# Projektovanje i implementacija softverskog rešenja za raspodelu nastavnih aktivnosti

Ivan Cicka, dr Bojana B. Dimić Surla

***Sadržaj* — Raspodela nastavnih aktivnosti predstavlja važan aspekt organizacije rada obrazovnih ustanova i zahteva sistematičan pristup. Ovaj rad prikazuje razvoj veb aplikacije koja automatizuje proces dodele nastavnika predmetima. Rešenje je implementirano korišćenjem Angular i Spring Boot okvira, uz skalabilnu arhitekturu. Sistem omogućava upravljanje podacima, kreiranje raspodela, izvoz u JSON i PDF formate, te efikasnu razmenu informacija putem REST servisa i DTO objekata.**

***Ključne reči* — raspodela nastavnih aktivnosti, veb aplikacija, Angular, Spring Boot, generisanje PDF.**

## I. UVOD

U savremenom obrazovnom sistemu, jedan od ključnih administrativnih zadataka je raspodela nastavnika na predmete. Mnoge škole i obrazovne ustanove još uvek obavljaju raspodelu ručno, što može biti dugotrajan i složen proces. Takođe, neke škole koriste zastarele ili nepouzdane softverske alate, koji ne obezbeđuju tačan obračun nastavnih obaveza. Ovakvi problemi mogu dovesti do neravnomerne raspodele radnog opterećenja među nastavnicima, otežati administrativni rad pa čak i dovesti do netačnog obračuna zarada.

Ovaj rad se bavi razvojem veb aplikacije otvorenog koda namenjene školama i obrazovnim ustanovama, koja omogućava jednostavnu i efikasnu raspodelu nastavnika na predmete. Aplikacija je osmišljena tako da omogući administratorima unos i upravljanje nastavnim obavezama, dok nastavnici mogu pregledati dodeljene predmete i broj časova po nedeljama, semestrima i ukupno za celu školsku godinu. Iako aplikacija ne vrši direktan obračun plata, ona omogućava precizno praćenje broja časova, što može biti osnova za dalji obračun zarade.

Aplikacija je prvenstveno namenjena visokoškolskim ustanovama, ali se uz određene izmene može uspešno primenjivati i u srednjim, pa čak i u osnovnim školama. Njen prilagodljiv dizajn omogućava lako usklađivanje sa specifičnim potrebama različitih obrazovnih ustanova. Za razliku od zatvorenih softverskih rešenja, ova

I. Cicka. Autor, Računarski fakultet, Beograd, Srbija (email: icicka2524m@raf.rs@raf.rs)
B. Dimić Surla. Autor, Računarski fakultet, Beograd, Srbija; (email: bdimicsurla@raf.rs)

aplikacija je otvorenog koda i razvijena pod slobodnom softverskom licencom (MIT), što omogućava obrazovnim ustanovama da je slobodno koriste, prilagođavaju i unapređuju njen kod.

Cilj ovog rada je da prikaže proces razvoja aplikacije, ključne tehničke odluke i način na koji ona rešava probleme u raspodeli nastave Pored tehničkih aspekata, rad ističe i potencijal aplikacije za unapređenje organizacije nastavnog procesa u obrazovnim institucijama.

## II. KREIRANJE I UPRAVLJANJE RASPODELOM NASTAVE

Raspodela nastave predstavlja proces dodeljivanja nastavnih predmeta odgovarajućim profesorima, uz precizno definisanje obima i tipa nastavnih aktivnosti. Sistem za upravljanje raspodelom nastave organizovan je u dva osnovna segmenta: predmeti sa već definisanom raspodelom i predmeti koji tek treba da budu dodeljeni profesorima. Raspodela se pritom oslanja na fond časova predavanja i vežbi koji su definisani akreditacijom studijskog programa, čime se obezbeđuje usklađenost sa zakonskim i akademskim normama.

### *A. Upravljanje postojećim raspodelama*

Za predmete koji su već dodeljeni, sistem omogućava sledeće funkcionalnosti:

- **Promena profesora dodeljenog predmetu** – Administrator može izvršiti zamenu profesora zaduženog za izvođenje nastave na određenom predmetu, u skladu sa potrebama ustanove.
- **Ažuriranje broja termina** – Sistem omogućava prilagođavanje broja termina predavanja ili vežbi koji su dodeljeni profesoru za određeni predmet.
- **Definisanje vrste nastave** – Za svaki predmet moguće je jasno naznačiti da li se radi o predavanjima ili vežbama, čime se postiže precizna organizacija nastavnog procesa.

### *B. Dodela preostalih predmeta profesorima*

Za predmete koji još uvek nisu raspodeljeni, sistem obezbeđuje sledeće korake:

- **Pregled predmeta bez dodele** – Administrator ima uvid u listu svih predmeta kojima još nije dodeljen profesor, uz opcije pretrage i filtriranja.
- **Dodela profesora predmetu** – Administrator selektuje profesora koji će biti zadužen za izvođenje nastave na izabranom predmetu, uzimajući u obzir stručnost i dostupnost.
- **Definisanje broja termina** – Nakon dodele profesora, administrator određuje broj termina predavanja ili vežbi koje će profesor realizovati u okviru predmeta.

### C. Generisanje izveštaja

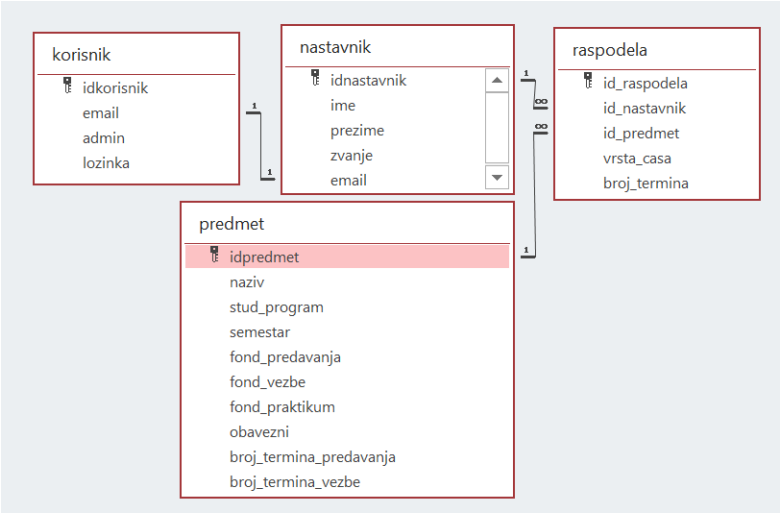
Sistem obezbeđuje mogućnost generisanja izveštaja u PDF formatu, kao i izvoza podataka u JSON formatu, čime se olakšava analiza raspodele nastave i omogućava integracija sa drugim informacionim sistemima fakulteta.

### D. Model baze podataka

Model baze podataka razvijen je sa ciljem da omogući efikasno funkcionisanje aplikacije, uz jasno definisane odnose između ključnih entiteta sistema. Posebna pažnja posvećena je eliminaciji suvišnog (redundantnog) skladištenja podataka, čime se obezbeđuje integritet i optimizacija performansi baze. Struktura baze podataka zasnovana je na principima relacionog modela, a vizuelna reprezentacija međusobnih veza između entiteta prikazana je putem ER (Entity-Relationship) dijagrama.

### E. ER dijagram

Struktura podataka i odnosi među entitetima sistema za raspodelu nastave prikazani su na slici 1, koja prikazuje ER dijagram sistema.




Sl 1. ER dijagram: Struktura baze podataka i odnosi između entiteta

ER dijagram prikazuje četiri osnovna entiteta: Korisnik, Nastavnik, Raspodela i Predmet. Entitet Korisnik povezan je sa entitetom Nastavnik odnosom jedan na jedan (1:1), pri čemu svaki korisnik odgovara tačno jednom nastavniku, povezan preko identifikatora korisnika i email adrese. Entitet Nastavnik povezan je sa entitetom Raspodela odnosom jedan prema više (1:N), omogućavajući da jedan nastavnik učestvuje u više raspodela. Slično tome, Predmet je sa entitetom Raspodela povezan takođe odnosom jedan prema više (1:N), omogućavajući da jedan predmet bude dodeljen u više različitih raspodela. Na ovaj način, model baze omogućava jasno definisane veze između korisnika, nastavnika, predmeta i njihovih raspodela.

## III. IMPLEMENTACIJA APLIKACIJE

U ovom poglavlju detaljno je prikazan proces razvoja aplikacije, sa posebnim osvrtom na korišćene tehnologije, arhitekturu sistema, ključne funkcionalnosti i izazove sa kojima se susretalo tokom implementacije. Akcenat je stavljen na tehničke odluke donete u toku razvoja, kao i na način međusobnog povezivanja različitih komponenti sistema.

Prvi deo poglavlja posvećen je izboru tehnologija korišćenih prilikom implementacije, uz obrazloženje razloga za njihov odabir i prednosti koje pružaju u kontekstu razvoja konkretne aplikacije. U nastavku se analizira arhitektura sistema, sa fokusom na način komunikacije između korisničkog dela aplikacije i serverskog dela aplikacije, koja se realizuje korišćenjem REST API-ja. Sledeći segmenti poglavlja obuhvataju detaljan opis ključnih funkcionalnosti sistema, kao i izazove koji su se pojavili tokom procesa razvoja, zajedno sa metodama i rešenjima koja su primenjena radi njihovog prevazilaženja.

### *A. Korišćene tehnologije*

Pri izboru tehnologija za razvoj aplikacije, uzeti su u obzir savremeni industrijski standardi, zahtevi tržišta, kao i prethodna praktična iskustva u radu sa različitim alatima i programskim okvirima. Pored tehničkih karakteristika, posebna pažnja posvećena je i mogućnostima dugoročne održivosti i skalabilnosti sistema.

Za implementaciju serverskog dela aplikacije (pozadinskog sistema), izabran je Spring Boot, jedan od najzastupljenijih programskih okvira u industriji razvoja softvera. Spring Boot predstavlja proširenje popularnog Spring okvira, sa ciljem da pojednostavi konfiguraciju i ubrza razvoj aplikacija. Njegove prednosti uključuju jednostavnu integraciju sa bazama podataka, lako povezivanje sa eksternim API servisima i mogućnost razvoja složenih, modularnih i skalabilnih poslovnih sistema. Posebno je značajna njegova podrška za automatizovano konfigurisanje i bogat ekosistem dodatnih biblioteka, što omogućava efikasniju implementaciju kompleksnih zahteva.

Za implementaciju korisničkog dela aplikacije (korisničkog interfejsa) odabran je Angular, jedan od vodećih okvira za razvoj dinamičkih veb aplikacija. Angular, razvijen od strane kompanije Google, zasnovan je na jeziku TypeScript, što obezbeđuje jaku tipizaciju, unapređenu čitljivost koda i smanjenu mogućnost grešaka pri razvoju. Ovaj okvir nudi modularnu arhitekturu i efikasan mehanizam za upravljanje stanjima komponenata, što značajno doprinosi razvoju skalabilnih i održivih korisničkih

interfejsa. Pored toga, struktura koju Angular nameće omogućava jasnu organizaciju projektnog koda, što dodatno olakšava buduće održavanje i nadogradnju aplikacije.

Odluka o izboru navedenih tehnologija nije doneta nasumično, već je zasnovana na detaljnoj analizi njihovih prednosti u kontekstu specifičnih zahteva aplikacije. Cilj primene ovih rešenja bio je izgradnja moderne, skalabilne i dobro organizovane veb aplikacije, u skladu sa savremenim principima razvoja softverskih sistema.

### *B. Implementacija serverskog sistema*

Serverski deo aplikacije razvijen je korišćenjem programskog okvira Spring Boot, koji omogućava ubrzani razvoj veb sistema na programskom jeziku Java. Inicijalna postavka projekta izvršena je putem alata Spring Initializer, čime je automatski generisana osnovna struktura aplikacije sa definisanim zavisnostima neophodnim za dalji razvoj.

### *C. Arhitektura serverskog sistema*

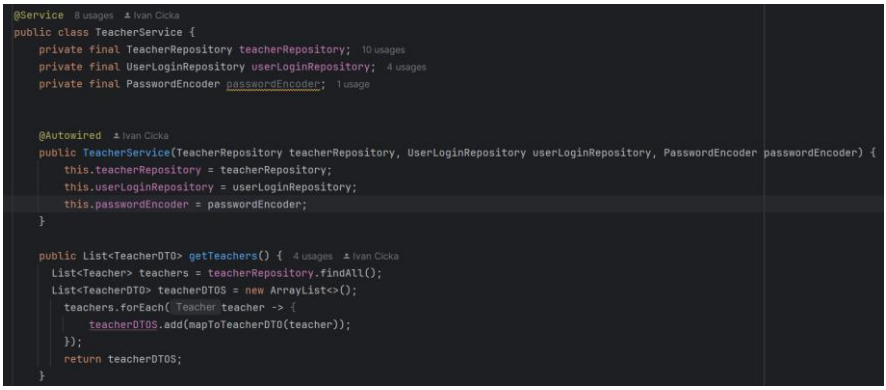
Aplikacija je organizovana u više slojeva, kako bi se obezbedila modularnost i lako održavanje koda. Struktura aplikacije podeljena je na sledeće komponente:

Kontroleri predstavljaju ulaznu tačku za sve HTTP zahteve koje aplikacija prima. Oni definišu REST API krajnje tačke i upravljaju usmeravanjem zahteva ka odgovarajućim servisnim metodama. Kontroleri su fokusirani isključivo na komunikaciju putem HTTP protokola, dok izvršavanje poslovne logike prepuštaju servisima. Primer implementacije kontrolera prikazan je na slici 2.

```
@RestController  Ivan Cicka
@RequestMapping(path = "/api/teachers")
@Tag(name = "Teacher API", description = "API for managing teachers")
public class TeacherController {

    private final TeacherService teacherService;  7 usages

    @Autowired  Ivan Cicka
    public TeacherController(TeacherService teacherService) { this.teacherService = teacherService; }

    @GetMapping  Ivan Cicka
    @Operation(summary = "Get all teachers", description = "Retrieve a list of all teachers")
    public ResponseEntity<List<TeacherDTO>> teachers() {
        List<TeacherDTO> teachers = teacherService.getTeachers();
        return ResponseEntity.status(HttpStatus.OK).body(teachers);
    }
```

Sl 1. Kontroler za profesore

Servisi implementiraju poslovnu logiku aplikacije i čine posrednički sloj između kontrolera i pristupa podacima. Na ovaj način se postiže jasno razdvajanje odgovornosti. Kontroleri ostaju jednostavni i orijentisani na obradu zahteva i odgovora, dok servisi obavljaju složenije operacije i pravila poslovanja. Primer implementacije servisa prikazan je na slici 3.




Sl 2. Servis za profesore

Modeli (entiteti) predstavljaju programske prikaze podataka koji se čuvaju u bazi. Svaki model opisuje strukturu entiteta kroz atribute i njihove tipove, kao i relacije među entitetima. Primer implementacije modela prikazan je na slici 4.

Repozitorijumi predstavljaju sloj za pristup podacima i realizovani su korišćenjem Spring Data JPA mehanizama. Njihova upotreba omogućava izvođenje standardnih operacija nad bazom podataka, kao što su čuvanje, ažuriranje, brisanje i pretraga entiteta, bez potrebe za pisanjem manuelnih SQL upita. Time se proces razvoja dodatno pojednostavljuje, a čitljivost i održavanje koda se značajno unapređuju. Primer implementacije repoziturijuma prikazan je na slici 5.

```java
@Entity  Ivan Cicka
@Table(name = "nastavnik")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Teacher {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "idnastavnik")
    private Long id;

    @Column(name = "ime")
    private String firstName;

    @Column(name = "prezime")
    private String lastName;

    @Column(name = "zvanje")
    private String title;

    @OneToOne
    @JoinColumn(name = "email", referencedColumnName = "email")
    private UserLogin userLogin;

    @OneToMany(mappedBy = "teacher", cascade = CascadeType.ALL)
    private List<Distribution> distributions;

}
```

Sl 3. Model profesora

```java
@RepositoryRestResource(exported = false)  Ivan Cicka
public interface TeacherRepository extends JpaRepository<Teacher, Long> {

    @Query("SELECT t FROM Teacher t WHERE t.firstName = ?1")  no usages  Ivan Cicka
    Optional<Teacher> findByName(String name);

    @Query("SELECT u FROM UserLogin u WHERE u.email = ?1")  3 usages  Ivan Cicka
    Optional<UserLogin> findByEmail(String email);

    @Query("SELECT t FROM Teacher t WHERE t.title = ?1")  2 usages  Ivan Cicka
    List<Teacher> findByType(String type);
}
```

Sl 4. Repozitorijum za profesore

### *D. Rad sa bazom podataka*

Povezivanje aplikacije sa sistemom za upravljanje bazama podataka realizovano je korišćenjem JDBC drajvera, pri čemu su sve neophodne konfiguracije definisane u fajlu *application.properties*. Ove konfiguracije obuhvataju naziv aplikacije, URL baze

podataka, korisničko ime, lozinku, kao i dodatne parametre potrebne za uspešno uspostavljanje konekcije.

Za potrebe ovog projekta odabrana je MySQL baza podataka. Odluka o izboru MySQL-a zasnovana je na više faktora. Pre svega, tokom studija stečeno je značajno iskustvo u radu sa ovim sistemom, što je omogućilo bržu i efikasniju implementaciju rešenja. Dodatno, budući da je profesorka dostavila početnu verziju baze u MySQL formatu, ovaj izbor se pokazao kao najprirodnije i najpraktičnije rešenje za nastavak razvoja.

Pored praktičnih razloga, MySQL se ističe svojom jednostavnošću upotrebe i lakim procesom učenja, zbog čega je među najčešće korišćenim rešenjima u industriji. Njegova široka primena, robusna dokumentacija i podrška zajednice dodatno su potvrdili opravdanost ove tehničke odluke. Primer povezivanja serverskog dela aplikacije sa bazom prikazan je na slici 6.

```
spring.application.name=eScheduler

spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:${MYSQL_PORT:2524}/${MYSQL_DB_NAME:raspodelanastave}
spring.datasource.username=${MYSQL_USER:root}
spring.datasource.password=${MYSQL_PASSWORD:Root.2021}
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql= true
server.port=2525
```

Sl 5. Povezivanje serveskog dela aplikacije sa MySQL bazom podataka

### *E. Konfiguracija Docker Compose okruženja*

Radi lakšeg postavljanja i pokretanja aplikacije u različitim radnim okruženjima, izrađen je konfiguracioni fajl za Docker Compose. Ovaj fajl definiše način pokretanja aplikacije u okviru Docker okruženja, uključujući konfiguraciju usluga, mrežna podešavanja i povezivanje sa sistemom za upravljanje bazama podataka. Na slici 7 prikazana je konfiguracija Docker Compose okruženja.

```yaml
version: '3.8'

services:
  database:
    image: mysql:5.7
    container_name: mysql_container
    environment:
      MYSQL_ROOT_PASSWORD: Root.2021
      MYSQL_DATABASE: raspodelanastave
    ports:
      - "2524:3306" # Mapira lokalni port 2524 na 3306 unutar kontejnera
    networks:
      - escheduler_network
    volumes:
      - db_data:/var/lib/mysql

  backend:
    image: ivancicka652/escheduler:latest_backend
    container_name: backend-container
    ports:
      - "2525:2525" # Mapira port 2525 za Spring Boot aplikaciju
    environment:
      - SPRING_DATASOURCE_URL=jdbc:mysql://database:3306/raspodelanastave
      - SPRING_DATASOURCE_USERNAME=root
      - SPRING_DATASOURCE_PASSWORD=Root.2021
    networks:
      - escheduler_network
    depends_on:
      - database

  frontend:
    image: ivancicka652/escheduler:latest_front
    container_name: frontend-container
    ports:
      - "2526:80" # Mapira port 2526 lokalno na 80 (Nginx default port) u kontejneru
    depends_on:
      - backend
    networks:
      - escheduler_network

networks:
  escheduler_network:
    driver: bridge

volumes:
  db_data:
```

Sl 6. Konfiguracija Docker Compose okruženja

### *F. Postavljanje baze podataka unutar Docker mreže*

Prvobitna zamisao bila je da se aplikacija poveže sa postojećom MySQL bazom podataka koja je već bila instalirana na školskom serveru. Međutim, zbog neuspešnih pokušaja uspostavljanja stabilne veze, zaključeno je da je pouzdanije i jednostavnije postaviti zasebnu instancu baze podataka unutar Docker mreže. Na taj način, svi delovi sistema, korisnički interfejs, serverska logika i sistem za upravljanje bazom podataka, mogu međusobno komunicirati putem interno definisane Docker mreže, čime se

izbegavaju poteškoće povezane sa spoljnim konekcijama i složenim podešavanjima pristupnih dozvola.

Radi postizanja potpune funkcionalnosti sistema, svi kontejneri su povezani na zajedničku mrežu u okviru Docker okruženja, čime je omogućena njihova međusobna komunikacija. Serverski deo sistema je dodatno konfigurisan da se povezuje direktno sa instancom baze podataka unutar mreže, koristeći propisno definisane pristupne podatke i parametre u okviru konfiguracionih datoteka aplikacije.

Automatizacija procesa postavljanja aplikacije korišćenjem Docker skladišta i GitHub automatizacije

Radi obezbeđivanja jednostavnog i efikasnog načina za postavljanje aplikacije na školski server, sprovedena je automatizacija procesa postavljanja korišćenjem kombinacije GitHub-ove funkcionalnosti za automatizaciju zadataka (*GitHub Actions*) i skladišta kontejnerskih slika (*Docker Hub*). Ovakav sistem omogućava da se svaka izmena u izvornom kodu automatski integriše i pripremi za objavljivanje, čime se eliminiše potreba za ručnim uplitanjem i ubrzava proces ažuriranja softverskog rešenja.

Kada programer unese izmene u kod i postavi ih na GitHub skladište, automatski se pokreće unapred definisan tok izvršavanja zadataka (*workflow*), koji prepoznaje da je došlo do promene. Sistem potom samostalno generiše nove kontejnerske slike (jednu za korisnički interfejs i jednu za serverski deo aplikacije). Ove slike nisu lokalne, odmah po kreiranju, šalju se na udaljeno skladište (*Docker Hub*), čime postaju dostupne za preuzimanje i pokretanje na bilo kom serveru.

Na školskom serveru se koristi *Docker Compose* konfiguracija kojom se, prilikom pokretanja, preuzimaju najnovije verzije kontejnerskih slika i aplikacija se pokreće u ažuriranom stanju.

### *G. Ograničenja trenutnog rešenja*

Iako se nove verzije aplikacije automatski objavljuju i postavljaju u Docker skladište, na školskom serveru ažuriranje zahteva ručnu intervenciju. Administrator mora zaustaviti postojeće kontejnere, obrisati zastarele slike i ponovo pokrenuti Docker Compose. Tek tada se nove slike preuzimaju i aplikacija pokreće u osveženom stanju. Ova potreba za ručnim održavanjem može usporiti proces ažuriranja i predstavlja ograničenje trenutne postavke.

### *H. Problemi sa rutama i unapred definisanim URL adresama*

Početna verzija aplikacije za korisnički interfejs bila je razvijena sa statički definisanim URL adresama za slanje HTTP zahteva prema sistemu za podršku (backend). Tokom lokalnog razvoja, aplikacija je koristila jednu definisanu rutu za komunikaciju sa serverom, što je u početku delovalo kao funkcionalno rešenje. Međutim, problem se pojavio kada je aplikacija trebalo da se pokrene u Docker okruženju i postavi na server.

Prilikom promene portova i konfiguracija u Docker okruženju, aplikacija više nije mogla da komunicira sa sistemom za podršku, jer su URL adrese bile statički podešene na određeni port i domenski naziv. To je dovelo do situacije u kojoj je aplikacija za korisnički interfejs bila u stanju da se renderuje i pokrene, ali nije uspevala da izvrši API zahteve prema sistemu za podršku, što je rezultiralo greškama u komunikaciji.

Kako bi se ovaj problem prevazišao, implementiran je mehanizam korišćenja promenljivih okruženja (environment variables) za definisanje API ruta. Kreirane su dve posebne konfiguracione datoteke:

Jedna za lokalni razvoj, koja sadrži URL adresu sistema za podršku dostupnog na lokalnom računaru.

Druga za produkciono okruženje, koja sadrži URL adresu sistema za podršku nakon što je aplikacija postavljena na server.

Na ovaj način, aplikacija za korisnički interfejs je mogla da koristi odgovarajuće rute u zavisnosti od okruženja u kojem se pokreće. Ovaj pristup omogućio je fleksibilnost u razvoju, testiranju i produkcionom radu, bez potrebe za ručnim izmenama koda prilikom prelaska iz jednog okruženja u drugo. Na slici 8 prikazano je korišćenje produkcionog okruženja putem fajla *environment.prod.ts*.

```
"configurations": {
  "production": {
    "fileReplacements": [
      {
        "replace": "src/environments/environment.ts",
        "with": "src/environments/environment.prod.ts"
      }
    ],
```

Sl 7. Konfiguracija produkcionog okruženja u fajlu environment.prod.ts

### *I. Problemi sa konfiguracijom Nginx servera*

Nakon uspešnog kontejnerisanja i pokretanja korisničkog interfejsa aplikacije u Docker okruženju, pojavio se novi problem vezan za način na koji veb server Nginx isporučuje sadržaj aplikacije. U početnoj fazi, kada bi korisnik pristupio početnoj stranici aplikacije, sve je funkcionisalo ispravno. Međutim, do poteškoća je dolazilo prilikom pokušaja direktnog pristupanja određenim adresama, kao što je na primer /login.

Kada bi korisnik direktno uneo adresu /login u adresnu traku internet pregledača ili osvežio stranicu dok je već bio na toj adresi, aplikacija bi prestala da funkcioniše i prikazivala bi poruku o grešci. Uzrok problema bio je u tome što Nginx nije znao kako da obradi zahteve za adresama koje nisu bile eksplicitno definisane u njegovoj konfiguraciji. Budući da je aplikacija razvijena kao jednostranička veb aplikacija (eng. *Single Page Application*), kompletna navigacija se odvija unutar same aplikacije, a ne putem klasičnih serverskih ruta.

Problem je rešen izmenom konfiguracije Nginx servera, tako da svi HTTP zahtevi koji ne odgovaraju statičkim datotekama budu automatski preusmereni na početnu HTML stranicu (*index.html*). Ovim podešavanjem omogućeno je da sama aplikacija

preuzme upravljanje rutama i na taj način ispravno prikaže sve potrebne prikaze korisničkog interfejsa.

Na taj način, bez obzira na to koju adresu korisnik unese, Nginx isporučuje početnu stranicu aplikacije, a aplikacija zatim, na osnovu interne logike, prikazuje odgovarajući deo sadržaja. Na slici 9 prikazano je rešenje problema sa rutiranjem u Nginx konfiguraciji, koje omogućava pravilno prikazivanje Angular aplikacije u slučaju nepostojeće rute.

```
location / {
    try_files $uri /index.html;
}

error_page 404 /index.html;
```

Sl 8. Nginx konfiguracija za rešavanje problema sa rutiranjem

## IV. GRAFIČKI PRIKAZ I OPIS KORISNIČKOG INTERFEJSA

U ovom poglavlju prikazan je detaljan pregled korisničkog interfejsa aplikacije, uz opis funkcionalnosti pojedinačnih stranica i grafički prikaz njihovog izgleda. Cilj poglavlja jeste da se predstavi način na koji je korisnički interfejs razvijan, kao i ključne funkcije koje aplikacija omogućava krajnjem korisniku.

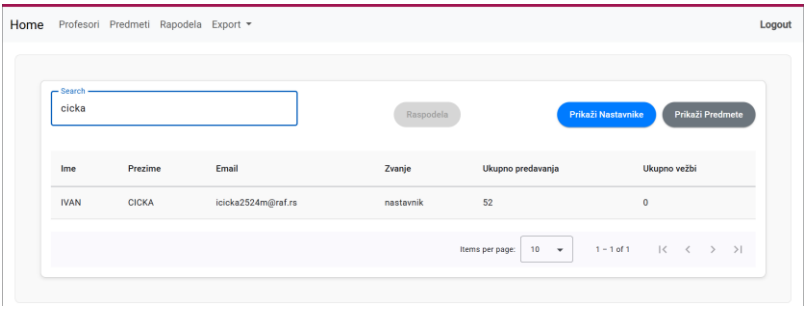
### *A. Početna stranica*

Početna stranica predstavlja centralni deo aplikacije i služi za pregled raspodele nastavnih obaveza. Ova stranica omogućava korisnicima da na intuitivan način prate i analiziraju podatke o nastavnim angažmanima kroz dva različita prikaza: po nastavnicima i po predmetima.

Po podrazumevanim postavkama, pri učitavanju aplikacije prikazuje se raspodela nastavnih obaveza po nastavnicima. Korisnici mogu promeniti prikaz klikom na jedno od dva dugmeta smeštena u gornjem desnom uglu tabele:

- **Prikaži nastavnike** – prikazuje podatke sortirane po nastavnicima
- **Prikaži predmete** – prikazuje podatke sortirane po predmetima

Ova funkcionalnost omogućava lako prebacivanje između dve različite perspektive pregleda raspodele časova. Na slici 10 prikazan je izgled početne stranice aplikacije.




Sl 9. Početna stranica

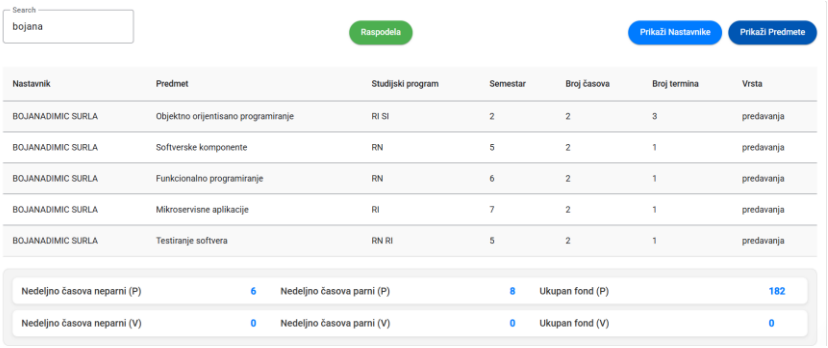
### *B. Prikaz raspodele po nastavnicima*

U ovom režimu rada, tabela prikazuje listu svih nastavnika sa osnovnim podacima o njihovim angažovanjima. Klikom na određeni red u tabeli korisnik dobija detaljan pregled predmeta koje taj nastavnik predaje, kao i obračun njegovog ukupnog nastavnog opterećenja.

Za svakog nastavnika se prikazuju sledeći podaci:

- Lista predmeta koje predaje
- Broj časova u parnom i neparnom semestru, posebno za predavanja i vežbe
- Ukupni fond nastavnih časova, kao zbir svih angažmana

Ovaj prikaz omogućava administratorima da brzo sagledaju opterećenje svakog pojedinačnog nastavnika i donesu informisane odluke o eventualnim preraspodelama. Detaljan prikaz raspodele opterećenja po nastavnicima dat je na slici 11.




Sl 10. Detaljan prikaz raspodele po nastavnicima

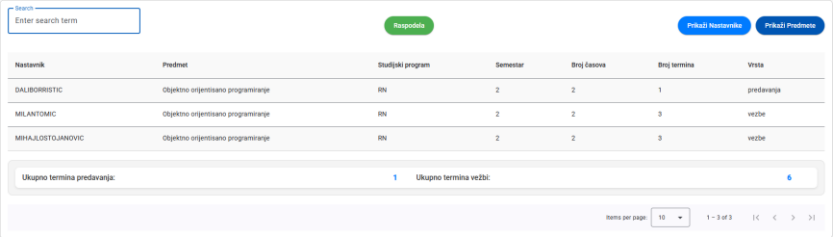
### C. Prikaz raspodele po predmetima

Klikom na dugme "Prikaži predmete", korisnik menja režim pregleda i dobija tabelu sa podacima organizovanim po predmetima. Svaki red u tabeli predstavlja jedan predmet, a klikom na njega prikazuje se spisak nastavnika angažovanih na tom predmetu, uz broj termina za predavanja i vežbe.

Za svaki predmet prikazani su sledeći podaci:

- Listu svih nastavnika koji su angažovani na tom predmetu
- Ukupan broj termina predavanja
- Ukupan broj termina vežbi

Ovaj prikaz omogućava korisnicima da analiziraju raspodelu nastavnih obaveza sa aspekta predmeta, što je korisno pri optimizaciji nastavnih resursa i planiranju akademskih obaveza. Detaljan prikaz nastavnih obaveza po predmetima dat je na slici 12.




Sl 11. Pregled nastavnih obaveza po predmetima

## V. ZAKLJUČAK I BUDUĆI RAD

Ovaj rad se bavi razvojem aplikacije za školstvo koja omogućava jednostavno kreiranje raspodele časova i olakšava izračunavanje plata profesora. Glavni cilj sistema jeste da unapredi rad školskih administratora i osoblja zaduženog za organizaciju nastave, smanjujući manuelni rad i mogućnost grešaka. Time se postiže veća efikasnost, a korisnici dobijaju intuitivno rešenje koje doprinosi boljoj organizaciji nastavnog procesa.

Sistem je u potpunosti razvijen u skladu sa unapred definisanim zahtevima, čime je potvrđena njegova funkcionalna ispravnost i primenljivost u stvarnom okruženju. Tokom razvoja javljali su se različiti izazovi, ali su oni uspešno prevaziđeni korišćenjem dostupnih resursa, uključujući obrazovne sadržaje na internetu i savremene tehnologije zasnovane na veštačkoj inteligenciji. Ovaj proces ukazuje na značaj pravilne upotrebe savremenih tehnoloških alata, koji značajno ubrzavaju razvoj i poboljšavaju kvalitet softverskih rešenja.

Ključne prednosti razvijenog sistema ogledaju se u njegovoj praktičnosti i jednostavnosti korišćenja. Administratori mogu brzo i lako kreirati raspodelu časova, profesori imaju bolji pregled svojih obaveza, a broj potencijalnih grešaka u planiranju

se smanjuje. Time se postiže bolja organizacija rada, što doprinosi efikasnijem funkcionisanju obrazovnih institucija.

Iako aplikacija u svojoj trenutnoj verziji nudi značajne mogućnosti, prostor za njeno dalje unapređenje i proširenje funkcionalnosti svakako postoji. Jedan od pravaca budućeg razvoja jeste prilagođavanje sistema različitim vrstama obrazovnih ustanova. Trenutno je aplikacija u većoj meri usmerena ka visokoškolskim ustanovama, ali bi buduće verzije mogle omogućiti korisnicima da biraju između rešenja namenjenog osnovnom, srednjem ili visokom obrazovanju.Dodatno, planira se implementacija funkcionalnosti automatskog generisanja rasporeda časova. Ova mogućnost bi značajno rasteretila administratore, jer bi omogućila optimizaciju rasporeda na osnovu dostupnosti nastavnika, slobodnih učionica i drugih relevantnih parametara. Ipak, realizacija ove funkcionalnosti zahteva dodatno istraživanje i primenu složenih algoritama za generisanje optimalnih rasporeda. U tom kontekstu, posebno se izdvajaju genetski algoritmi kao efikasno rešenje za rešavanje kompleksnih problema raspoređivanja. Ovi algoritmi su se pokazali uspešnim u generisanju rasporeda koji zadovoljavaju višestruka ograničenja i preferencije, uključujući raspoloživost resursa i pravila obrazovnih ustanova [5].

Kao još jedan korak u pravcu unapređenja sistema, predviđena je integracija sa spoljnim informacionim sistemima, kao što su elektronski dnevnik i druga softverska rešenja koja se koriste u obrazovnim ustanovama. Na taj način bi se povećala međusobna povezanost sistema i omogućila bolja razmena podataka između različitih digitalnih platformi. Sve navedene nadogradnje predstavljaju korake ka sveobuhvatnom, fleksibilnom i prilagodljivom rešenju za efikasno upravljanje nastavnim procesom u obrazovnim ustanovama.

## LITERATURA


[1] D. Speck, "Workload is creating ‘high stress’, say 80% of teachers," *Tes Magazine*, 28. oktobar 2019. [Online]. Available: https://www.tes.com/magazine/archive/workload-creating-high-stress-say-80-teachers

[2] A. Brown, "The Failings of Excel in the Education Sector," *SAAF Education Blog*, SAAF Education, 7. jun 2017. [Online]. Available: https://www.saafeducation.org/blog/failings-of-

[3] Benjamin, "7 Reasons Why Your School Needs Automated Scheduling," *Picktime Blog*. [Online]. Available: https://blog.picktime.com/school-needs-automated-scheduling [Accessed: May 4, 2025].

[4] Anna Firer, Anatoly Kopeikin, Igor Khramov, Features of automation of the university schedule management process through the "Class Schedule WEB application", ITM Web Conf. Volume 72, 2025, III International Workshop on “Hybrid Methods of Modeling and Optimization in Complex Systems” (HMMOCS-III 2024)

[5] Yu Chen, Mahmonir Bayanati, Maryam Ebrahimi, Sadaf Khalijian, A Novel Optimization Approach for Educational Class Scheduling with considering the Students and Teachers’ Preferences, Discrete Dynamics in Nature and SocietyVolume 2022, Article ID 5505631, 11 pages

[6] Angular Team, *Angular – Overview*, Angular Documentation, [Online]. Available: https://angular.io/docs [Accessed: May 15, 2025].
[7] C. Walls, *Spring Boot in Action*, Shelter Island, NY, USA: Manning Publications, 2019.
[8] Spring Boot Team, *Spring Boot Documentation*, [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/html/ [Accessed: May 15, 2025].
[9] Oracle Corporation, *MySQL 8.0 Reference Manual*, [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/ [Accessed: May 15, 2025].
[10] S. Bechtold, S. Brannen, J. Link, M. Merdes, M. Philipp, and C. Stein, *JUnit 5 User Guide*, Version 5.0.2. [Online]. Available: https://junit.org/junit5/docs/5.0.2/user-guide/index.pdf.
[11] T. Erl and E. B. Monroy, Računarstvo u oblaku: Koncepti, tehnologija, bezbednost I arhitektura. Beograd, Srbija: Kompjuter biblioteka, 2024
[12] S. Bechtold, S. Brannen, J. Link, M. Merdes, M. Philipp, and C. Stein, *JUnit 5 User Guide*, Version 5.0.2. [Online]. Available: https://junit.org/junit5/docs/5.0.2/user-guide/index.pdf.

## ABSTRACT

Abstract — The distribution of teaching activities is a key process in the organization of educational institutions, requiring a clear and systematic method for assigning subjects to teachers. This master's thesis presents the design and implementation of a web-based software solution that supports and automates the distribution process in schools. The application is developed using Angular for the frontend and Spring Boot for the backend, providing a scalable and maintainable architecture.

The system allows users to manage teachers and subjects, create distributions, export data in JSON format, and generate PDF documents for administrative and payroll purposes. The backend is based on RESTful API principles and uses DTO objects for structured and efficient data transfer, while the frontend offers a user-friendly interface for managing distributions.

**Design and implementation of a software solution for the distribution of teaching activities**

Ivan Cicka, dr Bojana B. Dimić Surla