtualni DOM. Tako ažurira samo dijelove stranice koji su se izmijenili, a ne cijelu stranicu, što znatno poboljšava performanse aplikacije.

3.1.2. ASP.NET Core

ASP.NET Core služi kao okvir za izradu serverskih aplikacija. MVC (Model-Viewer-Controller) arhitektura razdvaja odgovornosti sustava na tri dijela. Model komponenta je zadužena za logičku stranu aplikacije. Opisuje strukturu podataka i definira operacije nad istim podatcima. View dio je odgovoran za prikazivanje korisničkog sučelja i interakciju s korisnikom, a Controller obrađuje korisničke zahtjeve i služi kao posrednik između prethodne dvije komponente. Ovakav način raspodjele poslova čini aplikaciju modularnom i lakšom za testiranje. Također, okvir pruža velik broj .NET biblioteka koje olakšavaju razvoj novih funkcionalnosti.

3.1.3. Python

Od gore navedenih, Python je najmanje korištena tehnologija u sklopu razvoja ove aplikacije. Jednostavna i pregledna sintaksa, čine ovaj programski jezik idealnim za brzo i intuitivno pisanje kôda i razvoj jednostavnih programa.

3.2. Arhitektura aplikacije

3.2.1. Klijentska strana

Kako je navedeno u prijašnjem poglavlju, za izradu klijentske strane je korištena React knjižnica. Za lakše postavljanje strukture datoteka sustava, korišten je React + ASP.NET predložak unutar Visual Studio razvojne okoline. Na taj način je klijentska strana aplikacije postavljena i spremna za rad, a daljnji razvoj se svodi na stvaranje novih komponenata, njihovo uređivanje i implementacije logike iza njih.

Navedeno je ostvareno u sljedećim direktorijima:

- Components
- Styles
- Scripts

„Components“ mapa sadrži sve stvorene komponente u obliku .jsx datoteka. Ove datoteke koriste JSX sintaksu koja kombinira JavaScript i HTML te omogućuje lakše i intuitivnije pisanje kôda korisničkog sučelja. Sve komponente i njihovi opisi navedeni su u tablici (Tablica 3.1).

Tablica 3.1 Popis stvorenih komponenti.

App.jsx	Korijenska komponenta
Login.jsx	Zaslon za prijavu
Home.jsx	Početni zaslon
HomeContent.jsx	Grafovi i liste glavne nadzorne ploče
OccupancyChart.jsx	Grafovi i liste zauzetosti
EnergyDayChart.jsx	Grafovi i liste dnevne energetske potrošnje prostorije
EnergyDayChartFloor.jsx	Grafovi i liste dnevne energetske potrošnje kata
EnergyWeekChart.jsx	Grafovi i liste tjedne energetske potrošnje prostorije
EnergyWeekChartFloor.jsx	Grafovi i liste tjedne energetske potrošnje kata
EnergyMonthChart.jsx	Grafovi i liste mjesečne energetske potrošnje prostorije
EnergyMonthChartFloor.jsx	Grafovi i liste mjesečne energetske potrošnje kata
PrintTable.jsx	Tablica zaslona za ispis
Error.jsx	„Stranica nije pronađena“ komponenta

Bitna komponenta za spomenuti je korijenska App.jsx komponenta. Obavija sve ostale komponente te se sve promjene događaju unutar nje. Također, koristi react-router-dom knjižnicu za postavljanje i upravljanje rutama. Uz pomoć `createBrowserRouter` i `createRoutesFromElements` funkcija, stvara ruter koji prosljeđuje odgovarajuću komponentu s obzirom na zahtijevanu URL adresu. Isječak kôda izvedenog rutera je prikazan u nastavku (Kôd 3.1)

Kôd 3.1 Izvedba React rutera.

```
const router = createBrowserRouter(
    createRoutesFromElements(
        <Route>
            <Route path="/login" element={<Login />} />
            <Route path="/home" element={<Home /> } />
            <Route path="/" element={<Navigate to="/login" />} />
            <Route path="*" element={<Error /> } />
        </Route>
    )
)
```

Za prikaz grafova u određenim komponentama, korištena je react-chartjs-2 knjižnica [8]. Nudi velik broj različitih tipova grafova idealnih za prikaz svih vrsta podataka. U kontekstu razvijene aplikacije, uvezeni su stupčasti, linijski i prstenasti grafovi. Da bi se navedeni grafovi mogli ispravno generirati, potrebno im je proslijediti podatke u pravilnom formatu. Tu zadaću obavlja functions.js datoteka, unutar „Scripts“ mape. Sadrži `parseEnergyData` i `parseOccupancyData` funkcije koje dohvaćene podatke sa serverske strane, pretvaraju u format koji komponente grafova mogu koristiti. Primjer jednog takvog objekta, u ovom slučaju za generiranje prstenastog grafa mjesečne ukupne potrošnje energije, je prikazan na slici u nastavku (Slika 3.1).

```
▼ datasets: Array(1)
  ▼ 0:
    ▶ backgroundColor: (2) ['rgba(255, 99, 132, 0.8)', 'rgba(54, 162, 235, 0.8)']
    ▶ borderColor: (2) ['rgba(255, 99, 132, 1)', 'rgba(54, 162, 235, 1)']
      borderWidth: 1
    ▼ data: Array(2)
        0: 5813.3403
        1: 678.50354
        length: 2
      ▶ [[Prototype]]: Array(0)
    ▶ [[Prototype]]: Object
      length: 1
  ▶ [[Prototype]]: Array(0)
▶ labels: (2) ['thermal', 'electrical']
```

Slika 3.1 Primjer objekta za generiranje prstenastog grafa.

Kako bi smo stvorili konkretan graf, komponenti je potrebno proslijediti odgovarajući „data“ objekt (iz gore navedenog primjera) i „options“ objekt za postavljanje karakteristika. Primjer izvedbe prstenastog grafa je prikazan u dolje navedenom isječku kôda (Kôd 3.2). `Props.chartData[]` je lista koja sadrži sve „data“ objekte za generiranje energetskih grafova aplikacije.

Kôd 3.2 Komponenta prstenastog grafa.

```
<Doughnut options={options1} data={props.chartData[23]} />
```

3.2.2. Serverska strana

U ovom poglavlju opisat će se arhitektura i implementacijski detalji serverske strane aplikacije. Za izradu je korišten ASP.NET Core okvir sa kombinacijom MVC i RESTful API arhitekturnog stila. Uporaba okvira znatno olakšava proces izrade aplikacije. Pružaju unaprijed definiranu strukturu direktorija i datoteka zajedno s osnovnim funkcionalnostima. Na taj način je omogućen brzi početak, a daljnji razvoj uključuje prilagodbu osnovne strukture prema specifičnim potrebama aplikacije i dodavanje novih funkcionalnosti. Kôd serverske strane je pisan u programskom jeziku C#.

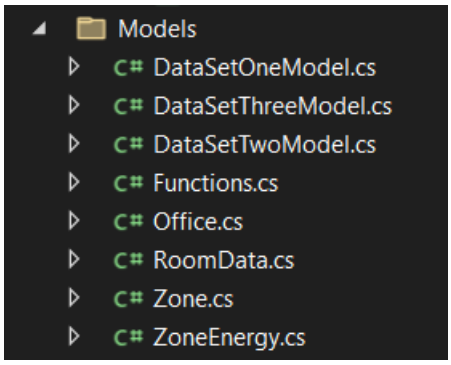
Dva navedena arhitekturna stila se kombiniraju u smislu da serverska strana implementira RESTful API koristeći MVC arhitekturu za organizaciju kôda. Točnije, kôd je podijeljen na „Model“ i „Controller“ komponente kako bi se lakše razdvojile odgovornosti aplikacije.

Funkcionalnost serverske strane je ostvarena sljedećim direktorijima:

- Models
- Controllers
- Services

„Models“ mapa sadrži klase koje opisuju strukturu podataka s kojima raspolaže. Na slici Slika 3.2 je prikazan sadržaj mape. DataSetModel klase se koriste prilikom čitanja .csv

datoteka. Ovisno o tipu ulazne .csv datoteke, svaki redak se zamjenjuje objektom odgovarajuće klase što omogućava lakši daljnji rad s uvedenim podatcima.



Slika 3.2 Struktura "Models" mape.

Kôd 3.3 DatSetOneModel klasa.

```
public class DataSetOneModel {
    [Name("batch_timestamp")]
    public string Timestamp { get; set; }
    [Name("reservations")]
    public string Reservation { get; set; }
    [Name("zone_type")]
    public string Zone { get; set; }
}
```

Preostale klase se koriste u različitim servisima i funkcijama kontrolera gdje također olakšavaju rukovanje podatcima. Od tih klasa izdvaja se klasa `Functions` koja sadrži metode za serijalizaciju podatkovnih objekata u json format. Kako bi se poslali podatci sa serverske na klijentsku stranu, nužno je da budu u prenosivom obliku. Upravo zato, se spomenute metode koriste u kontroleru, kao posljednji korak prilikom obrade zahtjeva.

„Controllers“ mapa sadrži samo jednu datoteku, MainController.cs. Obavlja ulogu kontrolera u kontekstu MVC arhitekture što znači da izlaže krajnje točke kojima pristupa klijentska strana putem HTTP zahtjeva. Općenito, klijentskoj strani je tako omogućeno izvođenje CRUD (Create, Read, Update, Delete) operacija nad resursima na poslužitelju.

Kako je aplikacija ovog završnog rada namijenjena za prikaz i pretraživanje podataka, serverska strana omogućuje samo Read operaciju preko HTTP POST zahtjeva. Sve krajnje točke, zajedno sa kratkim opisom, su navedene u nastavku.

Tablica 3.2 /home_data krajnja točka.

Dohvat podataka glavne nadzorne ploče za odabranu godinu	
URL	/home_data
HTTP	POST
Ulaz	{"Data":[{"Date":"2022-07-06"}]}
Izlaz	{"Data": {"totalWasted": {…}, "leastOccupiedClassroomsTop5": {…}, "wastedTop5": {…}, "wastedBottom5": {…}, "energyTop5": {…}, "energyBottom5": {…} }

Tablica 3.3 /graph_data krajnja točka.

Dohvat podataka zauzetosti	
URL	/graph_data
HTTP	POST
Ulaz	{"Data":[{"floor": "07", "Date": "2022-06", "Time": "00:00", "Room": "u11"}]}
Izlaz	{"Data": {"Graph1": {…}, " Graph2": {…}, " Graph3": {…}, " Graph4": {…}, {"Graph5": {…} }

Tablica 3.4 /energy_graph_data krajnja točka.

Dohvat podataka energetske potrošnje	
URL	/energy_graph_data
HTTP	POST
Ulaz	{"Data":[{"floor": "07", "Date": "2022-06", "Time": "00:00", "Room": "u11"}]}
Izlaz	{"Data":{"dayThermalCons":{…}, "dayElectricalCons":{…}, "weekThermalCons":{…}, "weekElectricalCons":{…}, "dayPieCons":{…}, "weekPieCons":{…},"dayThermalWas":{…}, "dayElectricalWas":{…}, "dayPieWas":{…}, "weekThermalWas":{…}, "weekElectricalWas":{…}, "weekPieWas":{…}, "monthThermalCons":{…}, "monthElectricalCons":{…}, "monthPieCons":{…}, "monthThermalWas":{…}, "monthElectricalWas":{…}, "monthPieWas":{…}, "dayThermalConsFloor":{…}, "dayElectricalConsFloor":{…}, "dayPieConsFloor":{…}, "dayThermalWasFloor":{…}, "dayElectricalWasFloor":{…}, "dayPieWasFloor":{…}, "weekThermalConsFloor":{…}, "weekElectricalConsFloor":{…}, "weekPieConsFloor":{…}, "weekThermalWasFloor":{…}, "weekElectricalWasFloor":{…}, "weekPieWasFloor":{…}, "monthThermalConsFloor":{…}, "monthElectricalConsFloor":{…}, "monthPieConsFloor":{…}, "monthThermalWasFloor":{…}, "monthElectricalWasFloor":{…}, "monthPieWasFloor":{…}, "dayWastedFloorList":{…}, "weekWastedFloorList":{…}, "monthWastedFloorList":{…} }

Tablica 3.5 /floor_data krajnja točka.

Dohvat podataka zauzetosti kata u konkretnom vremenskom trenutku	
URL	/floor_data
HTTP	POST
Ulaz	{"Data":[{"floor":"07","Date":"2022-07-06","Time":"00:00"}]}
Izlaz	{"Data":"[SLOBODNO, SLOBODNO, SLOBODNO, ZAUZETO]"}

Tablica 3.6 /print_data krajnja točka.

Dohvat sortiranih podataka za ispis godišnjeg izvještaja	
URL	/print_data
HTTP	POST
Ulaz	{"Data":[{"Date":"2022-07-06","Sort":"1"}]}
Izlaz	{"Data": [{ "Floor": "floor08", "Room": "classrom2", "Occupancy": 2.19, "Energy": 0.0, "WastedThermal": 0.0, "WastedElectrical": 0.0 }, … { "Floor": "floor08", "Room": "office12", "Occupancy": 97.58, "Energy": 3910.73, "WastedThermal": 0.0, "WastedElectrical": 0.0 }] }

„Services“ sadrži tri servisne klase – `CsvServices`, `StartupService`, `DataService`. Svaka od ovih klasa obavlja specifične operacije koje olakšavaju rad u ostalim dijelovima aplikacije. Ovakva organizacija omogućava lakšu nadogradnju i održavanje, a i općenito čini kôd preglednijim.

`CsvServices` klasa sadrži tri funkcije – `ReadCsv1`, `ReadCsv2`, `ReadCsv3`. Svaka od ovih funkcija se koristi samo jednom, pri pokretanju serverske strane aplikacije. Čitaju podatke iz odgovarajućih .csv datoteka i spajaju ih u tri podatkovne strukture. Svaka od struktura je organizirana na način da olakša i ubrza pretraživanje podataka prilikom obrade zahtjeva u kontroleru. Više o izvedbi navedenih podatkovnih struktura dano je u poglavlju Obrada podataka.

`StartupService` klasa ima sličnu ulogu kao i `CsvServices` u smislu da kreira podatkovnu strukturu prilagođenu za efikasniji daljnji rad. Razlika je u tome što `StartupService` klasa koristi strukture stvorene u `ReadCsv` funkcijama. Sadrži funkciju `printData` koja se također koristi samo jednom, pri pokretanju serverske strane. Grupira i računa podatke po prostorijama zgrade na godišnjoj razini i kreira jednu podatkovnu strukturu. Razlog izdvajanja ove funkcionalnosti u zaseban servis je što obrađuje veliku količinu podataka. Na ovaj način, ova vremenski zahtjevna operacija se izvršava samo jednom, a kontroler brže i jednostavnije obrađuje zahtjeve i dostavlja podatke klijentu.

Kôd 3.4 Definicija izlaznog rječnika printData funkcije.

```
Dictionary<string, List<RoomData>> printList = new Dictionary<string,
List<RoomData>>();
```

Kôd 3.5 Struktura izlaznog rječnika printData funkcije.

```
{ "2022" : { "floor8", "office1", "78.12", "3402.03", "670.65", "60.68"},
            …
           { "floor8", "office21", "77.51", "1634.45 ", "292.4", "32.15"}
```

`DataService` klasa sadrži tri polja i dvije funkcije za postavljanje tih polja. Primarna uloga ove klase je pružiti globalni pristup podatkovnim strukturama kreiranim u `ReadCsv` i `printData` funkcijama. `_occupancy_data` i `_energy_data` polja se postavljaju `setData` metodom dok se `_print_data` polje postavlja `setPrintData` metodom. Na ovaj način su grupirani svi podatci potrebni za daljnji rad aplikacije u jedan objekt, a globalna dostupnost je postignuta inicijalizacijom `DataService` objekta kao singleton-a. Singleton je oblikovni obrazac koji kreira samo jednu instancu objekta, dostupnu u svim dijelovima aplikacije. Konkretno postavljanje polja `DataService` objekta, prilikom pokretanja serverske strane aplikacije, je prikazano u tablici (Kôd 3.7).

Kôd 3.6 Postavljanje polja DataService objekta.

```
dataService.SetData(floor,csvService.ReadCsv1(files[0]),
csvService.ReadCsv2(files[1]), csvService.ReadCsv3(files[2]));
```

```
dataService.setPrintData(startupService.printData(dataService.Occupancy_data,
dataService.Energy_data));
```

Kôd 3.7 Inicijalizacija objekta kao "Singleton" prilikom konfiguracije servisa.

```
services.AddSingleton<DataService>();
```

4. Obrada podataka

Glavna uloga razvijene aplikacije je sažeti, grupirati i na jednostavan način korisniku prikazati informacije. Zbog toga je nužna kvalitetna obrada podataka. U nastavku će se opisati kompletan proces obrade, počevši od sirovih podataka pa sve do podataka spremnih za prikaz korisniku.

„CsvData“ mapa, na serverskoj strani aplikacije, sadrži pohranjene .csv datoteke grupirane po katovima. Tri su tipa .csv datoteka. Data_set_1.csv koje sadrže podatke o zauzetosti učionica u rezerviranim periodima (Tablica 4.1). Data_set_2.csv koje sadrže podatke o zauzetosti ureda u periodima od 15 minuta (

Tablica 4.2). Data_set_3.csv koje sadrže podatke o toplinskoj i električnoj snazi uređaja u periodima od jedne minute (

Tablica 4.3).

Tablica 4.1 Data_set_1.csv redak.

2022-01-01 00:05:00,"[{""reason"": ""predavanja SSDD"", ""starttime"": ""2021-12-31 10:00:00"", ""endtime"": ""2021-12-31 13:00:00""}]",classrom1

Tablica 4.2 Data_set_2.csv redak.

2022-01-01 00:00:30.734309,"{""starttime"": ""2022-01-01 00:00:00"", ""resolution"": ""15T"", ""prediction"": [1.0 … 1.0]}",office15

Tablica 4.3 Data_set_3.csv redak.

257,2022-01-01 00:01:06,0.127,0.0,office1

Prvi korak obrade podatka se obavlja python skriptama. `Csv_sort_script.py` skripta se poziva nad Data_set_2.csv datotekama i sortira retke po vremenskim zapisima uzlazno. `Csv_sort3_script.py` na isti način sortira podatke Data_set_3.csv datoteka, ali ih

još i grupira u vremenske blokove od jednog sata. Podatci o toplinskoj i električnoj snazi uređaja zadani na minutnoj bazi se zbrajaju unutar jednog sata i dijele se 60 kako bi dobili mjeru potrošene energije u tom satu (kWh). Razlog zašto ovaj korak nije implementiran na serverskoj strani aplikacije je sljedeći. Sortiranje unaprijed omogućava bržu i lakšu inicijalnu obradu podataka na serveru, a time i kraće vrijeme pokretanja. Grupiranje znatno smanjuje volumen podataka (u ovom slučaju čak i do 60 puta) čime se zauzima manje prostora za pohranu na serverskoj strani.

Sljedeći korak se odvija u ReadCsv metodama. Prilikom pokretanja serverske strane aplikacije, kôd naveden u isječku (

Kôd 4.1), prolazi kroz CsvData direktorij i sve njegove mape odnosno katove. Za svaki kat, izvršavaju se ReadCsv metode nad odgovarajućim .csv datotekama. ReadCsv1 kao ulaz prima Data_set_1.csv, izvlači relevantne podatke, grupira ih i sprema u podatkovnu strukturu prikazanu u isječku (Kôd 4.3).

Kôd 4.1 Pokretanje serverske strane aplikacije.

```
if (Directory.Exists("./CsvData")) {
    string[] directories = Directory.GetDirectories("./CsvData");
    foreach (string directory in directories)
    {
        var floor = directory.Substring(10, 7);
        string[] files = Directory.GetFiles(directory);
        dataService.SetData(floor,         csvService.ReadCsv1(files[0]),
csvService.ReadCsv2(files[1]), csvService.ReadCsv3(files[2]));
    }
}
```

Kôd 4.2 Definicija izlaznog rječnika ReadCsv1 funkcije.

```
Dictionary<string, Dictionary<string, List<Zone>>>
```

Kôd 4.3 Struktura izlaznog rječnika ReadCsv1 funkcije.

```
{ "classrom1" :
           { "2022-01-10" :
```

```
                        10:00, 1
                        11:00, 1
                        12:00, 1
            },
}
```

ReadCsv2 kao ulaz prima Data_set_2.csv, grupira podatke u vremenske blokove od jednog sata i sprema ih u podatkovnu strukturu istog tipa kao i ReadCsv1 metoda (Kôd 4.4).

Kôd 4.4 Struktura izlaznog rječnika ReadCsv2 funkcije.

```
{ "office1" :
            { "2022-01-01" :
                        00:00, 0
                        01:00, 0
                        02:00, 1
                        …
            },
            …
}
```

ReadCsv3 obavlja istu funkciju kao i prethodne dvije metode s time da kao izlaz ima malo drugačiju podatkovnu strukturu (Kôd 4.6).

Kôd 4.5 Definicija izlaznog rječnika ReadCsv3 funkcije.

```
Dictionary<string, Dictionary<string, List<ZoneEnergy>>>
```

Kôd 4.6 Struktura izlaznog rječnika ReadCsv3 funkcije.

```
{ "office1" :
            { "2022-01-01" :
                        00:00, 0.236, 0
                        01:00, 0.207, 0
                        02:00, 0.519, 0.24
                        …
```

```
            },
            …
}
```

Nakon čitanja i prikladne obrade datoteka ReadCsv funkcijama, izlazne strukture se postavljaju u DataService objekt `setData` metodom. `SetData` metoda dodaje još jedan stupanj organizacije grupiranjem podataka po katovima i spajanjem ReadCsv1 i ReadCsv2 izlaza. Definicije polja DataService objekta su dane u isječcima kôda u nastavku. Dodatno, postavlja se i printData struktura `setPrintData` funkcijom kako je navedeno u poglavlju Serverska strana.

Kôd 4.7 Definicija _occupancy_data polja DataService objekta.

```
Dictionary<string,          Dictionary<string,          Dictionary<string,
Dictionary<string, List<Zone>>>>> _occupancy_data
```

Kôd 4.8 Struktura _occupancy_data polja.

```
{ "floor8" :
            { "Data2" :
                        { "office1" :
                                    { "2022-01-01" :
                                                    00:00, 0
                                                    01:00, 0
                                                    02:00, 1
                                                    …
                                    },
                         },
            },
}
```

Kôd 4.9 Definicija _energy_data polja DataService objekta.

```
Dictionary<string,          Dictionary<string,          Dictionary<string,
List<ZoneEnergy>>>> _energy_data
```

Kôd 4.10 Struktura _energy_data polja.

```
{ "floor8" :
           { "office1" :
                       { "2022-01-01" :
                                       00:00, 0.236, 0
                                       01:00, 0.207, 0
                                       02:00, 0.519, 0.24
                                        …
                        },
           },
}
```

Time je završena inicijalna obrada podataka koje koristi serverska strana aplikacije. Kao rezultat je dobivena dobro postavljena baza za brzo i efikasno daljnje pretraživanje podataka i izvođenje računskih operacija nad istima.

5. Daljnji razvoj i nadogradnje aplikacije

U ovom poglavlju će se navesti prijedlozi za poboljšanje i implementaciju novih funkcionalnosti aplikacije u budućnosti. Kroz kontinuiranu nadogradnju, cilj je poboljšati performanse i osigurati da aplikacija zadovolji rastuće potrebe korisnika.

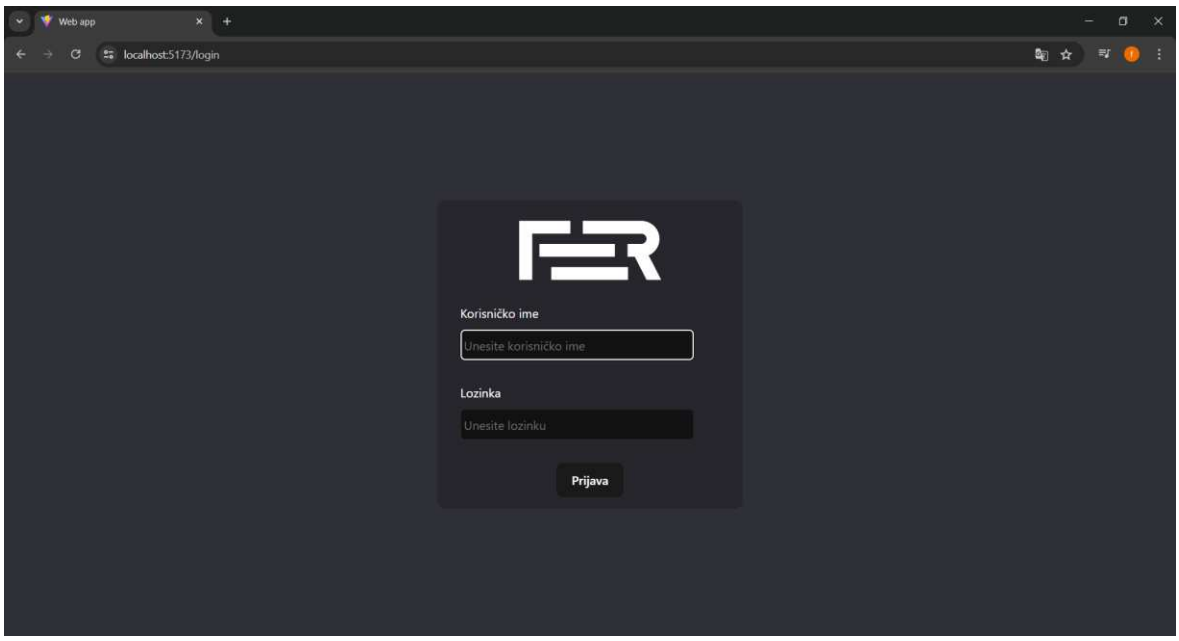
Kao jedna od mogućih nadogradnji, bilo bi osiguravanje pristupa aplikaciji samo za zaposlenike s računima na fer.hr domeni. U suradnji s tehničkim timom sveučilišta (CIP), trebalo bi osigurati pristup API-ju za autentifikaciju i implementirati autentifikacijske metode.

Također, moguće bi bilo i dodati chatbot s OpenAI API-jem za poboljšanje korisničkog iskustva. Pružanjem specifičnih uputa („prompt inženjering“), mogli bi se prilagoditi odgovori modela kako bi korisnik na jednostavan način mogao dobiti odgovore na pitanja o korištenju aplikacije i navigaciji kroz funkcionalnosti.

Od statističkih nadogradnji, moguća bi bila implementacija novih inferencijskih metoda za lakše donošenje informiranih odluka. Regresijska analiza za predviđanje buduće potrošnje energije na temelju povijesnih podataka i korelacijska analiza za provjeru povezanosti između zauzetosti prostorija i potrošnje energije.

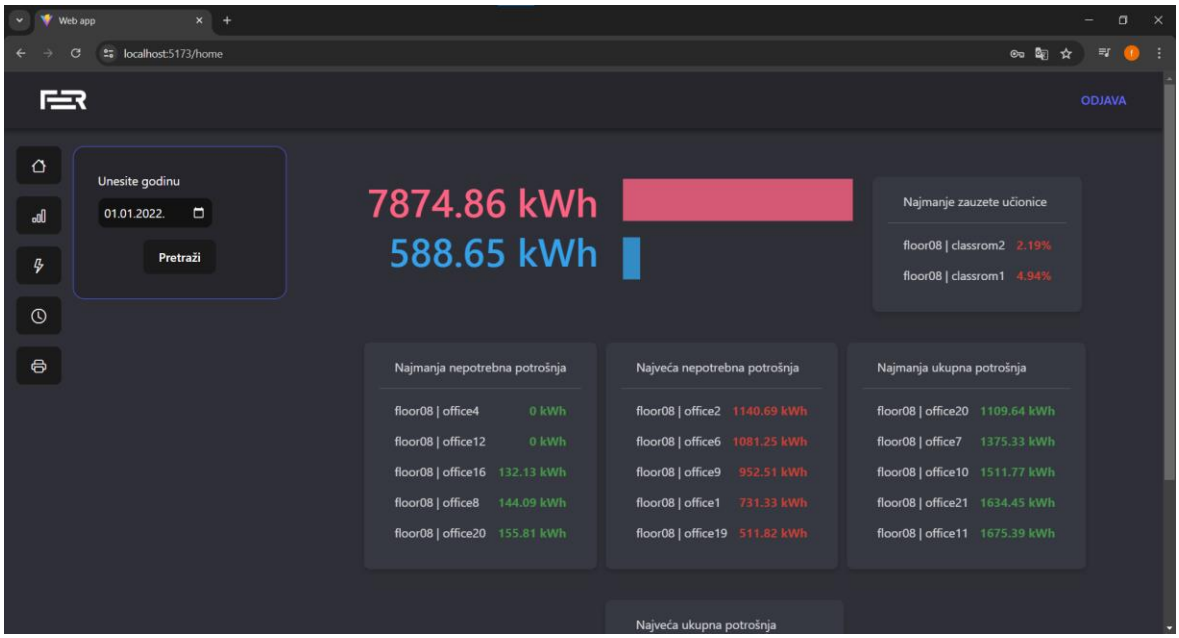
6. Korisničke upute

Pokretanjem aplikacije se otvara zaslon za prijavu (Slika 6.1). Kako autentifikacija nije ostvarena, prijava će biti uspješna za bilo koje unesene podatke.



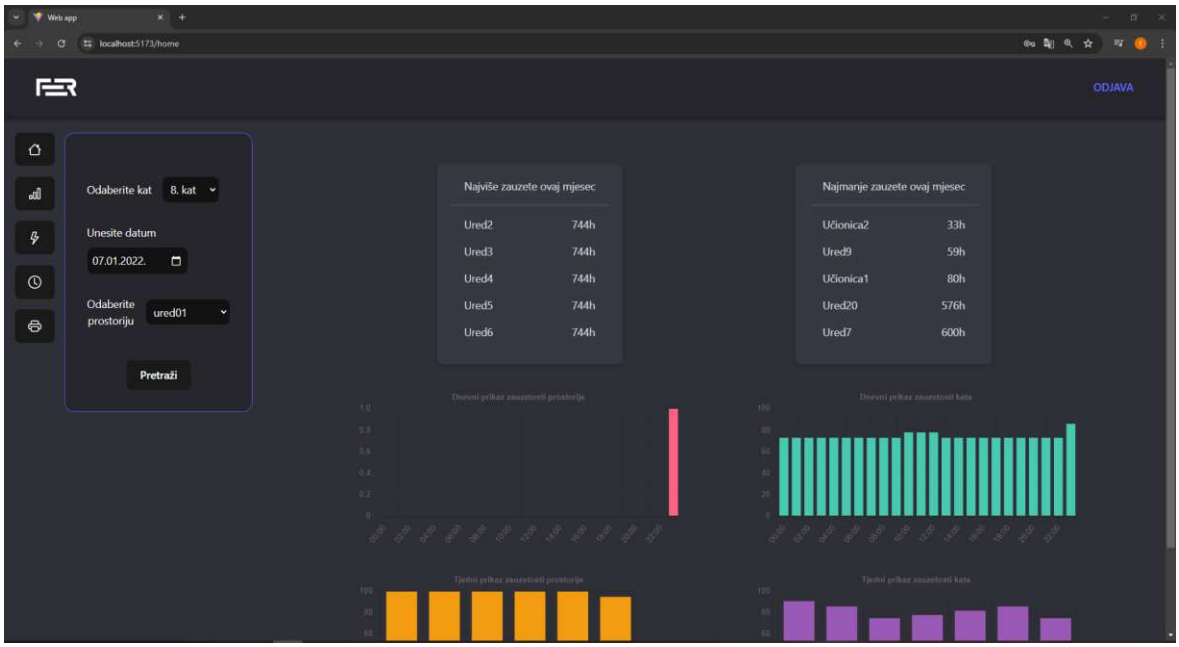
Slika 6.1 Zaslon za prijavu.

Nakon klika na gumb „Prijava“ i uspješne prijave, korisnik je usmjeren na glavnu nadzornu ploču (Slika 6.2). Glavna nadzorna ploča daje općeniti godišnji uvid u zauzetost i energetsku potrošnju zgrade. Prikazuje najznačajnije liste za nekog tko upravlja zgradom. Povratak na glavnu nadzornu ploču se ostvaruje klikom na ikonu kućice.



Slika 6.2 Glavna nadzorna ploča.

Ako korisnik želi detaljnije pregledati podatke o zauzetosti kata ili prostorija na tom katu, klik na ikonu grafu, s lijeve strane, ga vodi na nadzornu ploču zauzetosti (Slika 6.3). Unos željenih parametara i klik na gumb „Pretraži“ prikazuje podatke.

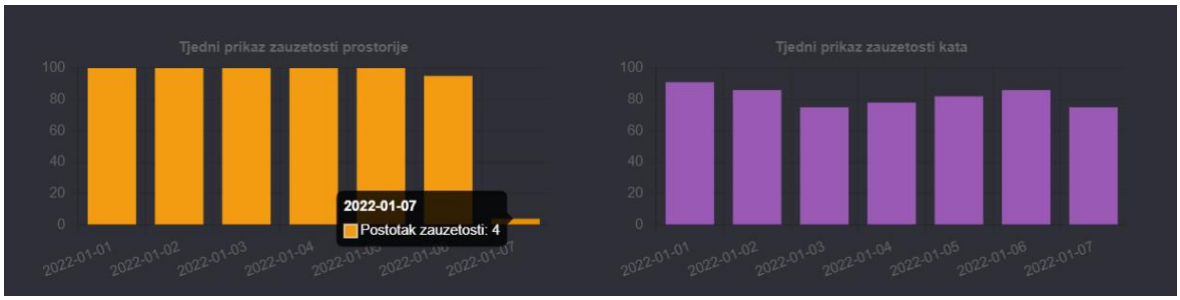


Slika 6.3 Nadzorna ploča za prikaz zauzetosti.

Pobliži prikaz grafova sa slike Slika 6.3 je dan u nastavku. Prikazani su podatci za ured1 na osmom katu zgrade dana 7.1.2022. Graf dnevne zauzetosti prostorije postavlja 0 ili 1 ovisno o zauzeću prostorije u svakom satu dana. Pošto se podatci uzimaju iz poslovnog sustava, prostorija može biti zauzeta svih 24 sata u slučaju da nisu evidentirani podatci o izostanku. Npr. ako je osoba koja koristi prostoriju odsutna radi bolesti. S druge strane, graf zauzetosti kata prikazuje postotak zauzetosti kata u svakom satu. Postotak predstavlja udio zauzetih prostorija u ukupnom broju prostorija kata; u 10:00 je bilo zauzeto 80% prostorija osmog kata.

Slika 6.4 Grafovi dnevne zauzetosti.

Graf tjedne zauzetosti prostorije prikazuje udio zauzetih sati u svakom danu posljednjih sedam dana. Graf tjedne zauzetosti kata prikazuje postotak zauzetosti odabranog kata. Kat 8 je 7.1.2022. zauzet 75%, odnosno 414 od ukupnih 552 sata (23 prostorije x 24 sata).



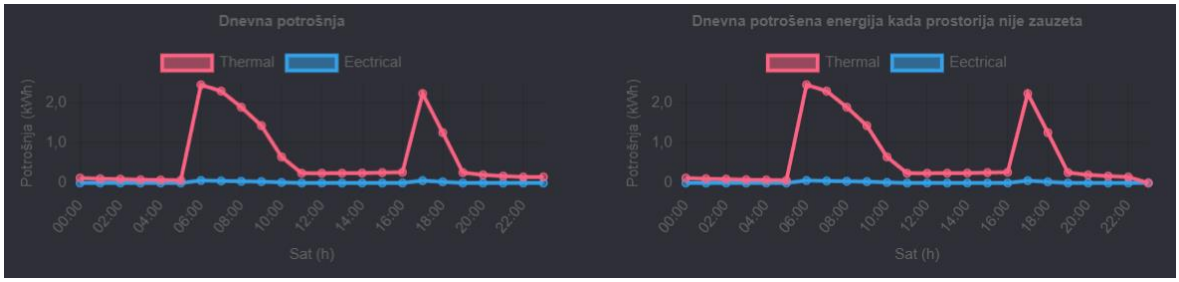
Slika 6.5 Grafovi tjedne zauzetosti.

Nadalje, moguć je prikaz podataka energetske potrošnje klikom na ikonu energije. Parametri odabrani u formi se prenose između nadzornih ploča. Ako korisnik pretražuje podatke o zauzetosti, prebacivanjem na nadzornu ploču energetske potrošnje podatci se automatski prikazuju. Dodatno, korisnik može kombinirati između dnevnog/tjednog/mjesečnog prikaza podataka za prostoriju/kat odabirom opcija ispod forme (Slika 6.6) (Slika 6.7).

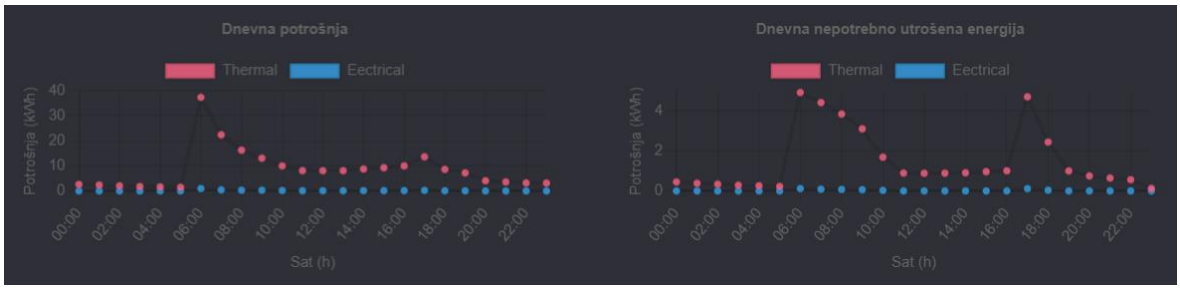
Slika 6.6 Nadzorna ploča za prikaz energetske potrošnje (dan, prostorija).

Slika 6.7 Nadzorna ploča za prikaz energetske potrošnje (dan, kat).

Pobliži prikaz grafova dnevne energetske potrošnje prostorije (Slika 6.8) i kata (Slika 6.9) je dan u nastavku. Crvena linija grafa predstavlja apsolutnu vrijednost potrošnje toplinske energije u kWh, a plava linija vrijednost električne potrošnje u svakom satu dana u kWh. U slučaju kata, vrijednosti se računaju istim principom te se zbrajaju za svaku prostoriju kata.

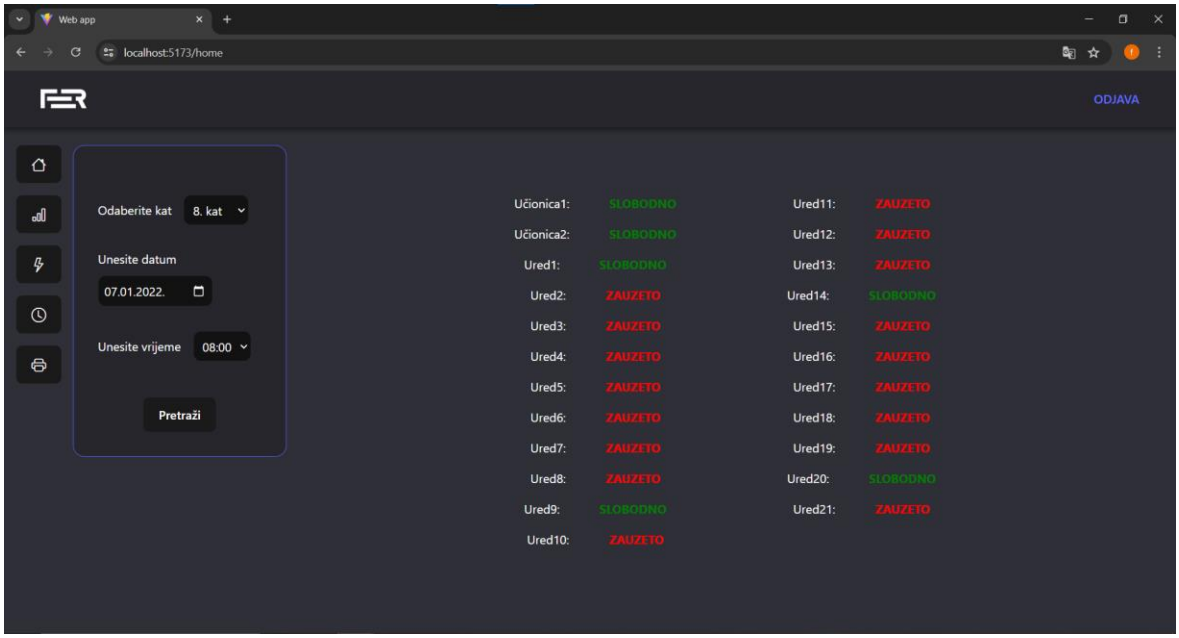


Slika 6.8 Grafovi dnevne energetske potrošnje prostorije.



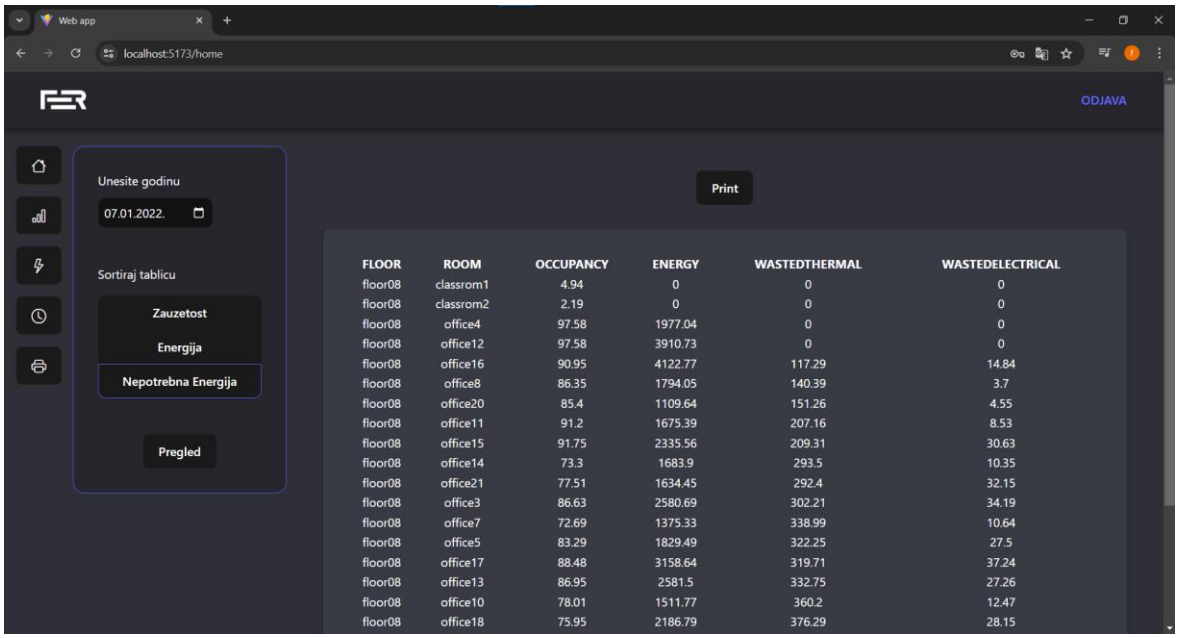
Slika 6.9 Grafovi dnevne energetske potrošnje kata.

Prikaz podataka zauzetosti u konkretnom vremenskom trenutku se ostvaruje klikom na ikonu sata. Odabirom parametra „vrijeme“, korisniku se prikažu podatci o tome koja je prostorija zauzeta, a koja slobodna u tom trenutku. Podatci su grupirani u vremenske blokove od jednog sata tako da se prostorija smatra slobodnom samo ako nije zauzeta u niti jednom trenutku sata. Kako je već navedeno, prostorija može biti prikazana kao zauzeta u slučaju da u sustavu nije evidentiran izostanak. Primjer detaljnog prikaza zauzeća za prostorije na 8. katu u vremenskom trenutku 08:00 dan je na slici Slika 6.10.



Slika 6.10 Zaslon za prikaz vremenske zauzetosti u konkretnom trenutku.

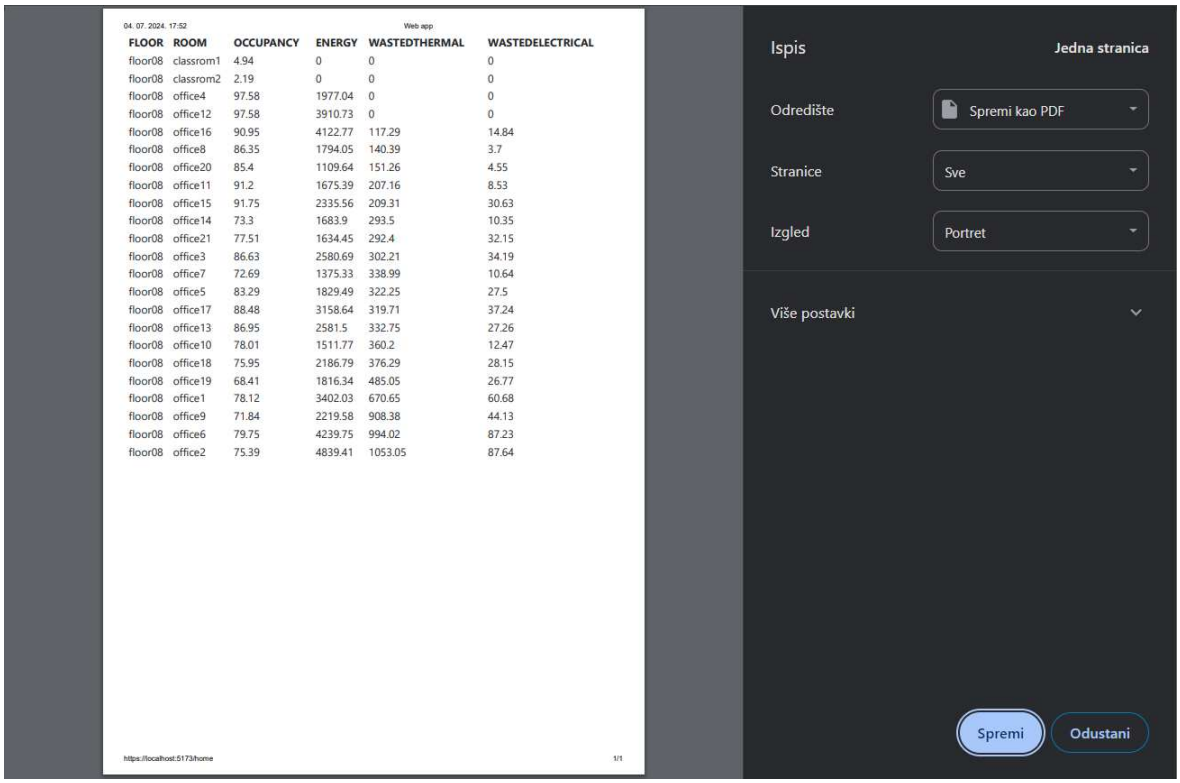
Ispis godišnjeg izvještaja se obavlja na zaslonu za ispis koji je predstavljen ikonom printera. Osim odabira željene godine izvještaja, moguć je i odabir jedne od opcija sortiranja. Klikom na „Zauzetost“, prostorije se sortiraju silazno po postotku zauzetosti u toj godini. Klik na „Energija“, sortira prostorije silazno po količini potrošene energije u toj godini. Klik na „Nepotrebna Energija“, sortira prostorije također silazno, ali po zbroju nepotrebno potrošene toplinske i električne energije u toj godini. Gumb „Prikaži“ daje pregled sortiranog godišnjeg izvještaja i postavlja gumb „Print“ (Slika 6.11). Ako je korisnik nakon pregleda izvještaja zadovoljan i želi ga ispisati, klik na gumb „Print“ otvara prozor za ispis (Slika 6.12).



FLOOR	ROOM	OCCUPANCY	ENERGY	WASTEDTHERMAL	WASTEDELECTRICAL
floor08	classrom1	4.94	0	0	0
floor08	classrom2	2.19	0	0	0
floor08	office4	97.58	1977.04	0	0
floor08	office12	97.58	3910.73	0	0
floor08	office16	90.95	4122.77	117.29	14.84
floor08	office8	86.35	1794.05	140.39	3.7
floor08	office20	85.4	1109.64	151.26	4.55
floor08	office11	91.2	1675.39	207.16	8.53
floor08	office15	91.75	2335.56	209.31	30.63
floor08	office14	73.3	1683.9	293.5	10.35
floor08	office21	77.51	1634.45	292.4	32.15
floor08	office3	86.63	2580.69	302.21	34.19
floor08	office7	72.69	1375.33	338.99	10.64
floor08	office5	83.29	1829.49	322.25	27.5
floor08	office17	88.48	3158.64	319.71	37.24
floor08	office13	86.95	2581.5	332.75	27.26
floor08	office10	78.01	1511.77	360.2	12.47
floor08	office18	75.95	2186.79	376.29	28.15

Slika 6.11 Zaslon za ispis (sortirano po nepotrebnoj energetskoj potrošnji).

„OCCUPANCY“ stupac tablice sa slike Slika 6.11 označava postotak zauzetosti prostorije u cijeloj godini. Učionica 1, na katu 8, je zauzeta 4.94% godine odnosno 432 sata od ukupnih 8 760 sati u 2022. godini. Pritom se prostorija smatra zauzetom samo ako je ona rezervirana preko službenog sustava FER-a za rezervaciju termina.



FLOOR	ROOM	OCCUPANCY	ENERGY	WASTEDTHERMAL	WASTEDELECTRICAL
floor08	classrom1	4.94	0	0	0
floor08	classrom2	2.19	0	0	0
floor08	office4	97.58	1977.04	0	0
floor08	office12	97.58	3910.73	0	0
floor08	office16	90.95	4122.77	117.29	14.84
floor08	office8	86.35	1794.05	140.39	3.7
floor08	office20	85.4	1109.64	151.26	4.55
floor08	office11	91.2	1675.39	207.16	8.53
floor08	office15	91.75	2335.56	209.31	30.63
floor08	office14	73.3	1683.9	293.5	10.35
floor08	office21	77.51	1634.45	292.4	32.15
floor08	office3	86.63	2580.69	302.21	34.19
floor08	office7	72.69	1375.33	338.99	10.64
floor08	office5	83.29	1829.49	322.25	27.5
floor08	office17	88.48	3158.64	319.71	37.24
floor08	office13	86.95	2581.5	332.75	27.26
floor08	office10	78.01	1511.77	360.2	12.47
floor08	office18	75.95	2186.79	376.29	28.15
floor08	office19	68.41	1816.34	485.05	26.77
floor08	office1	78.12	3402.03	670.65	60.68
floor08	office9	71.84	2219.58	908.38	44.13
floor08	office6	79.75	4239.75	994.02	87.23
floor08	office2	75.39	4839.41	1053.05	87.64

Slika 6.12 Ispis godišnjeg izvještaja (sortirano po nepotrebnoj energetskoj potrošnji).

Zaključak

U sklopu ovog završnog rada, razvijena je aplikacija za upravljanje zgradama na temelju stope zauzetosti prostora. Iako već postoje neka programska rješenja, ova aplikacija je napravljena da bude kompatibilna sa sustavima FER-ove neboderske zgrade.

React knjižnica se pokazala ključnom za izgradnju optimiziranog korisničkog sučelja s velikim brojem funkcionalnosti. ASP.NET Core okvir je znatno ubrzao proces izrade serverske strane i osigurao olakšan daljnji razvoj aplikacije. Nadalje, jasna separacija klijentske od serverske strane, omogućuje razvoj novih klijentskih aplikacija koje bi koristile podatke razvijenih krajnjih točaka.

Kao rezultat, korisnik dobiva pouzdan alat koji mu pruža uvid u trendove zauzetosti prostora i energetske karakteristike zgrade. Aplikacija čini dobru podlogu za daljnje djelovanje s ciljem efikasnijeg korištenja energije i boljeg planiranja prostornih resursa.

Implementacija chatbota, sigurnog pristupa aplikaciji i uvođenje dodatnih statističkih metoda predstavlja jedan od mogućih smjerova daljnjeg razvoja aplikacije.

Literatura

[1] International Energy Agency, https://www.iea.org/energy-system/buildings, 29. svibnja 2024.

[2] Energy Elephant, https://energyelephant.com/, 7. lipnja 2024.

[3] Watchwire, https://watchwire.ai/enterprise/, 7. lipnja 2024.

[4] GetApp, https://www.getapp.com/industries-software/a/watchwire/, 7. lipnja 2024.

[5] React, https://react.dev/reference/react, 12. lipnja 2024.

[6] ASP.NET Core, https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core, 12. lipnja 2024.

[7] W3schools, https://www.w3schools.com/python/python_intro.asp, 12.lipnja 2024.

[8] NPM , https://www.npmjs.com/package/react-chartjs-2, 14.lipnja 2024.

Sažetak

U sklopu ovog završnog rada, razvijena je aplikacija za upravljanje zgradama na temelju stope zauzetosti prostora. Namijenjena je da posluži kao koristan alat u analizi podataka neboderske zgrade Fakulteta elektrotehnike i računarstva (FER-a) i pomogne u identifikaciji potencijalnih problema korištenju prostora. Podatci o zauzetosti prikupljeni su iz poslovnog sustava FER-a, dok su podatci o potrošnji energije preuzeti iz sustava za gospodarenje energijom razvijenog u okviru FER-ovog Laboratorija za sustave obnovljivih izvora energije (LARES-a). Klijentska strana aplikacije je izvedena pomoću React knjižnice i nudi jednostavno korisničko sučelje za navigaciju između različitih nadzornih ploča. Serverska strana obrađuje i prosljeđuje podatke, a implementirana je korištenjem ASP.NET Core okvira.

Ključne riječi: pametno upravljanje zgradama; upravljanje prostornim resursima u zgradama; obrada podataka; web aplikacija; korisničko sučelje; gospodarenje energijom u zgradama; energetska učinkovitost

Summary

As a part of this paper, an application for building management based on occupancy rates has been developed. It is intended to serve as a useful tool for data analysis of the high-rise building of the Faculty of Electrical Engineering and Computing (FER) and to help identify potential issues with room usage. Occupancy data has been collected from FER's business system, while energy consumption data has been obtained from the energy management system developed by FER's Laboratory for Renewable Energy Systems (LARES). The client side of the application is built using the React library and offers a simple user interface for navigating between different dashboards. The server side processes and forwards data, and is implemented using the ASP.NET Core framework.

Keywords: smart building management; space management; data processing; web application; user interface; energy management in buildings; energy efficiency