SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

**Tomislav Ogrinec**

**Marija Turk**

**Benjamin Filip Šikač**

PROBLEMI UPISA, RANGIRANJA I RASPOREDA

projektna dokumentacija

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Tomislav Ogrinec 0016135746

Benjamin Filip Šikač 0016136096

Studij: Informacijsko i programsko inženjerstvo

Marija Turk 0016138108

Studij: Organizacija poslovnih sustava

PROBLEM UPISA, RANGIRANJA I RASPOREDA

PROJEKTNA DOKUMENTACIJA

Mentori:

Grabar Darko, mag. inf.

Prof. dr. sc. Divjak Blaženka

Varaždin, siječanj 2022.

**Sadržaj**

[1. Problem upisa, rangiranja i rasporeda 1](#_Toc93334616)

[2. Strategija rješavanja 2](#_Toc93334617)

[3. Karakteristike rješenja 3](#_Toc93334618)

[4. Interpretacija rezultata 5](#_Toc93334619)

[5. Popis slika 6](#_Toc93334620)

[6. Literatura 7](#_Toc93334621)

# Problem upisa, rangiranja i rasporeda

Glavni problem na koji nailazimo kod samog upisivanja fakulteta jest kako odrediti tko će upisati željeni fakultet, a tko neće. Možemo uvidjeti kako obje strane, fakulteti sa svojim preferencijama i studenti sa željenim fakultetima, biraju svoju listu prioriteta. U grubo, ovo možemo sagledati kao dva skupa koji se međusobno moraju povezati u neke relacije, bolje rečeno potskupove, gdje je jedan fakultet u relaciji sa određenim brojem studenata.

Pogled studenta posve je jednostavan. On se samo sastoji od popisa fakulteta koji su poredani od prioriteta najviše razine do prioriteta najniže razine. Na drugu ruku imamo pogled fakulteta koji u svojem popisu ima studente od najoptimalnijeg do zadnjeg ovisno o ostvarenim bodovima. Kada bi uzeli kao primjer upisivanje srednjoškolaca na fakultete, imamo prosjek ocjena, bodove sa matura i još neke dodatne mogućnosti kao na primjer državna natjecanja. Također, svaki fakultet ima svoje kriterije prema kojima vrednuje ostvarene bodove na maturama.

Kao što je navedeno u opisu problema iz prve faze, koristit ćemo Gale-Shapleyev algoritam kako bi implementirali sustav koji će upisati studente na željene fakultete/smjerove. U samoj implementaciji algoritma, možemo birati kako ćemo obrađivati podatke. Jedan od načina može biti da sam program čita podatke iz baze podataka te on sam obrađuje sveukupnu radnju upisa studenata. Drugi način može biti da nam programsko rješenje predstavlja samo posrednika koji se poziva kada su podaci već spremni i trebamo koristiti samo algoritam. Mi smo odabrali raditi drugi način gdje mi dobivamo podatke u XML obliku te ih obrađujemo na način da su sve obrade računanja bodova ostvarenih upisom već unaprijed izračunate. Time smo postigli otvorenost primjene našeg rješenja iz razloga što mogu postojati različiti načini računanja bodova ostvarenih prilikom upisa, te se onda naše rješenje može primijeniti kako bi se upis odvio do kraja.

# Strategija rješavanja

Kako smo ranije ustanovili neku početnu putanju razvoja našeg projekta, mi ćemo se ograničiti tako da je naše rješenje primjenjivo na više različitih problema upisa. Kako sam algoritam zahtjeva puno iteracija kroz skupove podataka, uzeli smo programski jezik Python kako bi sve te iteracije jednostavno riješili. Kako ne bi uskočili u moguće probleme, morali smo prvo istražiti tehnologiju te se dobro upoznati sa samim algoritmom i podacima koje ćemo uspoređivati.

Kako bi potvrdili odabir tehnologije napravili smo „PoC“ (*engl. Proof-of-Concept*), što je jedan od najpoznatijih načina postavljanja i rješavanja problema u IT industriji. Pomoću njega smo potvrdili da možemo riješiti jedan vrlo jednostavan primjer problema upisa. Nakon toga, morali smo se prisjetiti kako ide konverzija podataka iz XML oblika u klase podataka, te zatim iz klasa podataka u JSON oblik.

Praćenjem trenutnih ISO standarada, plan razvoja programskog rješenja bio je vođen vrlo kratkom SCRUM metodologijom. Imali smo samo jedan sprint (radni dvotjedan) u kojem smo sve uspješno napravili. Tehničku dokumentaciju ovog rješenja smo sastavili kao najtočniji i najpouzdaniji oblik opisa rješenja, te ćemo ju prikazati i obrazložiti kasnije u karakteristikama rješenja.

# Karakteristike rješenja

Sama ovisnost klasa unutar našega rješenja pokazana je na slici ispod kao UML dijagram klasa. Možemo vidjeti kako je ovo posve jednostavno rješenje, kako bi se vrlo lako moglo primijeniti na čim više različitih oblika upisa studenata na fakultete. Kako smo se odlučili raditi u programskom jeziku Python, morali smo ranije zadani/predloženi UML klasa prilagoditi našem rješenju.

Diagram

Description automatically generated

Slika 1 UML klasa (vlastita izrada)

Rješenje je napravljeno po svim zadanim karakteristikama rješenja iz prve faze projekta. Napravili smo vrlo jednostavan program koji se lako može prilagoditi i postaviti kao online servis. On može primati podatke ili kako već sad ima složeno, da sam prilikom pokretanja učitava xml podatke i provodi postupak upisa studenata te kao rezultat dobivamo JSON datoteku u kojoj se nalaze rang liste po fakultetima.

Samo rješenje možete pronaći na sljedećem linku: <https://github.com/al0cam/Gale-Shapley-algorithm>.

Kako bi pokrenuli rješenje, morate ga preuzeti preko git komandne linije ili u već neki programski editor koji ima mogućnost spajanja na GitHub. Unutar preuzetog direktorija, uočit ćete izvor literature kao i samo rješenje te Excel datoteku unutar koje se nalaze testni podaci. Kada bi se koristili pravi podaci, morala bi se samo zamijeniti Excel datoteka i preimenovati ili jednostavnije samo upisati naziv nove datoteke unutar klase Algorithm na liniji 14 i 15, kao što je prikazano na sljedećoj slici.



Slika 2 Pikaz klase Algorithm - unos naziva i/ili putanje željene Excel datoteke (vlastita izrada)

Možemo vidjeti kako popis fakulteta i njihovih kvota učitavamo da druge stranice pod nazivom „Kvote“, tako bi se i novi skup podataka ponašao. Ako je neki sadržaj na drugoj stranici unutar iste Excel datoteke, morali bismo dodatno napomenuti kao i u liniji 15.

Table

Description automatically generated

Slika 3 Prikaz testnih podataka za studente (vlastita izrada)

Graphical user interface, text, application, email

Description automatically generated

Slika 4 Prikaz kvota fakulteta (vlastita izrada)

Na slikama 3 i 4, možemo vidjeti oblik podataka kakvi su poželjni kako bi se moglo koristiti naše rješenje bez ikakvih promjena.

# Interpretacija rezultata

Opis rezultata koji smo postigli, jesmo li uspješno napravili zadatak i tak stvari vezano uz naše rješenje koje predlažemo kao novi i inovativan način rješavanja problema

Obavezno prikazite primjer JSONa

# Popis slika

[Slika 1 UML klasa (vlastita izrada) 3](#_Toc93334592)

[Slika 2 Pikaz klase Algorithm - unos naziva i/ili putanje željene Excel datoteke (vlastita izrada) 4](#_Toc93334593)

[Slika 3 Prikaz testnih podataka za studente (vlastita izrada) 4](#_Toc93334594)

[Slika 4 Prikaz kvota fakulteta (vlastita izrada) 4](#_Toc93334595)

# Literatura