Cloud aplikacija E-dimnjačar

Dajković, Antun Vedran

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI PREDIPLOMSKI/DIPLOMSKI STUDIJ RAČUNARSTVO

Antun Vedran Dajković, 0068225115

Cloud aplikacija E - dimljačar

Završni rad

Čakovec, rujan 2024.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

STRUČNI STUDIJ RAČUNARSTVO

Antun Vedran Dajković, **0068225115**

Cloud aplikacija E - dimljačar

Cloud application E – chimney sweep

Završni rad

Mentor:

dr. sc. Bruno Trstenjak

Čakovec, rujan 2024.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU

PRIJAVA TEME I OBRANE ZAVRŠNOG/DIPLOMSKOG RADA

Stručni prijediplomski studij:

☐ Računarstvo ☐ Održivi razvoj ☐ Menadžment turizma i sporta

Stručni diplomski studij Menadžment turizma i sporta: ☐

Pristupnik: Antun Vedran Dajković, JMBAG: 0068225115
(ime i prezime)

Kolegij: Složeni aplikacijski programi
(na kojem se piše rad)

Mentor: dr.sc. Bruno Trstenjak, prof. struč. stud.
(ime i prezime, zvanje)

Naslov rada: Oblak aplikacija E - dimljačar

Naslov rada na engleskom jeziku: Cloud application E – chimney sweep

Članovi povjerenstva: 1. dr.sc. Sanja Brekalo, prof. struč. stud., predsjednik
(ime i prezime, zvanje)
2. Jurica Trstenjak, v. pred., član
(ime i prezime, zvanje)
3. dr. sc. Bruno Trstenjak, prof. struč. stud., mentor
(ime i prezime, zvanje)
4. Marija Miščančuk, v. pred., zamjenski član
(ime i prezime, zvanje)

Broj zadatka: 2022-RAČ-I-65

Kratki opis zadatka: Izrada web jednostranične aplikacije u oblaku kojom će se omogućiti vođenje podataka o radu dimnjačarske službe na području grada. Aplikacija će omogućiti vođenje podatka o građanima i obavljenom čišćenju prema planiranom kalendaru. U aplikaciji će biti formiran kalendar i izrada plana pružanja usluga prema korisnicima, ulicama ili naselju. Aplikacija će za svoj rad koristiti relacijsku bazu (kao što je MySQL) pogodnu za rad u oblak okruženju. Aplikacija će biti izrađena u Spring Boot tehnologiji bazirano na Java objektnom programskom jeziku. Za ostvarivanje dinamičnosti web aplikacije koristi će se AngularJS biblioteka. U aplikaciji će bit implementirani administrativni dio koji će omogućiti uređivanje osnovnih podataka. Sadržaj podataka u bazi i razrada tablica bit će dogovoreno naknadno s mentorom.

Datum: 18.09.2024 **Potpis mentora:** dr.sc. Bruno Trstenjak, prof. struč. stud.

SAŽETAK

U suvremenom poslovnom okruženju, informacijski sustavi imaju bitnu ulogu u poslovanju, omogućujući poduzećima učinkovitiju raspodjelu resursa i bolju kontrolu nad poslovnim aktivnostima. U skaldu s tim, ovaj završni rad bavi se razvojem web aplikacije za potrebe dimljčara. Za razvoj aplikacije korišten je JHipster, generator aplikacija koji integrira moderne tehnologije poput Angulara i Spring Boota. Cilj aplikacije je omogućiti praćenje i upravljanje radnim nalozima, pružajući platformu za kreiranje, organiziranje i praćenje radnih zadataka dimljčarskog poduzeća.

Rad sustavno analizira i kategorizira korištene tehnologije, ovakva strukturirana analiza omogućava jasno razumijevanje uloge svake tehnologije u razvoju aplikacije. Angular je odabran za razvoj dinamičnog i responzivnog korisničkog sučelja, dok Spring Boot osigurava robusnu logiku. MySQL je izabran za upravljanje bazom podataka zbog svoje pouzdanosti i skalabilnosti, a Maven za upravljanje projektom i ovisnostima, olakšavajući razvojni proces. Svaka od ovih tehnologija odabrana je s ciljem postizanja optimalne funkcionalnosti, sigurnosti i skalabilnosti aplikacije.

Struktura rada obuhvaća uvod, ciljeve, pregled korištenih tehnologija, alate korištene za izradu aplikacije, tehničku strukturu i tok korištenja aplikacije, opis baze podataka, te poglavlja o testiranju i sigurnosti aplikacije.

Zaključno, rad demonstrira kako integracija modernih tehnologija, pažljivo planiranje i jasno definirana arhitektura mogu rezultirati razvojem skalabilne i pouzdane web aplikacije koja zadovoljava poslovne zahtjeve dimljačarskog poduzeća, istovremeno osiguravajući visoku razinu sigurnosti i dobre performanse.

Ključne riječi: JHipster, Angular, Spring Boot, MySQL, Maven, e-diljačar, web aplikacija

Abstract

In the modern business environment, information systems play an essential role in business, enabling companies to allocate resources more efficiently and have better control over business activities. In accordance with this, this final paper deals with the development of a web application for the needs of chimney sweeps. To create the foundation, JHipster, an application generator that integrates modern technologies such as Angular and Spring Boot, was used. The goal of the app is to enable the tracking and management of work orders, providing a platform for creating, organizing, and tracking the work tasks of a chimney sweeping company.

The paper systematically analyzes and categorizes the technologies used, such a structured analysis allows a clear understanding of the role of each technology in the development of the application. Angular was chosen to develop a dynamic and responsive user interface, while Spring Boot ensures robust logic. MySQL was chosen for database management because of its reliability and scalability, and Maven for project and dependency management, facilitating the development process. Each of these technologies has been selected with the aim of achieving optimal functionality, security and scalability of the application.

The structure of the paper includes an introduction, goals, an overview of the technologies used, the tools used to create the application, the technical structure and flow of application use, a description of the database, and chapters on testing and application security.

In conclusion, the paper demonstrates how the integration of modern technologies, careful planning and clearly defined architecture can result in the development of a scalable and reliable web application that meets business requirements of chimney sweeps, while ensuring a high level of security and good performance.

Keywords: JHipster, Angular, Spring Boot, MySQL, Maven, chimney sweep, web application

Sadržaj

1. UVOD

U današnjem dinamičnom poslovnom okruženju, digitalna transformacija ključna je za održavanje konkurentske prednosti. Poduzeća i javne službe sve više se oslanjaju na informacijske sustave koji omogućuju učinkovito upravljanje podacima, precizno praćenje poslovnih procesa i optimizaciju radnih zadataka. U skaldu s tim, ovaj završni rad bavi se razvojem oblak aplikacije E-Dimnjačar.

Aplikacija je namjenjena za evidenciju i upravljanje poslovima dimnjačarske službe na području grada. Omogućuje vođenje podataka o građanima, planiranje i praćenje obavljenih usluga prema definiranim kalendarima te organizaciju rada po ulicama i naseljima. Uz pomoć JHipstera, aplikacija je razvijena tako da integrira najbolje prakse modernih web aplikacija, s naglaskom na brzinu razvoja, sigurnost i jednostavnost korištenja.

Ključne tehnologije korištene u razvoju aplikacije uključuju Angular, koji pruža dinamično i interaktivno korisničko sučelje, te Spring Boot, koji omogućuje stabilan i siguran poslužiteljski dio. Aplikacija koristi MySQL bazu podataka, prilagođenu radu u oblak okruženju, te pruža cjeloviti sustav za upravljanje poslovnim procesima unutar dimnjačarske službe. istovremeno zadržava visok stupanj skalabilnosti i jednostavnosti održavanja. Ova oblak aplikacija zadovoljava trenutne poslovne zahtjeve, te postavlja temelje za budući rast i unapređenja sustava.

2. CILJ RADA

Cilj ovog završnog rada je istražiti i prikazati opće metode razvoja cloud aplikacija, s naglaskom na praktičnu primjenu kroz izradu sustava E-Dimnjačar za evidenciju dimnjačarskih usluga.

U završnom radu su integrirane funkcionalnosti kao što su upravljanja podacima građana, lokacija, rasporeda i uslugaa. Uz to, cilj je pokazati kako pravilna integracija modernih tehnologija može unaprijediti učinkovitost poslovnih procesa i omogućiti lakše planiranje i praćenje radnih zadataka.

Dodatno, u završnom radnu je prezentiran način korištenje modernih razvojnih alata i tehnologija. Kroz detaljnu analizu i kategorizaciju tehnologija, rad pruža sveobuhvatan pregled svih komponenti sustava, njihovih međuodnosa i uloga u cjelokupnoj arhitekturi

aplikacije. Ova strukturirana analiza omogućuje jasno razumijevanje uloge svake tehnologije u razvoju aplikacije.

Tijekom razvoja aplikacije korišten je Jhipster koj omogućuje brži razvoj aplikacije, te olakšava implementaciju sigurnosnih mehanizama. Rad također prikazuje kako je moguće kroz modularni i skalabilni pristup postići visoku pouzdanost aplikacije, prilagodljivost promjenjivim zahtjevima korisnika i osiguranje visokih standarda sigurnosti podataka.

3. KORIŠTENE TEHNOLOGIJE

3.1. Računalni jezici

Računalni jezici predstavljaju način na koji ljudi komuniciraju s računalima kako bi im zadali zadatke i definirali ponašanje aplikacija. Za razliku od strojnog koda, koji je direktno razumljiv računalima, programski jezici su dizajnirani tako da budu razumljivi ljudima. Oni omogućuju programerima da na jednostavan i intuitivan način napišu upute koje će računalo izvršiti. Postoji nekoliko vrsta računalnih jezika, kao što su:

Programski jezici, koriste se za pisanje aplikacija i softverskih rješenja. Java je korištena kao ključni programski jezik za poslužiteljski dio (engl. *backend*) aplikacije. Odgovoran je za logiku aplikacije i obradu podataka na strani poslužitelja, druge strane na klijentskoj dijelu (engl. *frontend)* korišten je TypeScript, on proširuje JavaScript, te omogućuje pisanje složenijeg i sigurnijeg koda

Skripti jezici često se koriste za automatizaciju. Bash, koj nije korišten direktno, bio je uključen u procesu automatizacije kroz JHipster, koji poziva Bash skripte za generiranje različitih struktura i datoteka [2].

Upitni jezici koriste se za rad s bazama podataka i manipulaciju podacima. U ovom projektu SQL je korišten za rad s bazom podataka, omogućujući pohranu, dohvaćanje i manipulaciju podacima unutar aplikacije.

Opisni jezici služe za definiranje strukture podataka ili izgleda aplikacije. Primjeri uključuju HTML ji (engl. *hypertext markup language),* koji se koristi za definiranje strukture web stranica, i CSS (engl. *cascade style sheet*) za definiranje izgleda HTML strukture.

Jezici za modeliranje koriste se za definiranje strukture i odnosa unutar aplikacija. U ovom projektu korišten je JDL (engl. *JHipster Domain Language*), specifičan jezik za modeliranje klasa unutar JHipster aplikacija. JDL je omogućio jednostavno definiranje klasa i njihovih odnosa te automatsko generiranje koda za te klase [8].

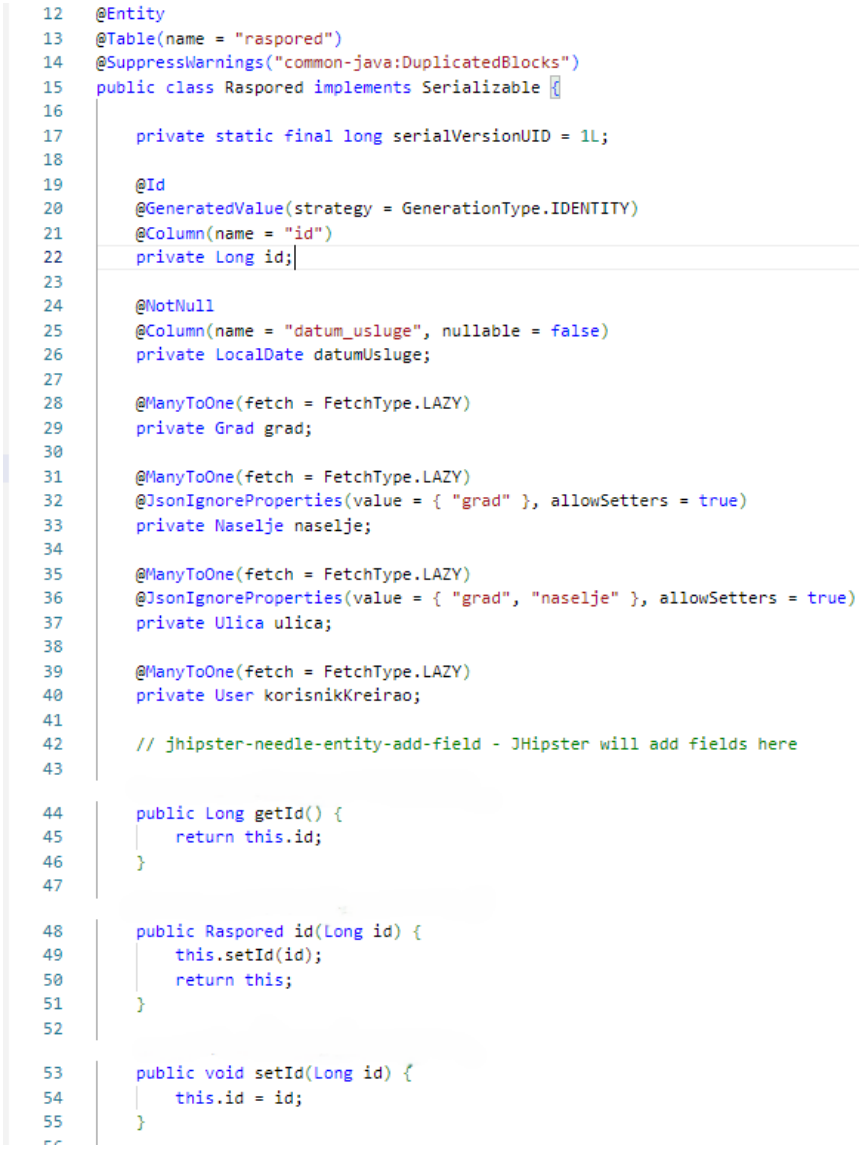
Ovi jezici, iako različiti po prirodi i svrsi, svi doprinose zajedničkom cilju. Oni omogućuju ljudima da efikasno upravljaju računalima i razvijaju kompleksne aplikacije na razumljiv i organiziran način. Slike u ovom podpoglavlju prikazuju definiciju iste klase raspored, s drugačijom tehnologijom. Ove vizualne usporedbe prikazuju razlike i sličnosti u programiranju među različitim jezicima. Prikazujuci kako se svaki jezik razlikuje strukturom i sintaksom.

3.1.1. Java

Java je složen, objektno orijentirani programski jezik koji je prvi put predstavljen 1995. godine, zahvaljujući radu Jamesa Goslinga i njegovog tima u tvrtki Sun Microsystems. Razvijena s naglaskom na sigurnost, jednostavnost i neovisnost o platformi, Java omogućuje izvršavanje programa na različitim operativnim sustavima bez potrebe za izmjenom koda. Ovu ključnu prednost osigurava JVM (engl. *Java virtual machine*), koji omogućuje prijenosnost i konzistentno izvršavanje Java programa na raznim platformama.

Osim što podržava objektno orijentirano programiranje, Java nudi podršku za generičko i funkcionalno programiranje, pružajući programerima fleksibilnost u razvoju softverskih rješenja. Zbog bogatog ekosustava alata, okvira i biblioteka, Java je omiljena za razvoj širokog spektra aplikacija, uključujući web i mobilne aplikacije, poslovne sustave, te igre. Nadalje, Java je poznata po svojoj skalabilnosti i stabilnosti, što je čini idealnim izborom za velike poslovne sustave. Slika 1 prikazuje dio Java koda, koj služi za definiranje klase raspored [7].

Slika 1. Java definiranje klase



```java
@Entity
@Table(name = "raspored")
@SuppressWarnings("common-java:DuplicatedBlocks")
public class Raspored implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @NotNull
    @Column(name = "datum_usluge", nullable = false)
    private LocalDate datumUsluge;

    @ManyToOne(fetch = FetchType.LAZY)
    private Grad grad;

    @ManyToOne(fetch = FetchType.LAZY)
    @JsonIgnoreProperties(value = { "grad" }, allowSetters = true)
    private Naselje naselje;

    @ManyToOne(fetch = FetchType.LAZY)
    @JsonIgnoreProperties(value = { "grad", "naselje" }, allowSetters = true)
    private Ulica ulica;

    @ManyToOne(fetch = FetchType.LAZY)
    private User korisnikKreirao;

    // jhipster-needle-entity-add-field - JHipster will add fields here

    public Long getId() {
        return this.id;
    }

    public Raspored id(Long id) {
        this.setId(id);
        return this;
    }

    public void setId(Long id) {
        this.id = id;
    }
```

Izvor: Autor

3.1.2. Typescript

TypeScript je napredna verzija JavaScript-a koja uvodi statičku tipizaciju i brojne napredne jezične značajke, čime poboljšava razvojni proces u odnosu na osnovni JavaScript. Razvijen od strane Microsofta 2012. godine, TypeScript omogućuje programerima pisanje čišćeg i strukturiranog koda, što olakšava održavanje projekata i poboljšava njihovu skalabilnost.

Jedna od ključnih prednosti TypeScript-a je statička tipizacija, koja omogućuje provjeru tipova varijabli, funkcija i povratnih vrijednosti tijekom razvoja. Što smanjuje mogućnost

pogrešaka prije nego što se kod izvrši, povećavajući sigurnost i pouzdanost aplikacija. Osim toga, TypeScript donosi napredne značajke poput klasa, sučelja, modula i generičkih tipova, što omogućuje programerima korištenje objektno orijentiranog i modularnog programiranja. Ove značajke poboljšavaju organizaciju koda i olakšavaju ponovno korištenje programskih komponenti [6].

TypeScript se kompajlira u JavaScript, što omogućuje njegovu kompatibilnost s većinom modernih preglednika. Zbog svoje sposobnosti da poveća produktivnost i sigurnost koda, TypeScript je postao popularan izbor za razvoj velikih mrežnih aplikacija i projekata koji zahtijevaju dugoročnu održivost i skalabilnost. Slika 2 primjer definiranja modela Raspored koristeći Typescript.

Slika 2. TypeScript definiranje modela

```typescript
import dayjs from 'dayjs/esm';
import { IGrad } from 'app/entities/grad/grad.model';
import { INaselje } from 'app/entities/naselje/naselje.model';
import { IUlica } from 'app/entities/ulica/ulica.model';
import { IUser } from 'app/entities/user/user.model';

Codeium: Refactor | Explain
export interface IRaspored {
  id: number;
  datumUsluge?: dayjs.Dayjs | null;
  grad?: IGrad | null;
  naselje?: INaselje | null;
  ulica?: IUlica | null;
  korisnikKreirao?: Pick<IUser, 'id' | 'login'> | null;
}


export type NewRaspored = Omit<IRaspored, 'id'> & { id: null };
```

Izvor: Autor

3.1.4. SQL

SQL (engl. *Structured query language*) je specijalizirani jezik za upravljanje i manipulaciju relacijskim bazama podataka. Razvijen 1970-ih godina, SQL omogućuje stvaranje, izmjenu i upravljanje bazama podataka, kao i efikasno dohvaćanje podataka putem jednostavnih i složenih upita.

SQL pruža mogućnost definiranja strukture baze podataka kroz DDL (engl. *Data definition language*) naredbe poput „CREATE“, „ALTER“ i „DROP“, koje omogućuju kreiranje i

modificiranje tablica, indeksa i drugih objekata unutar baze podataka. Također, SQL nudi DML (engl. *Data manipulation language*) naredbe poput „SELECT“, „INSERT“, „UPDATE“ i „DELETE“, koje omogućuju manipulaciju podacima unutar tablica.

Jedna od glavnih prednosti SQL-a je njegova standardizacija, što znači da većina relacijskih sustava za upravljanje bazama podataka podržava SQL u nekom obliku, omogućujući programerima rad s različitim bazama podataka uz minimalne prilagodbe koda.

SQL je temeljni jezik za rad s bazama podataka u raznim okruženjima, uključujući poslovne sustave, web aplikacije i analitiku podataka. U kombinaciji s programskim jezicima poput Jave i TypeScripta, SQL omogućuje razvoj složenih aplikacija koje efikasno upravljanje velikim količinama podataka. Zbog svoje fleksibilnosti i raširenosti, SQL je ključna tehnologija za razvoj baza podataka u modernih informatičkim sustavima. Slika 3 prikazuje definiranje tablice raspored u bazi podataka [10].

Slika 3. SQL definiranje tablice

```
--

CREATE TABLE `raspored` (
  `id` bigint(20) NOT NULL,
  `datum_usluge` date NOT NULL,
  `grad_id` bigint(20) DEFAULT NULL,
  `naselje_id` bigint(20) DEFAULT NULL,
  `ulica_id` bigint(20) DEFAULT NULL,
  `korisnik_kreirao_id` bigint(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Indexes for table `raspored`
--
ALTER TABLE `raspored`
  ADD PRIMARY KEY (`id`),
  ADD KEY `fk_raspored__grad_id` (`grad_id`),
  ADD KEY `fk_raspored__naselje_id` (`naselje_id`),
  ADD KEY `fk_raspored__ulica_id` (`ulica_id`),
  ADD KEY `fk_raspored__korisnik_kreirao_id` (`korisnik_kreirao_id`);


  --
ALTER TABLE `raspored`
  MODIFY `id` bigint(20) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=1506;
```

Izvor: Autor

3.1.5. JHipster Domain Language

JHipster Domain Language (JDL) je specifičan jezik za modeliranje domena unutar JHipster aplikacija. Omogućava definiranje entiteta, njihovih odnosa i konfiguraciju servisa unutar aplikacije putem jednostavnog i strukturiranog tekstualnog formata. JDL je posebno koristan jer omogućava brzo kreiranje kompleksnih modela.

Za ovaj projekt korišten je JDL za definiranje klasa s atributima i odnosima, što olakšava razvoj i održavanje aplikacija. JDL datoteke se jednostavno integriraju s JHipster generatorom, omogućujući automatsko generiranje koda za entitete, baze podataka, REST (engl. *REpresentational **State** Transfer*) API-jeve (engl. *application programming interface)* i druge komponente aplikacije na poslužiteljskoj i klijentskoj strani.

Na temelju JDL definicije, svi entiteti i prateće komponente su kreirani na poslužiteljskoj i klijentskoj strani, što značajno ubrzava proces razvoja i osigurava dosljednost implementacije. Zbog svoje jednostavnosti i učinkovitosti, JDL je koristan segment u razvoju JHipster aplikacija. Slika 4 prikazuje JDL kod za definiranje klase [8].

Slika 4. JDL definiranje klase

```
85
86  entity Raspored {
87    datumUsluge LocalDate required
88
89  }
90
91
92  relationship ManyToOne {
93    Raspored{grad(gradNaziv)} to Grad
94    Raspored{naselje(naseljeNaziv)} to Naselje
95    Raspored{ulica(ulicaNaziv)} to Ulica
96
97    Evidencija{Raspored(datumUsluge)} to Raspored
98
99    }
```

Izvor: Autor

3.2. Programski okviri

Programski okviri (engl. *framworks*) su tehnologije koje pružaju programerima strukturu i set biblioteka kako bi ubrzali proces razvoja programa. Ovi okviri omogućuju olakšano kreiranje aplikacija kroz ponovno korištenje koda, podršku za arhitekturu i funkcionalnosti, te automatsku konfiguraciju. Njihova svestranost često uključuje ugrađene alate za testiranje i upravljanje bazama podataka, što dodatno olakšava razvojni proces. Jedan od ključnih aspekata programskih okvira je da oslobađaju programere od infrastrukturnih detalja, omogućujući im da se usredotoče na poslovnu logiku aplikacije. To povećava produktivnost i učinkovitost, jer programeri mogu brže razvijati funkcionalnosti i manje

vremena troše na rješavanje tehničkih pitanja. Zahvaljujući programskim okvirima, programeri mogu graditi aplikacije na čvrstoj temeljnoj strukturi koja promiče dobre prakse i organizaciju koda. Ovo također olakšava održavanje aplikacija tijekom vremena, jer programeri mogu lakše pratiti i razumjeti kod koji je strukturiran i organiziran na temelju konvencija koje nudi okvir. Kroz upotrebu programskih okvira, programeri mogu iskoristiti širok spektar funkcionalnosti i gotovih rješenja za uobičajene zadatke, kao što su autentifikacija, autorizacija, rute, upravljanje stanjem, i mnoge druge. Što štedi vrijeme i resurse, a također osigurava dosljednost i kvalitetu u razvoju aplikacija. Kao rezultat, programski okviri imaju bitnu ulogu u modernom razvoju aplikacija, potičući produktivnost i olakšavajući programerima da kreiraju sofisticirane aplikacije koje zadovoljavaju visoke standarde kvalitete.

3.2.1. Spring Boot

SpringBoot je moćan programski okvir za razvoj Java aplikacija. On omogućuje brz i jednostavan razvoj samostalnih, produkcijski spremnih aplikacija. Spring Boot se temelji na Spring okviru i pruža mnoge njegove funkcionalnosti, ali s fokusom na jednostavnost konfiguracije i olakšavanje razvoja. Glavne značajke Spring Boota uključuju ugrađene poslužitelje, automatsku konfiguraciju, podršku za mikro servisnu arhitekturu, sigurnost, upravljanje bazama podataka i RESTful API-je. Također, Spring Boot ima bogat ekosustav proširenja, biblioteka i alata koji olakšavaju razvoj aplikacija. Spring Boot koristi princip "konvencija nad konfiguracijom". On pruža jednostavnu i deklarativnu sintaksu za definiranje konfiguracija, rute i interakcije s bazama podataka. Spring Boot također pruža podršku za testiranje aplikacija kroz ugrađene alate i biblioteke. Nadalje ima mogućnosti upravljanja ovisnostima, što pojednostavljuje upravljanje vanjskim bibliotekama i ažuriranjima. Uz sve ove značajke, Spring Boot je popularan izbor za razvoj mikro servisnih arhitektura, web aplikacija, servisa i poslovnih aplikacija. Njegova jednostavnost, skalabilnost i snažne mogućnosti čine ga moćnim okvirom za razvoj Java aplikacija [9].

3.2.2. Angular

Angular je otvoreni okvir za izradu dinamičkih i modernih web aplikacija. Napisan je u TypeScriptu i održava ga Google. Ovaj okvir omogućuje razvoj i testiranje velikih aplikacija s lakoćom. Angular se koristi za izradu aplikacija s jednom stranicom za mobilne uređaje i računala. Osnovne značajke Angulara uključuju deklarativno sučelje koje koristi

HTML za definiranje korisničkog sučelja aplikacije, te TypeScript, što pruža veću sigurnost s podrškom za tipove i pomaže u otkrivanju pogrešaka pri pisanju koda. Angular također olakšava testiranje, omogućava jednostavno automatizirano testiranje. Za razliku od drugih okvira, Angular koristi jedan modul za jednu datoteku, što pojednostavljuje upravljanje modulima. Angular se često koristi za izradu aplikacija s velikim obimom podataka i kompleksnih funkcionalnosti. Programeri vole Angular zbog njegove modularnosti, lakoće testiranja i produktivnosti. Ovaj okvir pruža snažne alate za upravljanje stanjem aplikacija, ruta, animacije i mnoge druge značajke koje olakšavaju izradu web aplikacija [6].

3.2.3. Angular i Spring Boot

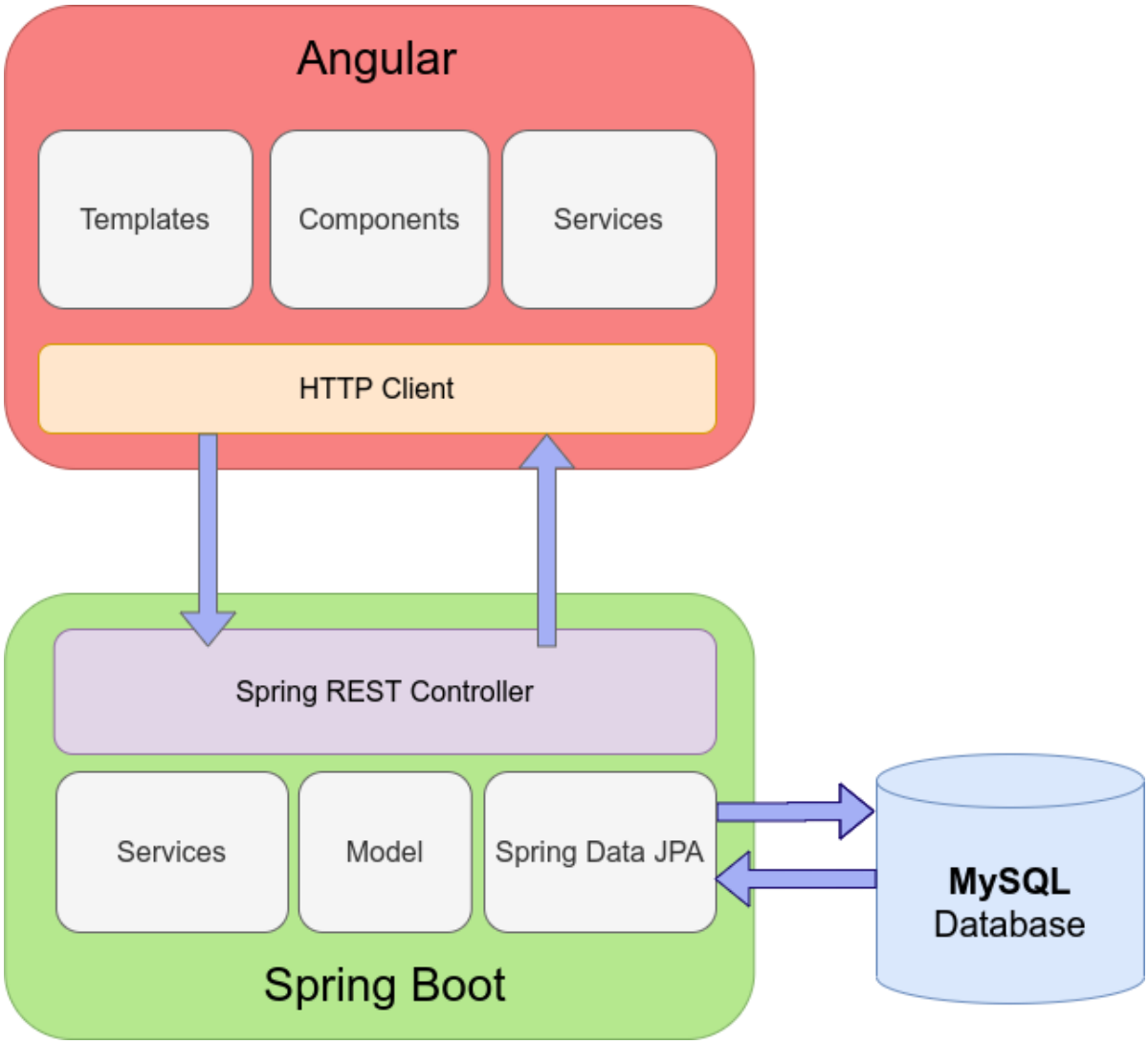
Spring Boot i Angular su dva različita, ali moćna programska okvira koji se često koriste u izradi modernih web aplikacija. Glavna sličnost između ova dva okvira je da oba omogućuju razvoj aplikacija s jednom stranicom koje donose bogato korisničko iskustvo. Oboje također podržavaju modularnost, što olakšava organizaciju i upravljanje velikim aplikacijama, omogućuje timovima da rade neovisno jedni o drugima na različitim dijelovima aplikacije, što povećava produktivnost i omogućuje lakše održavanje. Ipak postoje i razlike između Spring Boota i Angulara. Spring Boot se koristi za izradu serverskog dijela aplikacije, koji uključuje logiku poslovanja, pristup bazi podataka, autentifikaciju i autorizaciju. S druge strane, Angular se koristi za izradu klijentskog dijela aplikacije, koji uključuje korisničko sučelje, interakciju s korisnikom i prikazivanje podataka. Spring Boot koristi arhitekturu zasnovanu na MVC (engl. *Model-View-Controller*) strukturi uz podršku RESTful servisa za komunikaciju s klijentskim djelom. Angular koristi arhitekturu zasnovanu na komponentama, gdje svaka komponenta predstavlja neki dio korisničkog sučelja. Angular i Spring Boot su izvrsna kombinacija za razvoj modernih web aplikacija iz više razloga. Ove tehnologije pružaju kompletno rješenje za izgradnju aplikacija od početka do kraja. Spring Boot, kao programski okvir za logiku, omogućuje brzo i jednostavno postavljanje RESTful API-ja, što olakšava komunikaciju između klijentskog i serverskog dijela. S druge strane, Angular kao programski okvir za dizajn, pruža moćne alate za izradu korisničkog sučelja i omogućuje dinamičko prikazivanje podataka. Iduća prednost ove kombinacije je njihova popularnost i podrška zajednice. Oba imaju veliku zajednicu razvojnih programera, pa je lako pronaći resurse, dokumentaciju i rješenja za probleme koje se mogu pojaviti tijekom razvoja aplikacija. Međutim, postoje i neke mane koje treba uzeti u obzir. Za razvoj u ovoj kombinaciji potrebno je poznavanje Jave i TypeScripta, što može predstavljati izazov za neke programere, pogotovo za one koji su više usmjereni na samo jedan od ovih jezika. Također,

Angular i Spring Boot mogu zahtijevati veće tehničko znanje u usporedbi s nekim drugim razvojnim okvirima i alatima. Što može produžiti vrijeme potrebno za razvoj.

Integracija Angulara na Spring Boot u aplikacije je moguće povezati na više načina. Kao dvije odvojene aplikacije na odvojenim poslužiteljima, te kao Jednu aplikaciju na jednom poslužitelju. Obe od ovih metoda integracije imaju svoje prednosti i nedostatke. Jedna ključna razlika između ove dvije metode leži u tome što se u slučaju kada su aplikacije zapakirane kao jedna, komunikacija između korisničkog sučelja i logičkog dijela odvija interno, što znatno pojednostavljuje komunikaciju.

Za bolje razumijevanje integracije Angulara i Spring Boota. Slika 5 prikazuje arhitekturu web aplikacije koja koristi ove okvire zajedno s MySQL bazom podataka. Ova vizualna reprezentacija ističe interakciju između klijentske i serverske komponente, pokazujući kako zajedno rade kako bi stvorili kohezivnu aplikaciju [6].

Slika 5. Shema povezivanja Angular i SpringBoot aplikacije



Izvor:

https://www.javaguides.net/2020/07/spring-boot-angular-10-crud-example-tutorial.html (pristup: 10.02.2024)

3.3. Razvojne platforme

Razvojne platforme predstavljaju skup okvira, servisa koji omogućuju programerima izradu, testiranje i implementaciju softverskih aplikacija. Ove platforme integriraju različite tehnologije i pružaju unaprijed definirane strukture i arhitekture, čime ubrzavaju i pojednostavljuju proces razvoja. Platforme poput JHipstera posebno su korisne jer kombiniraju najbolje prakse i tehnologije u jednu cjelinu, omogućujući učinkovit i standardiziran razvoj aplikacija.

3.3.1. Jhipster

JHipster je razvojna platforma koja omogućuje brzo i učinkovito kreiranje modernih web aplikacija. Kombinira najbolje značajke Jave i TypeScripta, te integrira okvire kao što su Spring Boot i Angular. JHipster omogućuje programerima generiranje cjelokupne aplikacijske strukture, uključujući logiku, korisničko sučelje, i konfiguracije za implementaciju, čime se znatno skraćuje vrijeme potrebno za razvoj.

Platforma je nastala 2013. godine kao projekt Juliena Duboisa, francuskog inženjera, s ciljem pojednostavljenja procesa izrade modernih web aplikacija. JHipster je projekt otvorenog koda (engl. *open-source*), što znači da je njegov izvorni kod javno dostupan i da ga razvija i održava globalna zajednica programera. Ova otvorenost omogućuje brzu evoluciju platforme, konstantna poboljšanja i prilagodbe najnovijim trendovima u web razvoju [4].

JHipster dolazi s JDL-om, što dodatno ubrzava proces kreiranja baza podataka i povezivanja različitih komponenti aplikacije. Uz to, JHipster podržava integraciju s alatima za upravljanje verzijama, te omogućuje lako skaliranje i upravljanje aplikacijama u produkcijskom okruženju. Zahvaljujući aktivnoj zajednici i redovitim ažuriranjima, JHipster ostaje relevantan alat u svijetu razvoja enterprise aplikacija, prilagođavajući se novim tehnologijama i potrebama industrije.

4. ALATI KORIŠTENI ZA IZRADU APLIKACIJE

4.1. Naredbeni redak

Naredbeni redak (engl. *command line*) ili terminal predstavlja tekstualno sučelje za interakciju s operativnim sustavom direktnim unosa naredbi. Naprednim korisnicima omogućava efikasnije korištenje računala. U razvoju ove aplikacije korištenje naredbenog retka je neophodno, jer omogućava efikasno upravljanje alatima i ovisnostima te izradu aplikacije.

4.1.1. CMD

Command Prompt je naredbeni redak dostupan u Windows operativnim sustavima. U početku ovog projekta, CMD je korišten za instalaciju Node.js i npm (engl. *Node Package Manager*), koji su neophodni alati za pokretanje JHipstera. Također korišten je za instalaciju JDK-a, potrebnog za razvoj Java aplikacija, kao i za instalaciju JHipster generatora pomoću

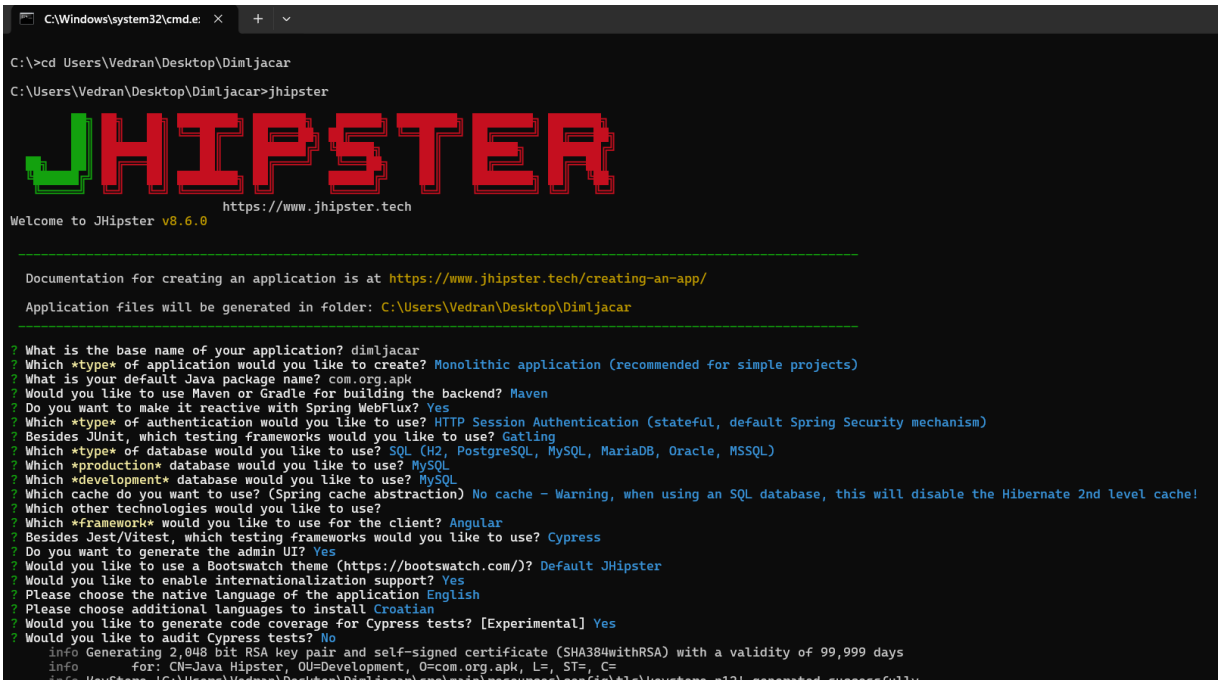
naredbe „npm install -g generator-jhipster“. Dobra praksa je nakon instalacije, provjera verzija tih te validirati ispravnu instalaciju, što je također moguće kroz naredbeni redak.

Osim za instalaciju, generiranje aplikacije, konfiguraciju i pokretanje. CMD je također korišten za pregledavanje logova i praćenje aktivnosti na serveru. Ovo je posebno korisno jer omogućuje uvid u procese i informacije koje nisu vidljive u konzoli preglednika, poput detaljnih logova. CMD je stoga ključan alat za praćenje rada i rješavanje problema tijekom razvoja aplikacije.

4.1.2. Kreiranje Jhipster aplikacije

Nakon pripreme okruženja, idući korak je izvršavanje naredbe „jhipster“ u željenom direktoriju, što pokreće generator aplikacije. Ovaj generator vodi korisnika kroz niz pitanja kako bi odabrao potrebne konfiguracije za aplikačujući izbor okvira za korisničko sučelje, baze podataka i dodatnih funkcionalnosti. Slika 6 prikazuje kreiranje aplikacije. Korištenje naredbenog retka omogućava jednostavan odabir različitih opcija. Kroz naredbeni redak, korisnik može jednostavno definirati ključne postavke za aplikaciju, prilagođavajući je specifičnim potrebama projekta.

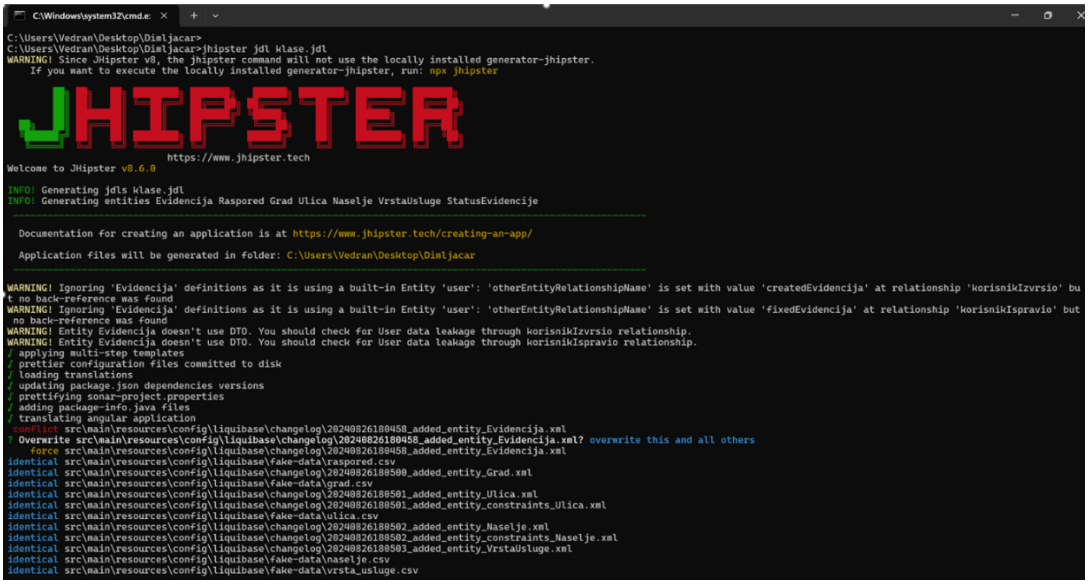
Slika 6. Kreiranje aplikacije



Izvor: Autor

Naredbom „jhipster import-jdl klase.jdl“ moguće je jednostavno uvesti JDL datoteku. Ova naredba automatski generira entitete i relacije koje su definirane u JDL datoteci, čime se ubrzava proes razvoja. Slika 7 prikazuje dodavanje klasa.

Slika 7. Dodavanje klasa

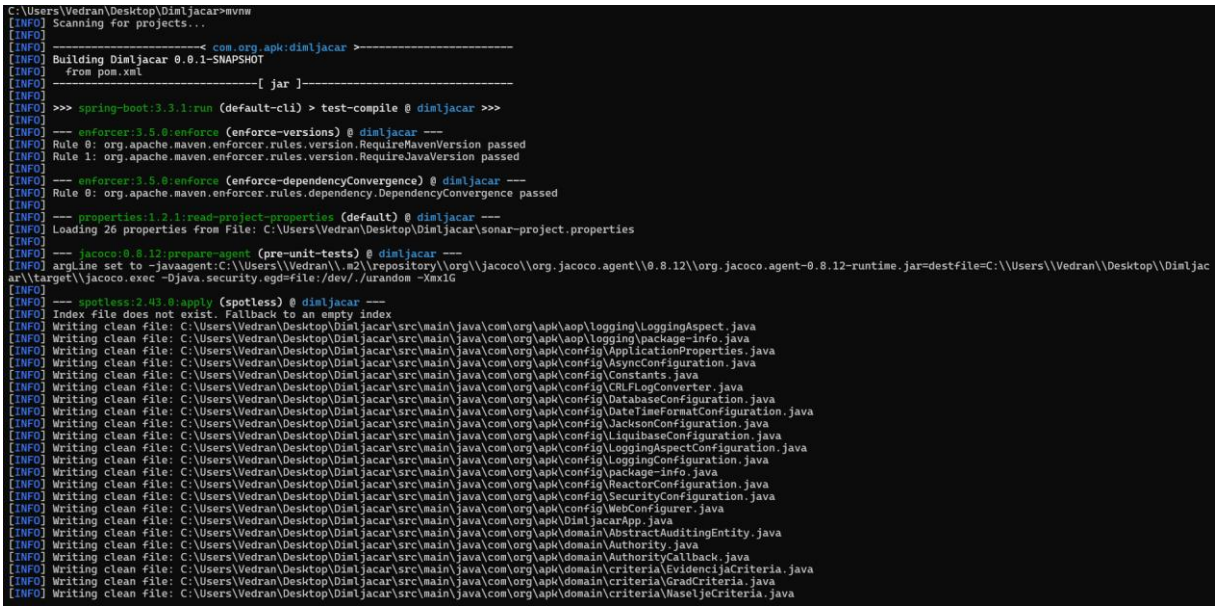


Izvor: Autor

Nakon što je konfiguracija završena, aplikacija se izgrađuje pomoću naredbe „mvnw“. Ova naredba upućuje računalo da izgradi aplikaciju koristeći Maven na Windows operativnom sustavu.

Ovaj proces omogućava efikasno i jednostavno generiranje, konfiguraciju i izgradnju JHipster aplikacije, bez i jednog klika miša. Slika 8 prikazuje pokretanje aplikacije.

Slika 8. Pokretanje aplikacije



Izvor: Autor

4.2. Uređivač teksta

Uređivači teksta nisu namijenjeni samo za pisanje koda. Oni predstavljaju univerzalne alate koji omogućuju obradu tekstualnih podataka u digitalnom formatu. Uređivači teksta mogu varirati od jednostavnih alata poput Notepada, koji omogućuje osnovno uređivanje tekstualnih datoteka, do složenijih sustava poput VIM-a (engl. *Vi Improved*), koji pruža napredne mogućnosti upravljanja tekstom putem naredbenog retka.

Uređivači teksta mogu se podijeliti u dvije glavne kategorije: oni koji nude grafičko sučelje prilagođeno korisnicima, i uređivače bez grafičkog sučelja, koji rade unutar terminala i oslanjaju se na korištenje tipkovnice.

Dosta programera koristi integrirano razvojno okruženje (IDE) tijekom programiranja, testiranje i otklanjanja pogrešaka, zbog integriranih alata i funkcionalnosti u jednom sučelju. Međutim, za ovaj projekt, IDE nije bio korišten. Složeni zadaci, poput otklanjanja pogrešaka i izgradnje aplikacije, uspješno su obavljeni kroz naredbeni redak.

4.2.1. Visual studio code

Visual Studio Code (VSC) je jedan od najpopularnijih uređivača teksta, posebno dizajniran za razvojne zadatke. Radi se o uređivaču s grafičkim sučeljem, kojeg je razvio

Microsoft, a koji nudi širok spektar funkcionalnosti prilagođenih programerima. VSC podržava razne programske jezike i tehnologije, omogućujući korisnicima da prošire njegove mogućnosti putem raznovrsnih ekstenzija.

Jedna od ključnih značajki VSC-a je integracija s alatima za kontrolu verzija poput Git-a. Osim toga, VSC pruža, integrirani terminal, podršku za pametno dovršavanje koda (engl. *IntelliSense*), te alat za jednostavno pokretanje skripti i izgradnju aplikacija izravno unutar uređivača [11].

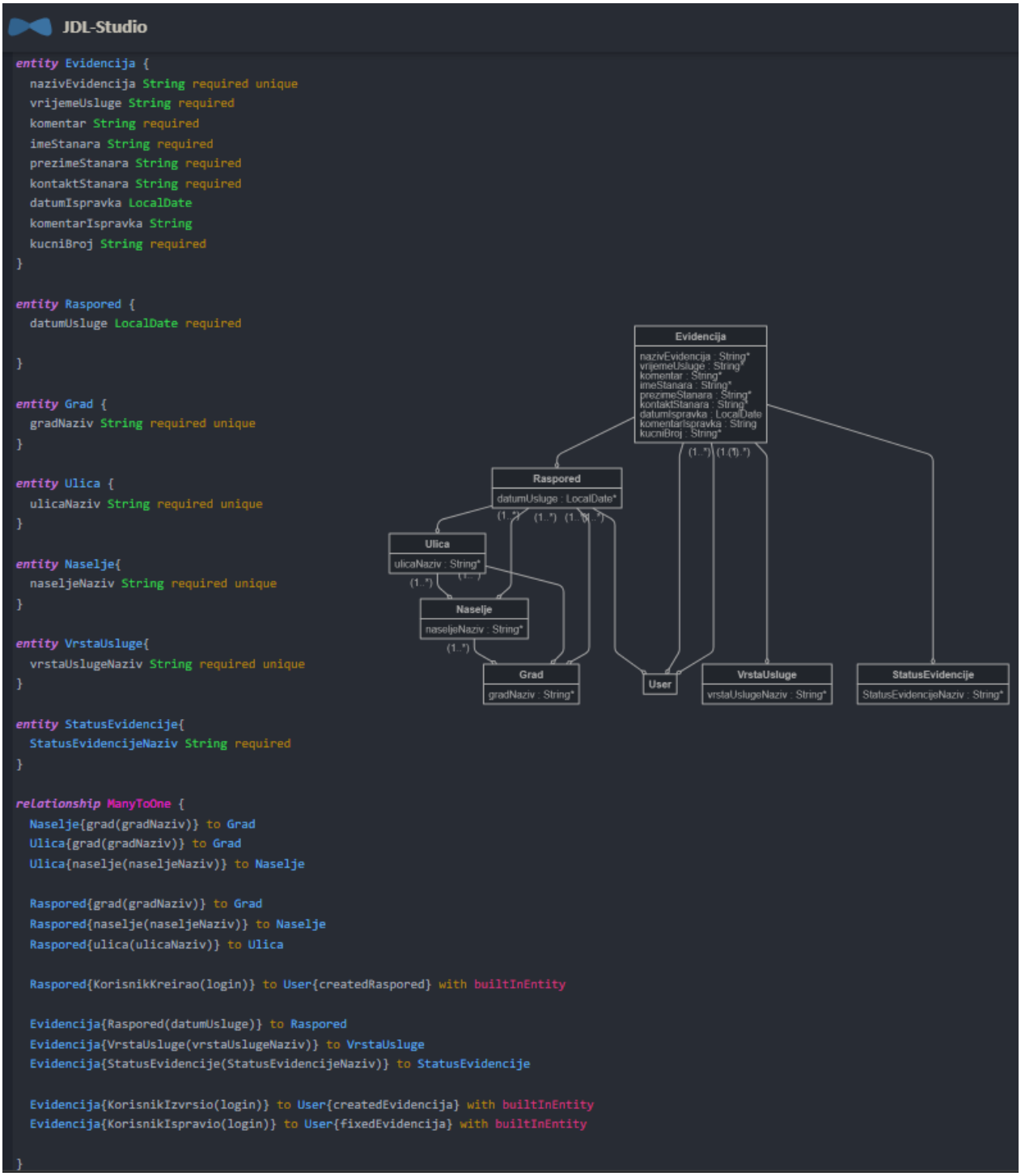
Iako VSC nudi mnoge značajke koje se nalaze u punopravnim IDE-ovima, njegova fleksibilnost i jednostavnost čine ga idealnim izborom za projekte koji ne zahtijevaju kompleksne razvojne okoline. U našem projektu, VSC je korišten kao primarni uređivač zbog svoje prilagodljivosti.

4.2.2. JDL studio

JDL Studio je web alat razvijen od strane JHipster zajednice, koji omogućuje jednostavno kreiranje i vizualizaciju JDL modela. Koristeći JDL Studio, programeri mogu na intuitivan način definirati arhitekturu aplikacije prije nego što krenu s generiranjem koda.

JDL Studio pruža grafičko sučelje za kreiranje, modificiranje i dijeljenje JDL modela. Nakon definiranja, JDL datoteka se može uvesti u aplikaciju za automatsko generiranje klasa, tablica i kontrolera. U ovom projektu, JDL Studio je korišten za definiranje entiteta i njihovih odnosa, omogućujući jasan prikaz strukture podataka i brze prilagodbe modela prema potrebama aplikacije kroz intuitivno sučelje. Slika 10 prikazuje osnovni model entitea i njihovih relacija, korišten za izradi aplikacije E dimljčar. Na temelju ovog koda Jhipster generira aplikaciju. Slika 9 također prikazuje dijagram entiteta i njihovih relacija [8].

Slika 9. JDL studio definiranje klasa



Izvor: Autor

4.3. Serverski alati

4.3.1. XAMPP

Cross-Platform, Apache, MariaDB, PHP, and Perl (engl. *XAMPP*) je besplatan i otvoreni paket softvera koji se koristi za brzo postavljanje lokalnog web poslužitelja. Sastoji se od Apache HTTP (engl. *HyperText Transfer Protocol*) servera, MySQL baze podataka, i skriptnog jezika PHP (engl. *Personal home page*) i Perl-a. Glavna prednost korištenja XAMPP-a je njegova jednostavnost u instalaciji i korištenju, što ga čini idealnim alatom za programere koji žele brzo testirati web aplikacije lokalno.

XAMPP omogućuje kreiranje lokalnog razvojnog okruženja u kojem programeri mogu testirati i razvijati svoje aplikacije u sigurnom okruženju prije nego što ih objave na produkcijskim serverima. U projektu, XAMPP je korišten za simulaciju servera, omogućujući testiranje klijentske funkcionalnosti aplikacije i interakciju s bazom podataka. Korištenje XAMPP-a značajno je ubrzalo proces razvoja, jer je omogućilo brze iteracije i testiranje bez potrebe za stalnim implementacijama na vanjske servere [15].

4.4. Alati za izgradnju aplikacije

Alati za izgradnju aplikacije omogućuju automatizaciju zadataka kao što su kompilacija koda, upravljanje ovisnostima, pokretanje testova i stvaranje izvršnih datoteka. Ovi alati osiguravaju da se aplikacija može dosljedno izgraditi na različitim računalima.

4.4.1. Maven

Maven je popularan alat za upravljanje projektima i izgradnju aplikacija, razvijen prvenstveno za projekte temeljene na Javi. Omogućuje jednostavno upravljanje ovisnostima i automatsku izgradnju projekta prema unaprijed definiranim konfiguracijama. U ovom projektu, Maven je korišten za upravljanje ovisnostima, te za automatizaciju procesa izgradnje aplikacije, osiguravajući konzistentnost i uštedu vremena tijekom razvoja.

4.5. Sustavi za upravljanje izvornim kodom

Sustavi za upravljanje izvornim kodom omogućuju timovima programera praćenje promjena, suradnju na projektima i praćenje verzija aplikacija. Većina modernih sustava

pohranjuje kod na udaljeni server, što timovima omogućuje rad na projektu bez obzira na lokaciju. Ovi sustavi pomažu u praćenju i kontroli promjena, smanjujući mogućnost pogrešaka i osiguravajući kontinuitet u razvoju.

4.5.1. Git

Git je distribuirani sustav za upravljanje verzijama koda, aplikacija otvorenog koda, koju je razvio Linus Torvalds 2005. godine. Za razliku od centraliziranih sustava, Git omogućuje svakom suradniku u projektu da ima potpunu kopiju cijele povijesti verzija, čime se omogućuje brži i fleksibilniji rad. Git je postao standard u industriji za praćenje promjena u softverskim projektima zbog svoje brzine, efikasnosti i mogućnosti rada bez stalne povezanosti s centralnim serverom.

4.5.2. Git Hub

GitHub je platforma za mrežno dijeljenje Git repozitorija, koja omogućuje programerima da pohranjuju i dijele svoje projekte online. Osim osnovnih funkcionalnosti Gita, GitHub pruža dodatne alate za suradnju, kao što su praćenje problema, upravljanje zahtjevima za povlačenje i integracije s drugim razvojnim alatima. GitHub je postao središnje mjesto za suradnju otvorenog koda, gdje milijuni programera dijele svoje projekte i doprinose razvoju softvera na globalnoj razini. [1]

5. STRUKTURA APLIKACIJE

5.1 Tehnička struktura

Tehnička struktura aplikacije temelji se na MVC (engl. Model-View-Controller) arhitekturi, koja odvaja prikaz, poslovnu logiku i pristup podacima radi boljeg upravljanja i održavanja aplikacije. Osim toga mogli bismo podijeliti aplikaciju u tri osnovne cjeline a to su klijentska strana, poslužiteljska strana, te baza podataka. Svaka od tih cjelina koristi različite tehnologije, te obavlja različite zadatke.

U razvoju modernih web aplikacija, ključno je postići optimalan balans između klijentske i poslužiteljske strane. Dok je cilj smanjiti opterećenje poslužitelja koji istovremeno opslužuje više korisnika, važno je zadržati određene operacije na serverskoj strani zbog sigurnosti i integriteta podataka.

Općenito, arhitektura aplikacije teži tome da se obrada podataka primarno odvija na poslužiteljskoj strani, dok je klijentska strana zadužena za prikaz i interakciju s korisnikom. Iako tehnološki napredak omogućuje izvršavanje složenih operacija i na klijentskoj strani, to nije uvijek preporučljiva praksa, posebno kada se radi o osjetljivim podacima ili pristupu bazi podataka.

Sigurnost i kontrola nad podacima ostaju primarni razlozi za zadržavanje kritičnih operacija na poslužiteljskoj strani. Ovakav pristup ne samo da štiti osjetljive informacije, već i osigurava konzistentnost podataka i omogućuje centralizirano upravljanje poslovnom logikom. Balansiranje opterećenja između klijenta i poslužitelja stoga postaje umjetnost optimizacije performansi, sigurnosti i korisničkog iskustva

5.1.1. Klijentska strana

Klijentska strana aplikacije odnosi se na dio koji korisnici izravno vide. Tako se naziva jer se sam kod izvršava u pretraživaču korisnika, na korisnikovom uređaju. U ovom slučaju Klijentska strana koristi TypeScript i Angulara za prikazivanje korisničkog sučelja. Svaka klasa na klijentskoj strani sadrži svoj modul za ažuriranje (engl. *update*), prikazivanje u list (engl. *list*), brisanje (engl. *delete*) i prikazivanje detalja (engl. *details*). Komponenta za ažuriranje se koriti za izradu novih ili uređivanje postojećih objekta određene klase.

5.1.2. Poslužiteljska strana

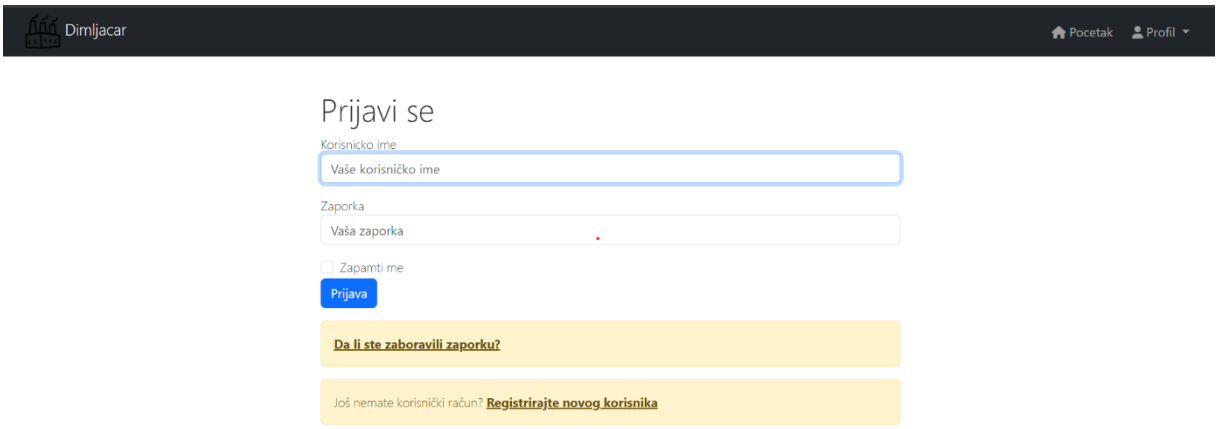
Poslužiteljska strana aplikacije obrađuje poslovnu logiku, upravlja podacima i komunicira s bazom podataka. Taj dio koda se izvršava kod poslužitelja ili na udaljenom serveru. U ovoj aplikaciji sve klase postoje na klijentskoj i serverskoj strani. Svaka klasa na poslužiteljska strana se sastoji se od nekoliko ključnih komponenti: modela, repozitorija, servisa i kontrolera. Model je definirana svojstvima, te predstavlja podatkovnu strukturu klase. Repozitorij je zadužen za pristup i upravljanje podacima u bazi podataka, a servisi odrađuju logiku obrade, te je veza između modela i kontrolera.

5.2. Tok korištenja aplikacije

Tok korištenja aplikacije prati slijed koraka koje korisnik poduzima prilikom interakcije s aplikacijom, počevši od prijave pa do stvaranja i upravljanja različitim entitetima.

Nakon sto je korisnik kreirao svoj korisnički račun, može se ulogirati u aplikaciju. Koristeći izabrano korisničko ime i lozinku. Slika 10 prikazuje stranicu za prijavu.

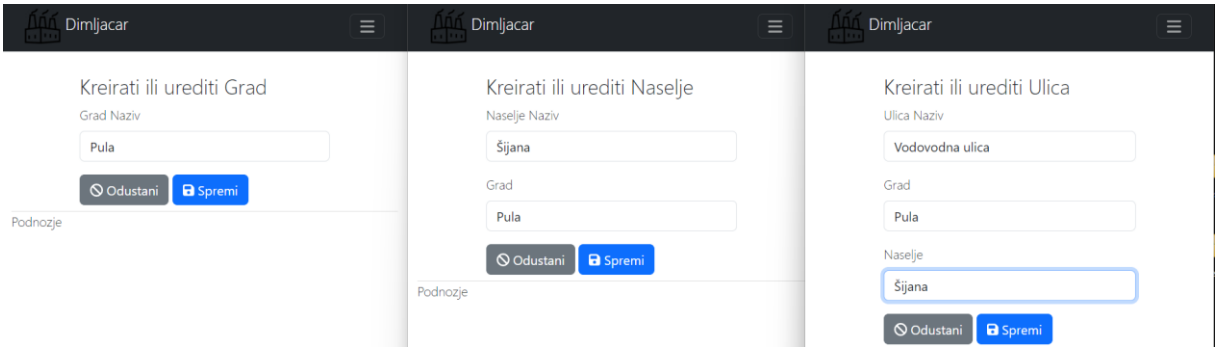
Slika 10. Prijava korisnika



Izvor: Autor

Nakon prijave, korisnik kreira lokaciju, na naći na da prvo kreira objekt tipa grad, nakon toga objekt tipa naselje dodaje u odabrani grad, te nakon toga kreira objekt ulica, koji pridružuje naselju i gradu. Slika 11 je skup od 3 slike, kojima se kreira lokacija.

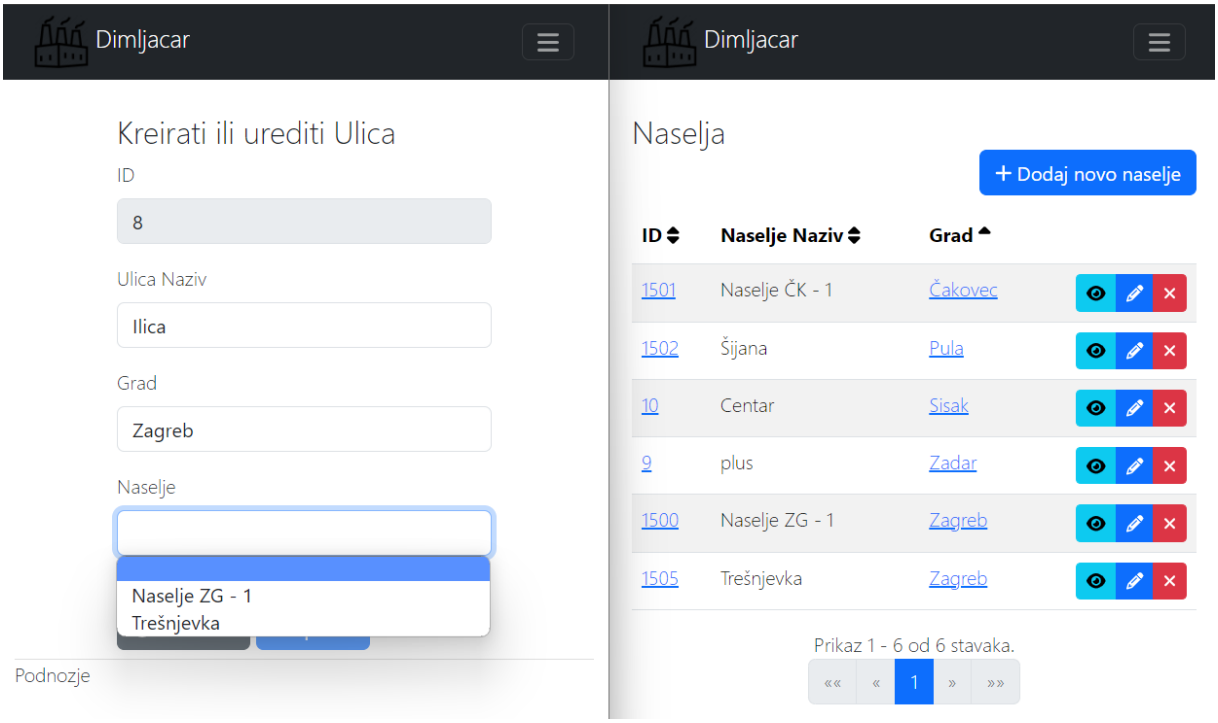
Slika 11. Kreiranje lokacije



Izvor: Autor

Za upisna polja veže se validacija koje onemogućava korisniku kreiranje gradova s istim nazivom, naselja s istim nazivom u istom gradu, te ulica s istim nazivom u istom naselju. Aplikacija navodi korisnika pri kreaciji ulice tako da, nakon korisnikovog odabira grada, u polju naselja prikazuju se samo naselja koja su pridružena odabranom gradu. Slika 12 prikazuje filtriranje neselja u gradu.

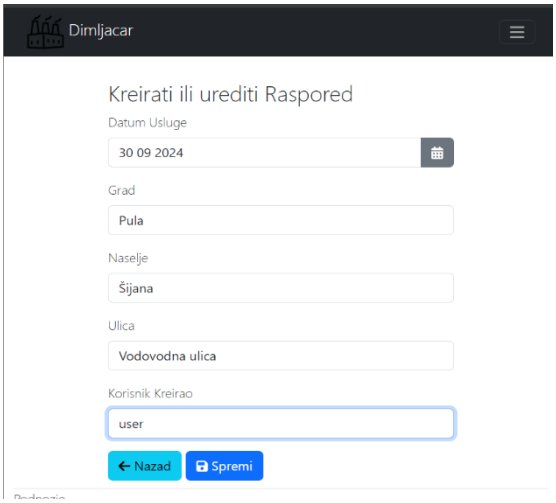
Slika 12. Filtriranje naselja u gradu



Izvor: Autor

Nakon što je korisnik kreirao lokaciju, u mogućnosti je kreirati raspored kojem se dodjeljuje odabrana lokacija. Osim lokacije za raspored se odabire datum usluge, te korisnik koji ga je kreirao. Slika 13 prikazuje kreiranje rasporeda.

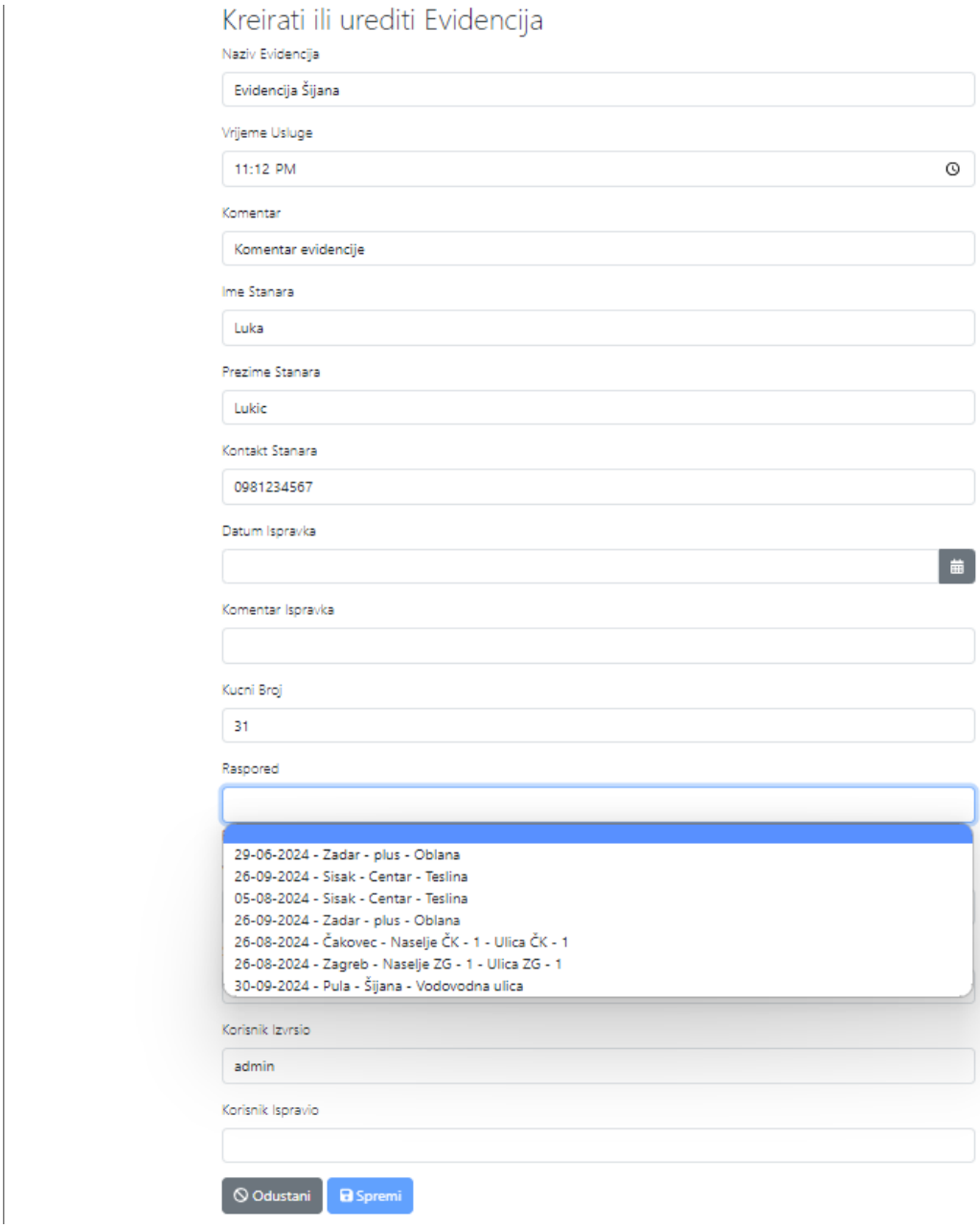
Slika 13. Kreiranje rasporeda



Izvor: Autor

Zadnji korak je evidencija, za kreaciju evidencije korisnik mora proći sve navedene korake, jer se svi ti objekti pridružuju direktno i indirektno evidenciji. Odabirom rasporeda korisnik povlači podatke o lokaciji i datumu. Kreira naziv evidencije, odabire vrijeme usluge, unosi podatke stanara, kućni broj, komentar. Odabire status evidencije, te vrstu usluge. U slučaju da je potrebna dodatna dorada za tu evidenciju moguće je odabrati datum ispravka i upisati komentar ispravka. Status evidencije i vrsta usluge su jednostavne klase koje povećavaju korisničku fleksibilnost aplikacije. Slika 14 kreiranje evidencije.

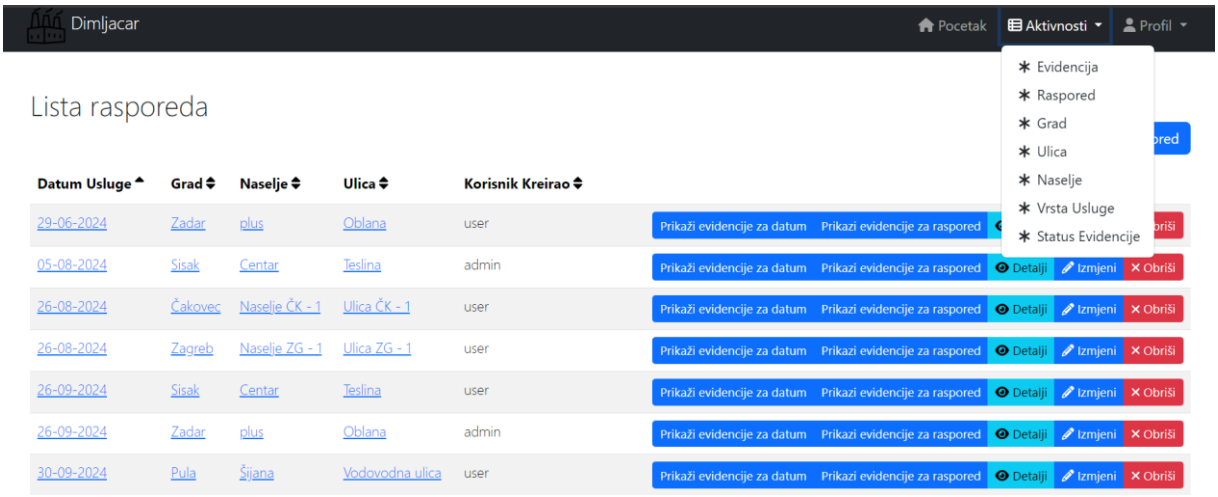
Slika 14. Kreiranje evidencije



Izvor: Autor

Jedna od funkcionalnosti aplikacije je mogućnost pretraživanja evidencija, po odabranom datum ili odabranom rasporedu. Te kreiranje dnevnog radnog naloga. Na naći da korisnik napravi PDF (engl. *Portable document format*) ispis za rezultate pretraživanja evidencija. Slika 15 prikazuje listu rasporeda.

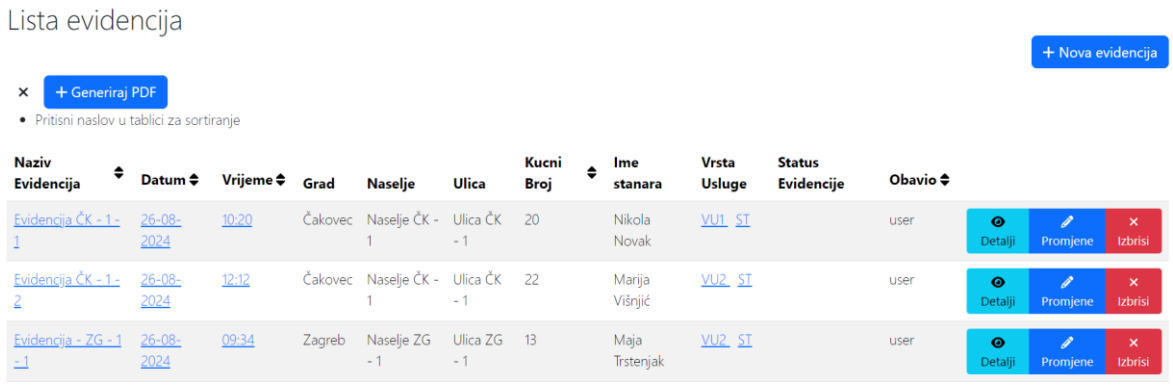
Slika 15. Lista rasporeda



Izvor: Autor

Ovaj slijed omogućuje korisniku logičan i strukturiran način unosa podataka u aplikaciju, od definiranja osnovnih podataka o lokaciji do stvaranja detaljnih evidencija. To sve omogučava efikasnije poslovanje i evidenciju rada. Slika 16 prikazuje filtriranje evidencija.

Slika 16. Filtriranje evidencija



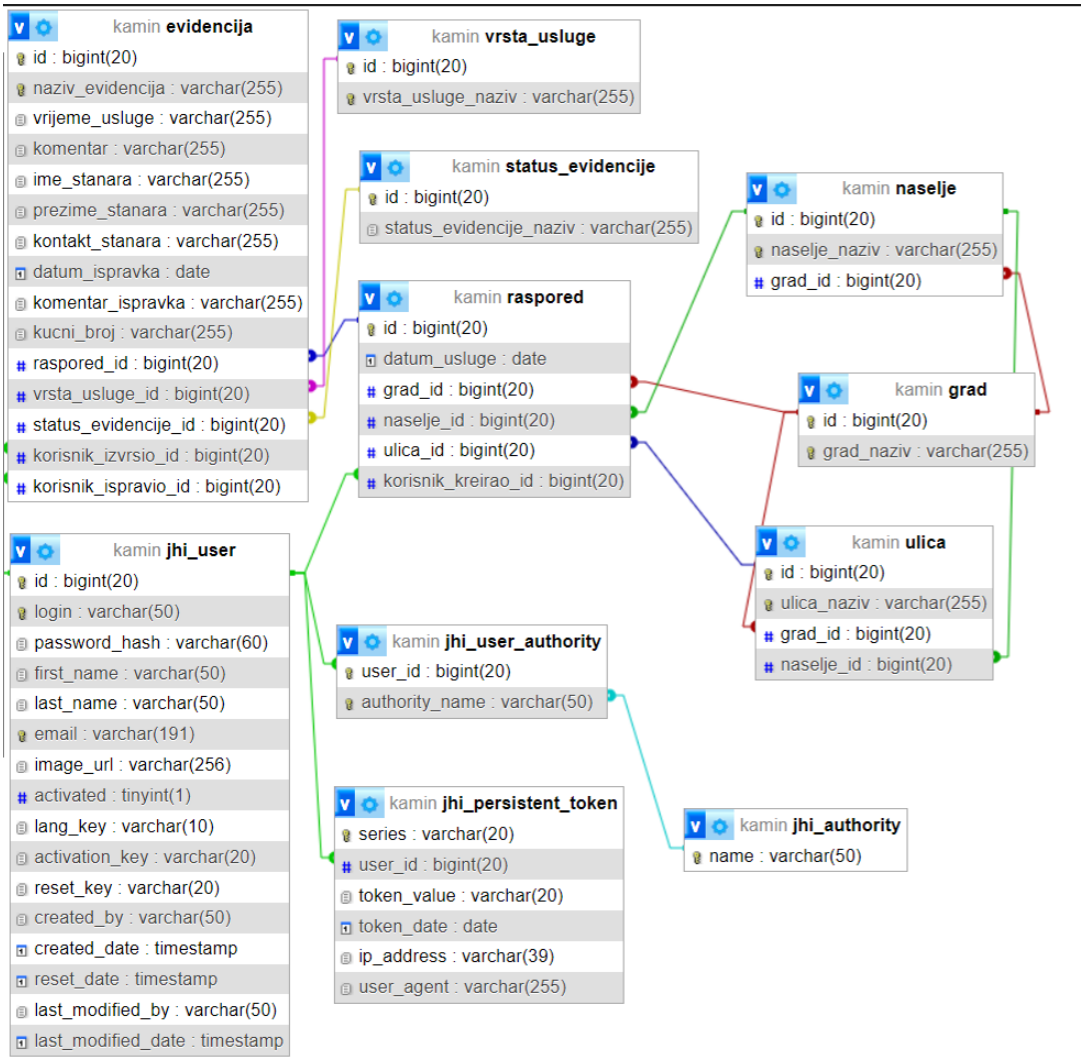
Izvor: Autor

Tehnička struktura aplikacije i korisnički tokovi pažljivo su osmišljeni kako bi omogućili jednostavan i logičan unos podataka, od kreiranja osnovnih entiteta do detaljne evidencije rada. Korisnici mogu lako pratiti proces kreiranja i upravljanja lokacijama, rasporedima i evidencijama, što značajno doprinosi efikasnosti poslovanja i preciznosti u vođenju evidencija. Ovakav pristup osigurava da svi koraci u aplikaciji budu povezani i da svaki unos podataka bude konzistentan i dobro integriran unutar sustava.

6. BAZA PODATAKA

Baza je bitan dio u radu aplikacije, omogućujući pohranu, pristup i upravljanje podacima na siguran i efikasan način.

Slika 17. Baza podataka



Izvor: Autor

U ovom projektu, baza podataka služi za pohranu svih informacija koje aplikacija koristi, od korisničkih podataka do poslovnih podataka. U ovom slučaju odabrana je relacijska baza podataka zbog svoje sposobnosti organiziranog pohranjivanja podataka u tablicama i podrške za složene upite.

U ovom projektu dio baze koj podataka koj se odnosi na korisnika je došao s aplikacijom. JDL je korištenj za kreiranje poslovne logike baze podataka. Slika 17 prikazuje dijagram baze podataka.

6.1. MySQL

MySQL je sustav za upravljanje relacijskim bazama podataka, otvorenog koda . Razvijen od strane MySQL AB, a sada u vlasništvu Oraclea, MySQL je jedan od najčešće korištenih sustav za upravljanje relacijskim bazama (engl. *RDBMS*). Njegova prednost leži u jednostavnosti korištenja, brzini i fleksibilnosti, što ga čini idealnim za širok raspon aplikacija, od malih web stranica do velikih sustava za upravljanje podacima. MySQL koristi SQL za upravljanje bazama podataka, omogućujući korisnicima kreiranje i manipulaciju podacima u strukturiranim tablicama. Za ovaj projekt, MySQL je odabran kao baza podataka zbog svoje pouzdanosti, performansi i široke podrške u zajednici. [10]

6.2. Apache

Apache HTTP Server je veb poslužitelj, s primarnom funkcijom serviranja web stranica i aplikacija korisnicima putem HTTP protokola. Apache je otvorenog koda, što omogućava prilagodbu i korištenje od strane zajednice. Tako da je u ovom projektu, Apache korišten za upravljanje HTTP zahtjevima između korisnika i poslužitelja, omogućujući aplikaciji da funkcionira na mreži.

6.3 Tomcat

Apache Tomcat je aplikacijski poslužitelj specijaliziran za izvršavanje Java web stranica. Tomcat pruža okruženje za pokretanje web aplikacija razvijenih u Javi i omogućava njihovu integraciju s bazom podataka i ostalim komponentama sustava. Osim osnovnih funkcionalnosti, Tomcat nudi podršku za sigurnost, upravljanje sesijama i druge usluge potrebne za rad web aplikacija. [15]

7. TESTIRANJE APLIKACIJE

Testiranje aplikacije je bitno za osiguravanju njezine kvalitete i stabilnosti. Kroz proces testiranja mogu se rano otkriti greške, optimizirati performanse te provjeriti ispravnost funkcionalnosti. U izradi ove aplikacije korištene su različite vrste testiranja, uključujući funkcionalno, potupuno (engl. *end-to-end*) i ručno testiranje.

Za potpuno testiranje korišten je Cypress, alat za automatizirano testiranje koji provjerava cjelokupni rad aplikacije, od početka do kraja. Ovakvo testiranje osigurava da sve komponente aplikacije međusobno surađuju kako je predviđeno te da nema problema u komunikaciji između različitih dijelova sustava [14].

JHipster automatizirano generira testne datoteke prilikom kreiranja aplikacije. Ove datoteke uključuju testove za svaku generiranu komponentu i klasu, pokrivajući funkcionalnosti poput prikaza, uređivanja, ažuriranja i prikaza u listi. Ova automatizacija značajno ubrzava razvojni proces i povećava pokrivenost testovima.

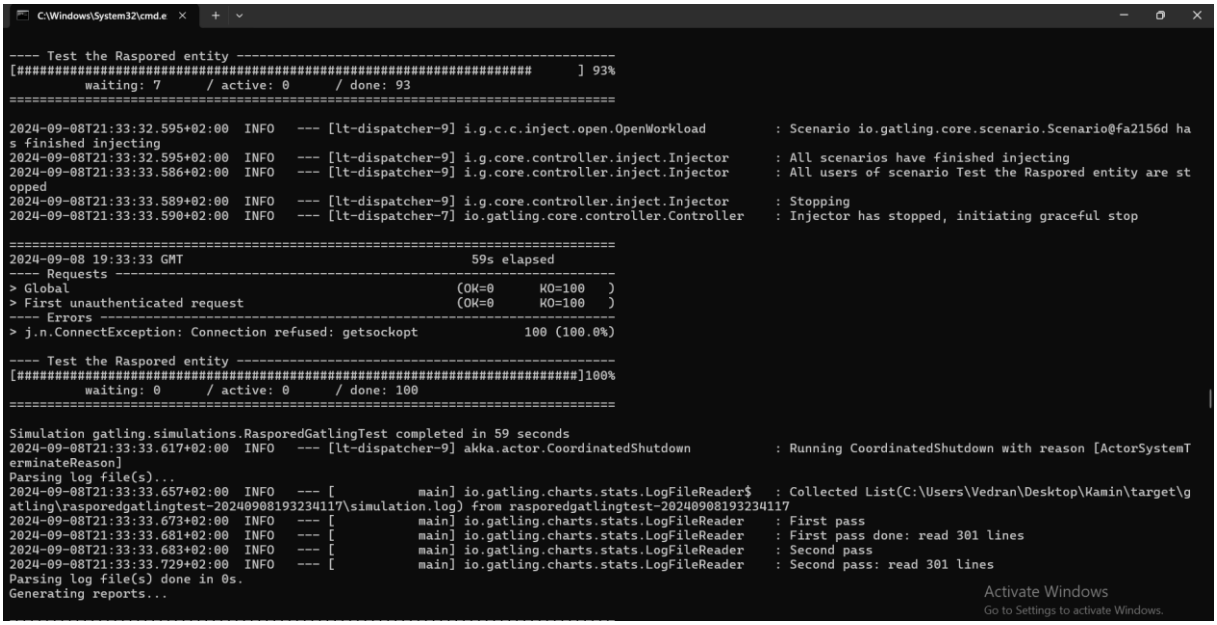
Uz automatizirano testiranje, tijekom izrade projekta korišteno je i ručno testiranje kako bi se dodatno provjerila ispravnost rada aplikacije. Ručno testiranje omogućuje programerima da izravno provjere funkcionalnost i korisničko iskustvo, što je bitno za otkrivanje suptilnih problema koje automatizirani alati možda neće uočiti.

Posebno važan aspekt ručnog testiranja je praćenje logova. Kreiranje logova unutar koda i pravilno postavljanje razina logova (engl. *logging level*) omogućuju detaljan uvid u rad aplikacije. Logovi pružaju informacije o tome kako aplikacija funkcionira u stvarnom vremenu, bilježeći sve ključne događaje i potencijalne greške. To omogućava brzu identifikaciji i rješavanju problema, kao i u praćenju performansi aplikacije.

Korištenjem konzole preglednika (engl. *browser console log*) i naredbenog retka, omogućava praćenje i analizu ponašanje aplikacije u različitim uvjetima. Pravilno konfigurirani logovi i ispravno postavljene razine logova osiguravaju da su svi bitni podaci zabilježeni, bez preopterećenja sustava nepotrebnim informacijama. Ova praksa značajno doprinosi kvaliteti i pouzdanosti aplikacije.

Za pokretanje Cypress alata, potrebno je koristiti naredbu.Slika 18 prikazuje primjer ispisa logova nakon pokretanje automatiziranjog testiranja.

Slika 18. Testiranje aplikacije



Izvor: Autor

8. SIGURNOST APLIKACIJE

Sigurnost je bitna za sve vrste aplikacija. Pogotovo za poslovne budući da se radi o osjetljivim poslovnim podacima. Potrebno je osigurati da alati i tehnologije imaju snažne sigurnosne mjere, te da su u skladu s relevantnim industrijskim propisima i standardima kao što su enkripcija podataka i kontrola pristupa.

JHipster koristi Spring Security kao osnovni sigurnosni okvir, koji implementira standardne sigurnosne mjere kao što su autentifikacija i autorizacija korisnika. Uz to, JHipster automatski implementira i nekoliko dodatnih sigurnosnih mehanizama:

Heširanje (engl. hashing) lozinki, lozinke korisnika se ne pohranjuju u izvornom obliku, već se prije pohrane u bazu podataka prolaze kroz proces heširanja. JHipster koristi algoritam bcrypt za heširanje lozinki, čime se osigurava da su lozinke zaštićene čak i ako dođe do kompromitacije baze podataka.

Cross-Site Request Forgery Protection (engl. *CSRF*) zaštita sprječava neovlaštene radnje koje bi napadač mogao pokušati izvršiti u ime ovlaštenog korisnika. JHipster automatski uključuje CSRF zaštitu, koja štiti aplikaciju od ovih vrsta napada.

Clickjacking je napad u kojem se zlonamjerne web stranice pokušavaju prikazati u okviru unutar aplikacije kako bi prevarile korisnika da izvrši neželjene radnje. JHipster štiti aplikaciju od clickjackinga postavljanjem zaglavlja „X-Frame-Options“ koje onemogućuje učitavanje aplikacije u okviru aplikacije s druge domene drugoj domeni.

Content Security Policy (engl. *CSP*) je sigurnosna mjera koja pomaže u zaštiti od napada kao što su Cross-Site Scripting (engl. *XSS*) tako što kontrolira resurse koje stranica može učitati. JHipster omogućuje konfiguraciju CSP-a, što dodatno poboljšava sigurnost aplikacije.

Uz ove standardne sigurnosne značajke, u ovom projektu implementirane su i dodatne mjere, kao što su dvostruka validacija unosa podataka, koja provodi validaciju na strani klijenta i na strani poslužitelja, te slojevita arhitektura aplikacije (model, servis, repozitorij, kontroler) koja osigurava dodatnu zaštitu na svakom sloju aplikacije.

Sve ove sigurnosne mjere zajedno pružaju čvrst temelj za zaštitu aplikacije od različitih sigurnosnih prijetnji, osiguravajući da su podaci korisnika i funkcionalnosti aplikacije dobro zaštićeni.

9. ZAKLJUČAK

Sustav u kojem je moguće pratiti i imati pod kontrolom različite poslovne procese može biti izuzetno bitan poduzećima. Dobar informacijski sustav dovodi do poboljšanja produktivnosti, smanjenih pogrešaka i bolje koordinacije među odjelima. Nudi objedinjeni prikaz poslovnih aktivnosti. Takav pristup podacima omogućuje informirano donošenje odluka, jer na taj način voditelji mogu analizirati trendove podataka kako bi donijeli strateške odluke koje pokreću rast i profitabilnost tvrtke.

Cilj završnog rada bio je izraditi web jednostraničnu aplikaciju u oblaku za vođenje podataka o radu dimnjačarske službe na području grada. Ovaj cilj je uspješno ostvaren kroz razvoj aplikacije koja omogućuje vođenje podataka o građanima i obavljenom čišćenju prema planiranom kalendaru. Aplikacija koristi relacijsku bazu podataka MySQL i izrađena je u Spring Boot tehnologiji, uz korištenje Angulara za dinamičnost web aplikacije.

Ova struktura aplikacije omogućuje dodavanje funkcionalnosti kao što su implementacija naprednih analitičkih alata za bolje praćenje i analizu podataka, razvoj mobilne verzije aplikacije, te integracija s drugim sustavima za upravljanje poslovanjem kako bi se omogućila još bolja koordinacija i učinkovitost. Ove smjernice mogu pomoći u daljnjem razvoju i unapređenju aplikacije, čime će se dodatno poboljšati učinkovitost i kvaliteta usluga dimnjačarske službe.

Izjava o autorstvu

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
Bana Josipa Jelačića 22/a, Čakovec

IZJAVA O AUTORSTVU

Završni/diplomski rad isključivo je autorsko djelo studenta te student odgovara za istinitost, izvornost i ispravnost teksta rada. U radu se ne smiju koristiti dijelovi tuđih radova (knjiga, članaka, doktorskih disertacija, magistarskih radova, internetskih i drugih izvora) bez pravilnog citiranja. Dijelovi tuđih radova koji nisu pravilno citirani, smatraju se plagijatom i nezakonitim prisvajanjem tuđeg znanstvenog ili stručnoga rada. Sukladno navedenom studenti su dužni potpisati izjavu o autorstvu rada.

Ja, Antun Vedran Dajković (ime i prezime studenta) pod punom moralnom, materijalnom i kaznenom odgovornošću, izjavljujem da sam isključivi autor/ica završnog/diplomskog rada pod naslovom Cloud aplikacija E-dimljačar

te da u navedenom radu nisu na nedozvoljeni način (bez pravilnog citiranja) korišteni dijelovi tuđih radova.

Student/ica:

(vlastoručni potpis)

Literatura

1. About GitHub and Git, https://docs.github.com/en/get-started/start-your-journey/about-github-and-git, (Datum pristupa: Veljača. 2024.)
2. Bash documentation, https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html, (Datum pristupa: Veljača. 2024.)
3. Bernot, Matt. ERP Safety and Cybersecurity: What You Need To Know, https://www.striven.com/blog/erp-safety-and-cybersecurity-what-you-need-to-know, (Datum pristupa: Veljača. 2024.)
4. Create an application, https://www.jhipster.tech/creating-an-app/, (Datum pristupa: Veljača. 2024.)
5. Frontend VS Backend – What's the Difference? FreeCodeCamp, https://www.freecodecamp.org/news/frontend-vs-backend-whats-the-difference/, (Datum pristupa: Veljača. 2024.)
6. Introduction to the Angular Angular, https://angular.io/docs, (Datum pristupa: Veljača. 2024.)
7. Java documentation Oracle, https://docs.oracle.com/en/java/index.html, (Datum pristupa: Veljača. 2024.)
8. JHipster Domain Language (JDL), https://www.jhipster.tech/jdl/intro, (Datum pristupa: Veljača. 2024.)
9. Spring Boot Reference Documentation SpringBoot, https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/, (Datum pristupa: Veljača. 2024.)
10. SQL documentation, https://docs.oracle.com/cd/E11882_01/server.112/e41084/toc.htm, (Datum pristupa: Veljača. 2024.)
11. Visual Studio documentation, https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022, (Datum pristupa: Veljača. 2024.)
12. Web vs desktop apps: a weigh-up, https://www.parkersoftware.com/blog/web-vs-desktop-apps-a-weigh-up/, (Datum pristupa: Veljača. 2024.)
13. What Is A Programming Language? Codeinstitute, https://codeinstitute.net/global/blog/what-is-a-programming-language/, (Datum pristupa: Veljača. 2024.)
14. Why cypress, https://docs.cypress.io/guides/overview/why-cypress,(Datum pristupa: Veljača. 2024.)
15. XAMPP documentation, https://www.apachefriends.org/docs/, (Datum pristupa: Veljača. 2024.)

Popis ilustracija