Tehničko veleučilište u Zagrebu  
Prijediplomski studij računalstva

**IZRADA WEB SUSTAVA ZA ADMINISTRACIJU AKADEMSKE INSTITUCIJE U  
PROGRAMSKOM JEZIKU JAVA**

Seminarski rad

Kolegij: Metodologija stručnog i istraživačkog rada  
Ime i prezime nastavnika (nositelj kolegija): Sara Slamić  
Ime i prezime studenta: Tim Pavić

Zagreb, siječanj 2025.

# Sažetak

U ovom radu biti će prikazan i opisan proces izrade web aplikacije za administraciju akademskog sustava u programskom okviru Spring Framework koji proširuje programski jezik Java. Rad je podijeljen na tri dijela.

U prvom će dijelu biti opisan plan izrade sustava, koji se sastoji od istraživanja tržišta, obrade ciljeva projekta i opisom tehnologija potrebnih za izradu. Drugi će se dio baviti opisom implementacije korisničkog sučelja te funkcionalnosti vezanih za komunikaciju s korisnikom, tzv. *frontend* aplikacije. Treći će se dio fokusirati na temeljne funkcionalnosti aplikacije, model baze podataka i način integracije te povezivanja različitih dijelova projekta u cjelinu, tzv. *backend* aplikacije.

Ključne riječi: web aplikacija, Spring Framework, Java, backend, frontend

Sadržaj

[Sažetak 1](#_Toc206984243)

[Popis kratica i oznaka 3](#_Toc206984244)

[Popis slika i tablica 4](#_Toc206984245)

[1. Uvod 5](#_Toc206984246)

[2. Tehnologije 6](#_Toc206984247)

[2.1. Java 6](#_Toc206984248)

[2.2. Spring Framework 7](#_Toc206984249)

[2.3. MySQL DBMS 8](#_Toc206984250)

[2.4. Thymeleaf pogonitelj predložaka 8](#_Toc206984251)

[2.5. Flyway alat za migraciju baze podataka 9](#_Toc206984252)

[3. Izrada korisničkog sučelja 11](#_Toc206984253)

[3.1. Thymeleaf predlošci 11](#_Toc206984254)

[3.2. Izgled web stranice 12](#_Toc206984255)

[4. Poslovna logika 13](#_Toc206984256)

[4.1. MVC arhitektura 13](#_Toc206984257)

[4.1.1. Model 13](#_Toc206984258)

[4.1.2. View 14](#_Toc206984259)

[4.1.3. Controller 14](#_Toc206984260)

[4.2. Modularizacija, API baze podataka 14](#_Toc206984261)

[5. Baza podataka 16](#_Toc206984262)

[5.1. MyBatis sustav za komunikaciju sa bazom podataka 16](#_Toc206984263)

[6. Povezivanje modula 18](#_Toc206984264)

[6.1. HTTP zahtjevi 18](#_Toc206984265)

[6.2. Autentifikacija pristupa 19](#_Toc206984266)

[6.3. Serijalizacija i deserijalizacija 19](#_Toc206984267)

[7. Zaključak 20](#_Toc206984268)

[8. Popis literature 21](#_Toc206984269)

# Popis kratica i oznaka

OOP – Objektno orijentirano programiranje

DBMS – *Database Management System*

HTML – *Hypertext Markup Language*

CSS – *Cascading Style Sheet*

SQL – *Structured Query Language*

API *– Aplication Programming Interface*

HTTP – *Hypertext Transfer Protocol*

DTO – *Data Transfer Object*

URL – *Uniform Resource Locator*

UUID – *Universally Unique Identifier*

MVC – *Model-View-Controller*

JSON – *Javascript Object Notation*

XML – *Extensible Markup Language*

# Popis slika i tablica

[Slika 1 Primjer klase AuthorizationController 6](#_Toc206983989)

[Slika 2 Slika ekrana Spring Initializr web stranice 7](#_Toc206983990)

[Slika 3 Primjer HTML koda koji koristi Thymeleaf elemente 9](#_Toc206983991)

[Slika 4 Flyway migracijski direktorij 9](#_Toc206983992)

[Slika 5 Primjer Spring kontrolne jedinice web sučelja 11](#_Toc206983993)

[Slika 6 Primjer klase korištene za instanciranje podataka 13](#_Toc206983994)

[Slika 7 Primjer kontrolne jedinice koja dostavlja podatak u obliku JSON objekta 15](#_Toc206983995)

[Slika 8 Primjer MyBatis repozitorija 16](#_Toc206983996)

[Slika 9 Primjer MyBatis mapper datoteke 17](#_Toc206983997)

[Slika 10 Primjer implementacije različitih funkcionalnosti na istoj krajnjoj točci 19](#_Toc206983998)

[Tablica 1 Najčešće HTTP metode zahtjeva 18](#_Toc206982870)

# Uvod

Za razliku od klasičnih samostalnih aplikacija (eng. *desktop application*) koje se izvršavaju lokalno na korisnikovom računalu, web aplikacije se obično izvršavaju na udaljenom poslužitelju te se s njima komunicira putem web preglednika kao što su *Google Chrome*, *Mozilla Firefox* ili *Microsoft Edge*. U slučaju programskog okvira *Spring Framework*, aplikacija presreće korisnikov *Hypertext Transfer Protocol* (daljnje HTTP) zahtjev te umjesto statičkog resursa korisniku dostavlja dinamički generiran resurs uporabom poslovne logike.

U današnje vrijeme u većini se institucija, bile one službene ili osobne, koriste suvremeni načini komunikacije putem interneta. Zbog ovog razloga sve više se koriste web aplikacije i usluge zbog mogućnosti automatizacije komunikacijskih i poslovnih procesa. Svaka moderna akademska institucija ima svoje web sjedište koje nudi mnoge usluge poput kontaktiranja osoblja, pregled programa, online upisa, i sl. te su takve aplikacije važan sastavni dio svake organizacije koja ih koristi.

U ovom će radu biti prikazan proces izrade web sjedišta za jednu takvu instituciju koje će imati mogućnosti registracije i prijave profesora i studenata, upisivanja i pregleda ocjena, izrade izvještaja o ocjenama, izrade i modifikacije kolegija te administracije korisnika. Izrada je podijeljena na dva dijela – izrada korisničkog sučelja kojim korisnik komunicira s aplikacijom (eng. *frontend*)i izrada poslovne logike koja upravlja korisničkim sučeljima i procesima aplikacije (eng. *backend*).

# Tehnologije

U ovoj sekciji rada navedene su i opisane sve tehnologije, programski jezici i njihove stavke potrebne za izradu web aplikacije.

## Java

A computer screen shot of a program code

Description automatically generatedJava je visokorazinski objektno-orijentirani programski jezik čija se prva inačica pojavila 1995. godine [1]. Objektno orijentirano programiranje (nadalje OOP) temelji se, kao što ime ukazuje, na uporabi objekata koji su pojedinačne instance klasa – programski realiziranih predložaka. Objekt može sadržavati atribute (imenovane podatake) i metode (implementacije poslovne logike za obradu podataka). Ova svojstva mogu biti privatna, što znači da im se može pristupiti samo iz opsega samog objekta, ili javna, što znači da se mogu pozvati u bilo kojem opsegu u kojem je objekt instanciran. Objekt može služiti kao element za spremanje prijenos i obradu podataka, sučelje za vanjski alat, sučelje za metode i sl.

Slika Primjer klase AuthorizationController

(Izvor: Autor)

U gornjem prikazu (Slika 1) može se vidjeti definicija klase *AuthorizationController* koja  
sadrži jedan atribut („*professorService*“) i tri metode („*showAuthorizationView*“, „*processAuthorization*“ i „*initModel*“).

Java se od ostalih programskih jezika izražava time što se aplikacije programirane u Javi ne izvršavaju u korisnikovom izvornom programskom okruženju, već su enkapsulirane u *Java Virtual Machine* – virtualno okruženje koje može izvršavati bilo koji program pisan u Javi. Ovime se postiže neovisnost aplikacije o operacijskom sustavu koji ju pokreće tako da se aplikacije pisane u Javi mogu izvršavati na bilo kojem računalu koje ima instaliranu određenu inačicu Java platforme.

## Spring Framework

A screenshot of a computer

AI-generated content may be incorrect.Spring Framework je programski okvir za Javu razvijen ranih 2000.-ih čija je današnja inačica objavljena 2022. godine [2]. Programski okvir je paradigma za programski razvoj koja olakšava razvoj pružajući korisniku skup alata i modula. Dva najvažnija alata iz okvira Spring nazivaju se *Spring Boot* i *Spring Initializr*.

Slika Slika ekrana Spring Initializr web stranice

(Izvor: https://start.spring.io/)

Spring Initializr koristi se za inicijalizaciju projekta te je vanjski alat. U njemu je moguće odabrati željene funkcionalnosti projekta, inačicu Jave te inačicu Spring Boot alata koja će se koristiti (Slika 2). Rezultirajuća datoteka je inicijalizirani projekt koji sadrži sve zavisnosti (eng. *dependency*) potrebne za realizaciju projekta.

Spring Boot je alat koji „pokreće“ Spring aplikaciju, tj. alat koji omogućuje da Spring projekt bude samostalna aplikacija. U njega su uključeni razni ostali alati poput *Apache* *Tomcat servera* (aplikacijski web poslužitelj za Javu) [3].

## MySQL DBMS

Baza podataka je skup podataka organiziran u tablice koje su strukturirane na način povoljan korisniku baze za njegovu upotrebu. Za upravljanje bazom podataka koriste se *Database Management System* (nadalje DBMS)– sustavi razvijeni za svaku od izvedba baza podataka od kojih su danas najpopularniji MySQL i PostgreSQL.

MySQL je relacijski DBMS. Kao takav podržava relacijske veze između tablica, tj. veze koje opisuju način na koji su tablice povezane te njihov međusobni odnos, koje će biti temeljne za izvedbu ovog projekta. Za upravljanje bazom koristi se *Structured Query Language* (nadalje SQL), deklarativni programski jezik kojim se dobavljaju i spremaju podaci, izvršavaju transakcije, definiraju strukture tablica i sl.

## Thymeleaf pogonitelj predložaka

Thymeleaf je pogonitelj predložaka (eng. *template engine*) korišten za stvaranje dinamičkog *Hypertext Markup Language* (HTML) koda u trenutku korisničkog zahtjeva. Njime se ostvaruje veza između HTML-a koji preglednik prikazuje korisniku aplikacije i aplikacijskog koda te njegove poslovne logike. HTML kod je uobičajeno statički resurs, no uporabom pogonitelja predložaka, programer aplikacije može napraviti predložak koda koji pogonitelj, u trenutku kada korisnik zatraži taj resurs, puni zadanim vrijednostima te preoblikuje po potrebi. U primjeru (Slika 3) vidljivi su elementi označeni svojstvima koja imaju prefiks „*th:*“. Takvi elementi podložni su Thymeleaf pogonitelju te svoje vrijednosti dobavljaju iz Java aplikacije. Osim dobavljanja dinamičkih vrijednosti, uporabom Thymeleaf pogonitelja moguće je i izvršavati grananje korištenjem „*th:if*“ svojstva te dijeljenje HTML koda u module korištenjem „*th:replace*“ svojstva.

A screenshot of a computer code

AI-generated content may be incorrect.(Izvor: autor)

Slika Primjer HTML koda koji koristi Thymeleaf elemente

## Flyway alat za migraciju baze podataka

A screenshot of a computer

AI-generated content may be incorrect.Flyway je alat za migraciju baze podataka. Njime se postiže automatska validacija i ažuriranje baze podataka na najnoviju verziju određenu SQL skriptama koje programer postavlja u alatom određeni direktorij (Slika 4).

Slika Flyway migracijski direktorij

(Izvor: autor)

Skripte u tom direktoriju moraju imati prefiks koji označava inačicu baze podataka, koji se sastoji od slova „V“ (eng. kratica za *version*), broja inačice te dvije donje crte (eng. *dunder*) kako bi ih Flyway alat mogao prepoznati kao migracijske skripte (nije dovoljno samo staviti ih u direktorij). Također, unutar baze podataka stvara se nova tablica „*flyway\_history\_schema*“ koja služi za validaciju uporabom *checksum* vrijednosti same skripte. *Checksum* je jedinstvena vrijednost izračunata zbrajanjem brojčanih vrijednosti znakova u skripti. Kao takva, kako bi se skripta pravilno verificirala, nakon inicijalne migracije na određenu inačicu, ne smije se mijenjati, čak i ako promjena ne utječe na izvršavanje koda kao npr. dodavanje novog praznog reda ili praznog znaka (razmaka). U slučaju semantičke pogreške u skripti koja se izvršava pravilno no daje krivi rezultat, potrebno je napraviti novu migracijsku skriptu koja ju ispravlja.

# Izrada korisničkog sučelja

Ova sekcija rada bavit će se izradom sučelja za komunikaciju sa korisnikom aplikacije. Sučelje je realizirano Thymeleaf kodom koji se kompajlira u HTML te poslužuje korisniku kroz njegov web preglednik. U HTML je također nadovezan *Cascading Style Sheet* (CSS) kod koji definira izgled straničnih elemenata dok sami elementi definiraju strukturu web stranice.

## Thymeleaf predlošci

Thymeleaf predlošci imaju izgled klasičnog HTML-a, no u elementima stranice moguće je definirati svojstva koja dobavljaju svoje vrijednosti iz određenog objekta „*model*“ koji je definiran kontrolnom jedinicom (eng. *controller*) stranice. Svaka stranica ili skup povezanih stranica ima svoju kontrolnu jedinicu kojom programer prosljeđuje podatke dobivene poslovnom logikom aplikacije u predložak. Osim vađenja vrijednosti iz aplikacije u predložak, moguć je i obratan tok podataka, tj. uporabom raznih tehnika (poput HTML forma, JavaScript skripta, itd.) moguće je slati podatke koje korisnik upisuje ili programer definira u aplikaciju za obradu. Ovom se metodom omogućava eksternalizacija funkcionalnosti aplikacije u programski jezik Java, dok korisničko sučelje postaje lako dostupna web stranica bez potrebe da korisnik preuzme posebnu samostalnu aplikaciju. Također, korisnik nema pristup jezgri aplikacije kojim se osigurava povjerljivost izvornog koda i resursa koji se nalaze na udaljenom poslužitelju.

Slika Primjer Spring kontrolne jedinice web sučelja

(Izvor: <https://www.theserverside.com/tutorial/Spring-MVC-tutorial-How-Spring-Boot-aids-Java-web-development>, pristupljeno: 13.8.2025.)

U primjeru (Slika 5) prikazana je struktura tipičnog Java kontrolera koji sadrži metodu za posluživanje predloška „*result.html*“ ili predloška „*indeks.html*“ (u slučaju nepostojećeg ulaznog podatka) pristupom krajnjoj točci (eng. *endpoint*) „*/playagame*“. Kontroler prima ulazni podatak „*theChoice*“ koji je u predlošku referenciran sa „*choice*“. Unutar metode izvršava se logika kojom se određuje ishod „*theOutcome*“ te se sprema u objekt za prosljeđivanje predlošku kao izlazni podatak „*outcome*“.

## Izgled web stranice

Lorem ipsum

# Poslovna logika

Poslovna logika obuhvaća cijelu funkcionalnost aplikacije. To uključuje primanje i obrađivanje korisničkih podataka, posluživanje korisničkih sučelja, čitanje, spremanje i modifikaciju podataka u bazi podataka, autorizaciju i autentifikaciju korisnika te sve povezane operacije nevidljive korisniku.

## MVC arhitektura

Struktura aplikacije definirana je obrascem dizajna koji se koristi kod razvoja. U ovoj aplikaciji koristi se *Model-View-Controller* (MVC) arhitekturni obrazac. Prema obrascu, funkcionalnosti projekta strukturirane su u zasebne datoteke. Kao što ime sugerira MVC arhitektura sastoji se od tri dijela.

### Model

A screen shot of a computer program

AI-generated content may be incorrect.Pojam *Model* se u MVC arhitekturi odnosi na prikaz podataka unutar aplikacije. Unutar Java aplikacije model obuhvaća sve podatkovne klase i pripadajuće repozitorije.

Slika Primjer klase korištene za instanciranje podataka

(Izvor: Autor)

Podaci se unutar aplikacije instanciraju definiranim klasama koje sadrže epitete koji opisuju podatak te metode kojima se taj podatak obrađuje (Slika 6). Klasa označava predložak konkretnog podatka, a objekt označava instancu klase popunjenu epitetima (nižim objektima ili atomskim podacima).

### View

*View* predstavlja prezentacijski sloj. Ovaj sloj varira u implementaciji ovisno o uporabi aplikacije. Odgovoran je za pripremu podataka te njihov izlazni oblik. U slučaju web aplikacije podaci se pripremaju za prikaz u obliku web stranica. Osim prikaza podaci se mogu pripremiti i za prijenos kao *Javascript Object Notation* (nadalje JSON) ili *Extensible Markup Language* (nadalje XML) objekt [4].

### Controller

*Controller* je kontrolna jedinica MVC arhitekture. S njom korisnik komunicira putem prezentacijskog sloja te manipulira podacima koji ustraju kroz isti. Kontrolni se sloj sastoji od mnogih kontrolnih jedinica od kojih je svaka odgovorna za svoj dio web sjedišta (svaka web stranica ima svoju kontrolnu jedinicu). Ovaj sloj komunicira sa servisnim slojem aplikacije kako bi dohvatio potrebne podatke iz baze podataka.

## Modularizacija, API baze podataka

*Application Programming Interface* (nadalje API) je programsko rješenje koje služi komunikaciji između dva aplikacijska procesa. U primjeru (Slika 7) vidljiva je implementacija kontrolne jedinice koja prima zahtjeve na krajnjoj točci „*/api/v1/students/{id}*“ (označenoj sa „URL\_STUDENT“ i „URL\_ID“) te dostavlja JSON objekt koji sadrži podatak iz tablice u bazi podataka.

Aplikacija koja šalje zahtjev mora biti spremna interpretirati odgovor na zahtjev, tj. pretvoriti dobiveni objekt u njoj razumljiv oblik. Pretvorba tokom koje se objekt dijeli u manje dijelove za cilj interpretacije naziva se parsiranje. JSON objekt nije namijenjen biti čitljiv čovjeku, već strukturira podatke u jednostavnu strukturu koju je lako parsirati.

A computer screen shot of a program code

AI-generated content may be incorrect.(Izvor: Autor)

Slika Primjer kontrolne jedinice koja dostavlja podatak u obliku JSON objekta

Modularizacija je tehnika kojom programer izdvaja funkcionalnosti projekta u zasebne module. Moduli mogu biti samostalne aplikacije ili vanjske zavisnosti poput biblioteka. Ova je tehnika uporabljena kako bi se komunikacija s bazom podataka izdvojila u zaseban modul, što pridonosi skalabilnosti projekta. Na ovaj način sučelje baze podataka implementirano je odvojeno od aplikacije koja upravlja web sjedištem te u slučaju promjene temeljnog okvira aplikacije moguća je upotreba sučelja baze podataka bez modifikacija.

# Baza podataka

Baza podataka je strukturirano skladište podataka kojem se pristupa u svrhu trajne pohrane podataka te dobavljanja istih. Baze su podataka sveprisutne u suvremenim programskim rješenjima. Baza aplikacije implementirana je u MySQL relacijskom DBMS-u što olakšava pristup podacima i povezuje podatke u koherentan skup.

## MyBatis sustav za komunikaciju sa bazom podataka

Budući da je aplikacija pisana u Java programskom jeziku izravan pristup bazi podataka nije praktičan te postoje mnogi alati koji tu ulogu obavljaju učinkovitije. Jedan takav alat je MyBatis koji mapira Java metode na SQL izjave. Ovime se olakšava i pojednostavljuje pristup bazi podataka nasuprot alatima poput *Java Persistence API* koji izravno mapiraju podatke iz baze podataka na Java objekte te zahtijevaju definiranje odnosa podataka u samim Java klasama. MyBatis alatom također je moguće detaljno podešavanje SQL izjava po potrebama aplikacije.

Implementacija MyBatis sustava svodi se na dva dijela: deklaracija Java metoda kojom se definiraju ulazni i izlazni podaci te definiranje XML datoteka za preslikavanje (eng. *mapper*) kojima se podaci dobavljaju iz baze podataka te preslikavaju u Java objekte. Također postoji neobavezan treći korak kojim programer može definirati način na koji se podaci iz baze pretvaraju u Java objekte ili atomske podatke uporabom posebnih pretvorbenih klasa (eng. *converter*). U slučaju da pretvorbene klase nisu definirane MyBatis sadrži rudimentarne pretvorbe koje su dovoljne za većinu primjena.

A black background with white text

AI-generated content may be incorrect.U primjeru (Slika 8) vidljiva je implementacija MyBatis repozitorija gdje je klasom „*RefreshTokenRepository*“ moguće dobaviti objekt tipa *JwToken* (koji osim značke sadrži i informacije o isteku i vlasniku) za danu značku ili izbrisati podatak iz baze prema korisničkom imenu.

Slika Primjer MyBatis repozitorija

(Izvor: Autor)

A screenshot of a computer code

AI-generated content may be incorrect.(Izvor: Autor)

Slika Primjer MyBatis mapper datoteke

U slijedećem primjeru (Slika 9) vidljiva je izvedba pripadajuće datoteke za preslikavanje. Unutar „*mapper*“ objekta definirana su tri niža objekta. Prvi objekt „*resultMap*“ služi agregaciji rezultata pretrage baze podataka u Java objekt. Druga dva objekta definiraju SQL izjave za svaku od metoda deklariranih u repozitoriju.

# Povezivanje modula

Moduli projekta realizirani su u obliku dvije zasebne aplikacije. Komunikacija između njih vrši se pomoću HTTP zahtjeva. Kao takva moguća je preko lokalne mreže (ako se obje aplikacije izvršavaju na istom računalu ili mreži) ili preko interneta (u slučaju da se aplikacije nalaze na dva različita udaljena poslužitelja). Prvi modul je aplikacija koja korisniku poslužuje web stranice (*core* modul), a drugi modul je aplikacija s kojom se vrši komunikacija putem programskih sučelja u svrhu dobavljanja podataka iz baze podataka (*api* modul).

## HTTP zahtjevi

HTTP temeljni je protokol kojim se vrši komunikacija putem računalne mreže. Definira nekoliko metoda zahtjeva prikazanih u tablici (Tablica 1):

Tablica Najčešće HTTP metode zahtjeva

|  |  |
| --- | --- |
| Metoda | Namjena |
| GET | dohvat podataka |
| POST | slanje podataka |
| PUT | ažuriranje postojećih podataka |
| DELETE | brisanje podataka |

Moguće je implementirati više funkcionalnosti na jednoj krajnjoj točci uporabom ovih HTTP metoda. U primjeru (Slika 10) vidljiva je implementacija kontrolne jedinice koja koristi različite HTTP metode za izvršavanje različitih funkcija u bazi podataka (čitanje, pisanje, ažuriranje i brisanje) uporabom iste krajnje točke (osim DELETE metode koja na krajnju točku dodaje „*/{id}*“)

A computer screen shot of a program

AI-generated content may be incorrect.(Izvor: Autor)

Slika Primjer implementacije različitih funkcionalnosti na istoj krajnjoj točci

## Autentifikacija pristupa

Budući da je sučelje baze podataka izloženo internetu javlja se potreba da to sučelje bude osigurano od neautoriziranog pristupa.

## Serijalizacija i deserijalizacija

Lorem ipsum

# Zaključak

Lorem ipsum

# Popis literature

1. M. R. Hoffman, „Pre Java 1.0“, javaalmanac.io, <https://javaalmanac.io/jdk/pre1.0/> (pristupljeno 13.1.2025.)
2. J. Hoeller, „Spring Framework 6.0 goes GA“, Spring blog, <https://spring.io/blog/2022/11/16/spring-framework-6-0-goes-ga> (pristupljeno 14.1.2025.)
3. S. Nicoll, „spring-boot“, Github, <https://github.com/spring-projects/spring-boot> (pristupljeno 14.1.2025.)
4. S. Wanigasooriya, „A Deep Dive into Spring MVC: Architecture, Request Flow, and Modern Best Practices“, Medium, <https://sithara-wanigasooriya.medium.com/a-deep-dive-into-spring-mvc-architecture-request-flow-and-modern-best-practices-7fc5a9d510ed> (pristupljeno 13.8.2025.)