Web aplikacija za rješavanje problema rasporeda pomoću genetskog algoritma

Jurinčić, Jurica

Undergraduate thesis / Završni rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Pula / Sveučilište Jurja Dobrile u Puli**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:137:031393

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-13**

Repository / Repozitorij:

Digital Repository Juraj Dobrila University of Pula

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

JURICA JURINČIĆ

WEB APLIKACIJA ZA RJEŠAVANJE PROBLEMA RASPOREDA POMOĆU GENETSKOG ALGORITMA

Završni rad

Pula, 9. rujna, 2020. godine

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

JURICA JURINČIĆ

WEB APLIKACIJA ZA RJEŠAVANJE PROBLEMA RASPOREDA POMOĆU GENETSKOG ALGORITMA

Završni rad

JMBAG: 0303075593, redoviti student

Studijski smjer: Informatika

Predmet: Funkcijsko programiranje

Znanstveno područje: Tehničke znanosti
Znanstveno polje: Računarstvo
Znanstvena grana: Programsko inžinjerstvo
Mentor: doc.dr.sc Siniša Miličić

Pula, 9. rujna, 2020. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Jurica Jurinčić, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Jurica Jurinčić

U Puli, 9. rujna, 2020. godine

IZJAVA
o korištenju autorskog djela

Ja, Jurica Jurinčić dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „Web aplikacija za rješavanje problema rasporeda pomoću genetskog algoritma„ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 9. rujna, 2020. godine

Potpis

Jurica Jurinčić

Sadržaj

1. Uvod…. 1
2. Motivacija 2
3. Razrada funkcionalnosti 3
3.1. Use case dijagram 3
3.2. Class dijagram 4
4. Korisničko sučelje 5
5. Implementacija 7
5.1. Genetski algoritam 7
5.1.1. Mutate 8
5.1.2. Crossover 8
5.1.3. Nastavnik/Studij shuffle 8
5.1.4. Kapacitet/Računala shuffle 9
5.2. Razgovor genetskog algoritma sa front end-om 10
5.2.1. Polling 10
5.2.2. Pohranjivanje privremenih rezultata 11
5.2.3. Zaustavljanje računanja prirodnom konvergencijom 11
5.2.4. Zaustavljanje računanja proizvoljnim uvjetom 12
6. Korisničke upute 13
7. Zaključak 23
8. Popis literature i alata 24
9. Prilozi 25
10. Sažetak 26

1. Uvod

Problem pronalaska optimalne konfiguracije sa kompliciranim ograničenjima se javlja kod izrade akademskih rasporeda gdje je pri izradi potrebno uzeti u obzir vremensku dostupnost dvorana, profesora, semestra, kapacitet dvorana u usporedbi sa potrebnim kapacitetom predmeta, dostupnost računalih dvorana za računalne predmete i sl.

Ne postoji efikasan (polinomni) algoritam koji uzima u obzir sve potrebne činjenice i vraća optimalan raspored predmeta što na raspolaganju ostavlja samo tzv. „brute force“ algoritme. Jedan pametniji brute force algoritam koji je pogodan za rješavanje problema ovakvog tipa je „genetski algoritam“ koji ima funkciju cilja i u iteracijama deterministički ili slučajno mijenja stanje objekata i ocjenjuje koliko dobro ti novi objekti zadovoljavaju postavljena ograničenja. Nakon svake iteracije samo N najbolje ocijenjenih objekata služi kao ulaz u sljedeću iteraciju. Testiranjem smo došli do zaključka da je naš genetski algoritam zaista u stanju proizvesti koristan raspored. Vrijeme potrebno da vrati raspored uvelike ovisi o početnim ograničenjima, primjerice nije isto ako su sve dvorane uvijek slobodne ili ako su slobodne samo u tijesnim terminima. U drugom slučaju će algoritam morati duže raditi i vjerojatnost da vrati manje kvalitetno rješenje (s gorom ocjenom) je veća.

Pored genetskog algoritma, aplikacija također sadrži i cjelokupno korisničko sučelje za unos svih potrebnih podataka i tzv. „Workspace“ na kojem korisnik može ručno manipulirati predmetima u rasporedu. Moguće je osim kreacije čistog rasporeda, kao početni raspored postaviti i genetski izgenerirani raspored te ga dalje ručno modificirati.

2. Motivacija

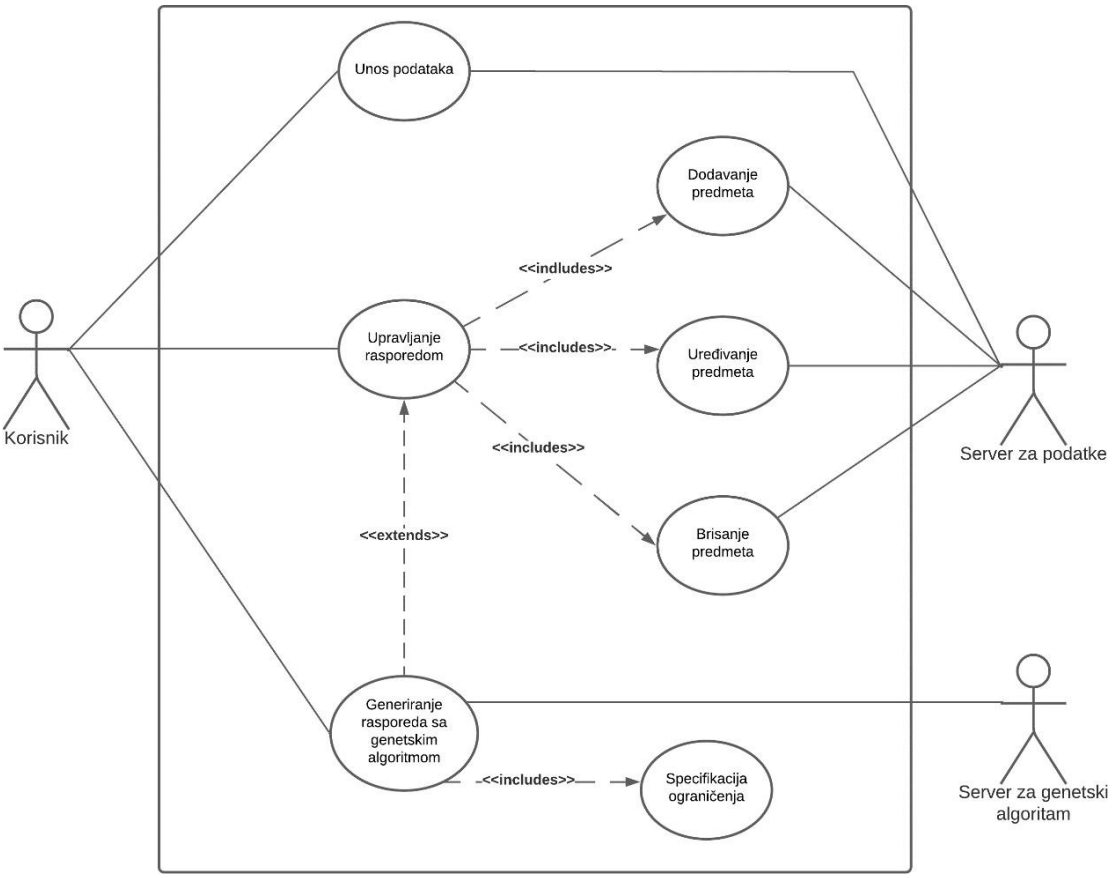
Mnoga sveučilišta i dalje svoj raspored izrađuju ručno na način da definiraju odgovornu osobu koja će svaki predmet staviti u određenu dvoranu, dan i sat te ručno izračunati da li su se desile kolizije između slobodnih vremena profesora, semestra, dvorana, kapaciteta dvorana i potrebe za računalnosti. Takav proces, iako će kad-tad dati rezultate je veoma spor i mukotrpan.

Aplikacija uvelike olakšava cjelokupan proces na način da pruža interaktivno sučelje koje dozvoljava lak unos, filtriranje i uređivanje predmeta te upozorava na bilo kakve kolizije prilikom unosa predmeta što znači da se skoro cijeli teret računanja kolizija skida sa korisnika. Uz to, prisutan je i genetski algoritam koji dopušta specifikaciju ograničenja po kojima generira cjelokupan raspored koji je moguće dalje ručno uređivati po želji.

Za realizaciju aplikacije biti će potreban hosting za front-end, hosting za mikroservis unosa, uređivanja i brisanja podataka te virtualni privatni server koji će provoditi izračune genetskog algoritma. Važno je da virtualni privatni server bude dovoljno jak da podrži ciljani broj korisnika aplikacije.

3. Razrada funkcionalnosti

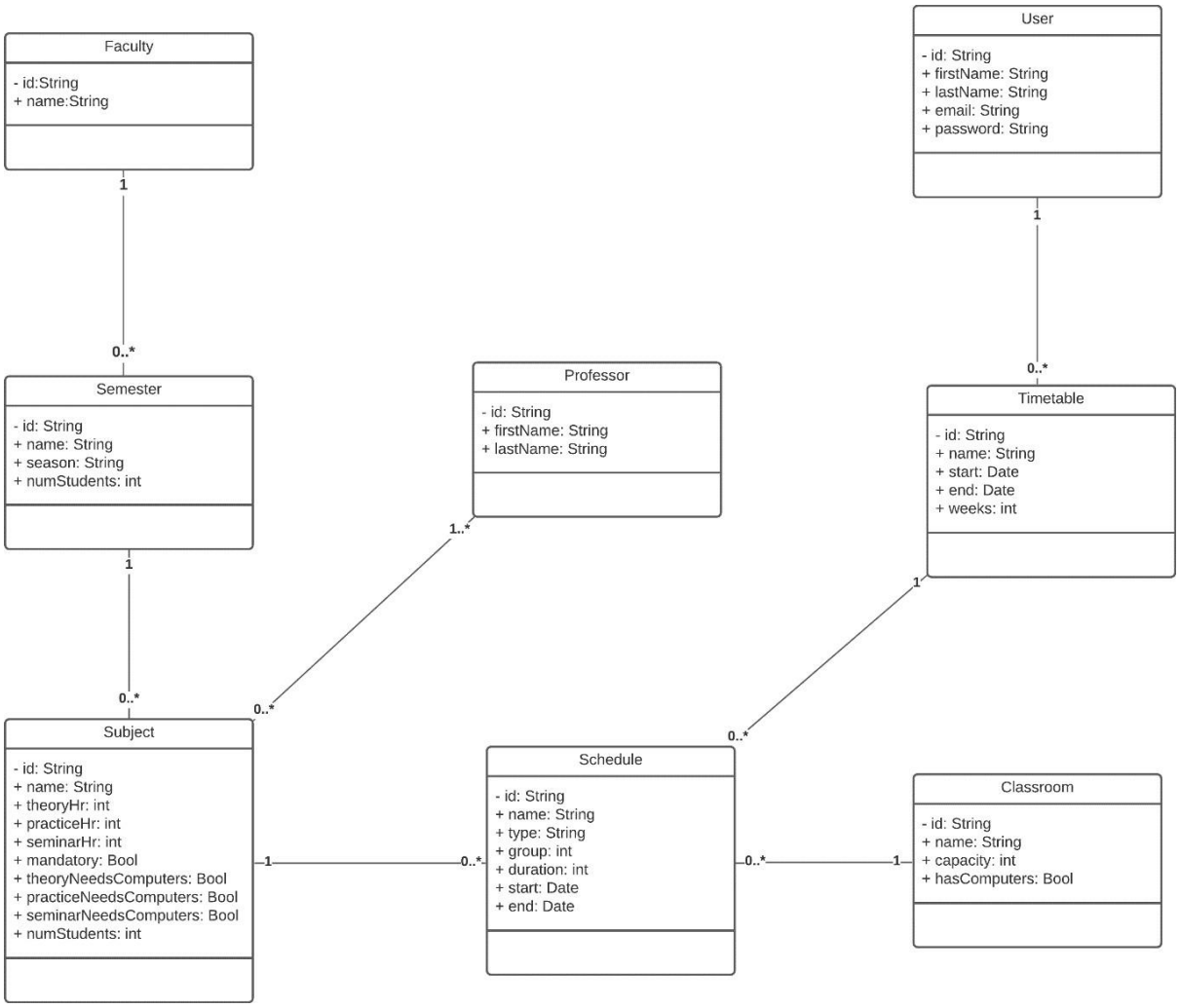
3.1. Use case dijagram



Slika 1 - Use case dijagram
Izvor: Lucidchart aplikacija

Use case dijagram prikazuje „krovni“ prikaz aplikacije gdje korisnik ima pristup funkcionalostima kao što su unos podataka, upravljanje rasporedom i generiranje rasporeda sa genetskim algoritmom. Za bilo koje upravljanje podacima zaslužan je server za podatke dok genetskim algoritmom upravlja zaseban server.

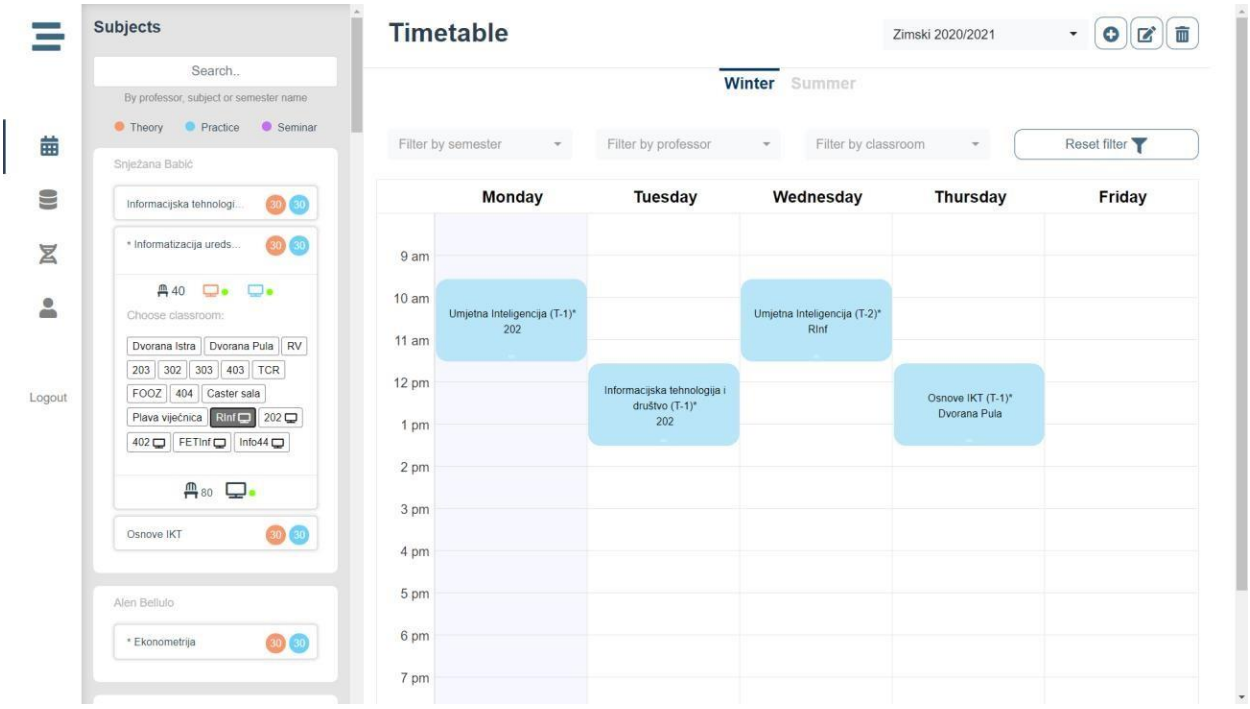
3.2. Class dijagram



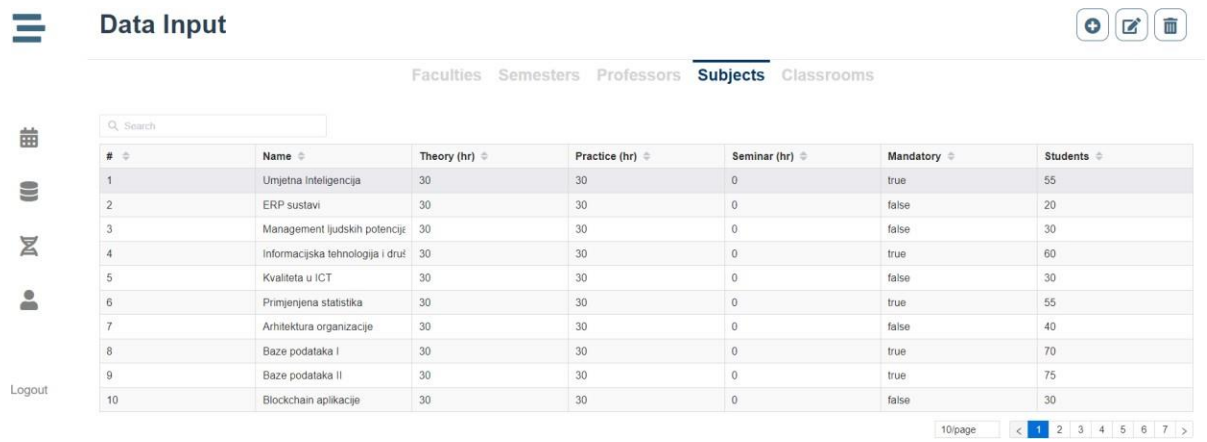
Slika 2 - Class dijagram, Izvor: Lucidchart aplikacija

Class dijagram prikazuje klase u kojima su enkapsulirani podaci. Fakultet ispod sebe može imati više semestara, a svaki semestar može imati više predmeta. Svaki predmet mora imati barem 1 profesora, dok svaki profesor može imati više predmeta. Predmeti mogu imati više tzv. „Schedule-a“ koji reprezentiraju jedinicu predmeta koja ide u raspored i sadrži početak, kraj i dvoranu održavanja. Svaki Schedule ima striktno jednu dvoranu dok jedna dvorana može imati više Schedule-a. Jedan raspored može imati više Schedule-a te jedan korisnik može imati više rasporeda.

4. Korisničko sučelje

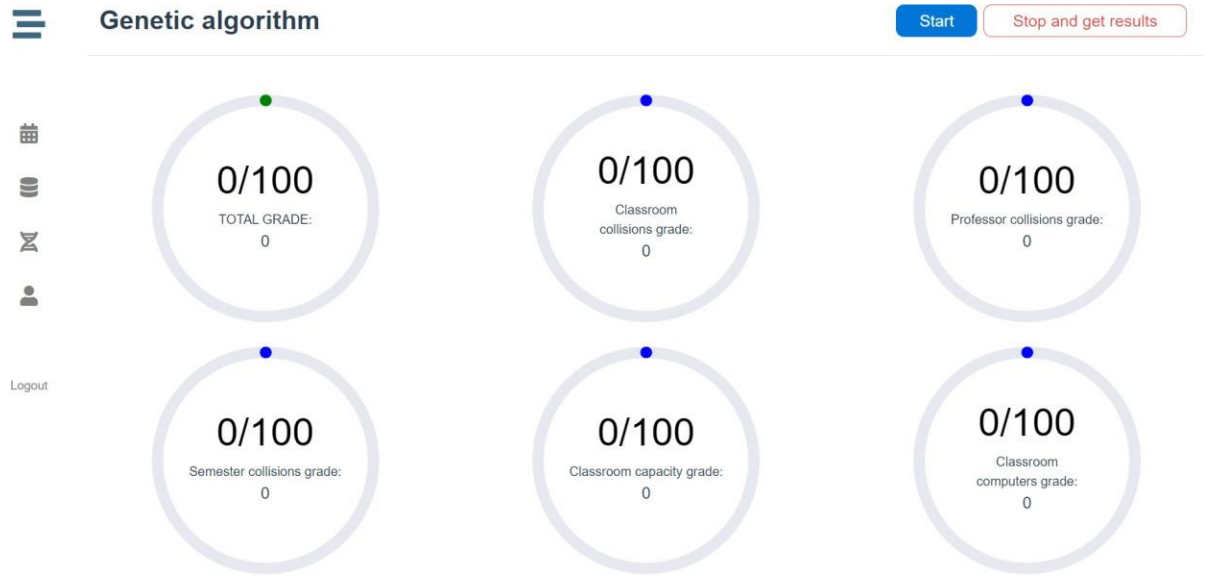


Slika 3 - Početna stranica
Izvor: Vlastita aplikacija



#	Name	Theory (hr)	Practice (hr)	Seminar (hr)	Mandatory	Students
1	Umjetna Inteligencija	30	30	0	true	55
2	ERP sustavi	30	30	0	false	20
3	Management ljudskih potencija	30	30	0	false	30
4	Informacijska tehnologija i druš	30	30	0	true	60
5	Kvaliteta u ICT	30	30	0	false	30
6	Primjenjena statistika	30	30	0	true	55
7	Arhitektura organizacije	30	30	0	false	40
8	Baze podataka I	30	30	0	true	70
9	Baze podataka II	30	30	0	true	75
10	Blockchain aplikacije	30	30	0	false	30

Slika 4 - Unos podataka
Izvor: Vlastita aplikacija



Slika 5 - Genetski algoritam
Izvor: Vlastita aplikacija

5. Implementacija

Front end je implementiran u okviru Vue JS, servis za upravljanje podacima u okviru Express JS, dok je servis za genetski algoritam razvijen u Python-u i omotan u mikro okvir Flask. O implementaciji samog front end-a i servisa sa upravljanje podacima neće biti puno govora jer koriste standardnu logiku koja rješava standardne probleme dizajna i unosa/uređivanja/brisanja podataka. Sam kod ih dovoljno dobro dokumentira.

5.1. Genetski algoritam

Tipično genetski algoritmi imaju ograničenja, funkciju cilja i optimizatore. Pri svakoj iteraciji se pokušava uz pomoć optimizatora dobiti što bolja vrijednost funkcije cilja. Za problem rasporeda su definirana sljedeća ograničenja:

- Dvorana ne smije imati više od jednog predmeta u isto vrijeme
- Ako dvorana ima inicijalno nedostupne termine, moraju se poštovati
- Profesor ne smije imati više od jednog predmeta u isto vrijeme
- Ako profesor ima inicijalno nedostupne termine, moraju se poštovati
- Semestar ne smije imati više od jednog predmeta u isto vrijeme
- Dvorana mora imati dovoljno velik kapacitet za sve studente svojih predmeta
- Dvorana mora biti računalna ukoliko je predmet računalan

Svako kršenje ograničenja ugrožava vrijednost funkcije cilja, a ona je optimalna kada su sva ograničenja zadovoljena.

Optimizatori pokušavaju poboljšati vrijednost funkcije cilja tako što mijenjaju stanje objekata. To mijenjanje može biti deterministički – „ako je uvjet zadovoljen onda x promijeni u y“, slučajno – „slučajnim odabirom promijeni stanje objekata“ ili kombinirano pr. „ako je uvjet zadovoljen onda slučajnim odabirom promijeni stanje objekata“.

5.1.1. Mutate

Optimizator mutate jedan je od temeljnih u genetskim algoritmima. Implementiran je na način da u rasporedu slučajnim odabirom odabere inačicu predmeta, zatim dohvati sve dostupne termine (dvorana, dan, sat) i slučajnim odabirom odabere jedan termin. Taj termin se pridružuje inačici predmeta.

5.1.2. Crossover

Optimizator crossover je također jedan od temeljnih u genetskim algoritmima. Implementiran je na način da prima dva različita rasporeda, zatim iterira kroz prvi i svakoj inačici predmeta na tom rasporedu pridružuje termin (dvorana, dan, sat) koji će biti slučajan odabir između već postojećeg termina u prvom rasporedu i drugog termina u drugom rasporedu.

5.1.3. Nastavnik/Studij shuffle

Optimizatori shuffle su kombinirani – deterministički i slučajni, iako veliki dio posla obavljaju u deterministčkom stilu. Iteriraju kroz objekte koji reprezentiraju zadovoljenost ograničenja, primjerice koliko inačica predmeta nastavnici ili studiji imaju u isto vrijeme (na isti dan, sat u tjednu) te traže prvu koliziju, odnosno nezadovoljenost ograničenja. Kada algoritam pronađe nastavnika (ili studij) sa danom i satom koji je preopterećen onda traži u rasporedu sve inačice predmeta koje izvodi taj nastavnik i koji se održavaju na pronađeni dan i sat. Zatim dohvaća samo one termine (dvorana, dan, sat) u kojima problematične inačice predmeta mogu postojati bez kršenja ograničenja. Jedan termin je iz tog skupa slučajno odabran i pridružen prethodno problematičnoj inačici predmeta. U praksi se pokazalo da je ovakav pristup značajno smanjio potrebno vrijeme računanja optimalnog rasporeda.

5.1.4. Kapacitet/Računala shuffle

Kapacitet i računala su još dvije varijante shuffle ideje za optimizator. Iteriraju kroz raspored i ispituju da li dodijeljena dvorana ima dovoljno velik kapacitet da podrži studente koji pohađaju inačicu predmeta te da li je dodijeljena dvorana računalna ukoliko inačica predmeta zahtijeva računalnost. U slučaju da pronađu kršenje ograničenja, dohvaćaju one termine (dvorana, dan, sat) u kojima problematične inačice predmeta mogu postojati bez kršenja ograničenja. Termin će biti iz skupa slučajno odabran i pridružen prethodno problematičnoj inačici predmeta. Vrlo je korisno imati više varijanta shuffle ideje za optimizator jer su ograničenja ulaznih podataka kvantitativno nepredvidiva i nije moguće znati koji će optimizator u kojem trenutku dati najbolje pomake prema optimalnosti.

5.2. Razgovor genetskog algoritma sa front end-om

Postavlja se pitanje kako će genetski algoritam komunicirati sa front end-om ako izvršava vremenski dugo računanje. Bilo koja klasična HTTP request/response konekcija će doživiti timeout prije nego uopće dođe blizu rješenja. Ispada da je ovaj tip problema dobro poznat u softverskim krugovima i postoji nekoliko različitih rješenja. U ovoj aplikaciji je implementirano najjednostavnije rješenje koje koristi bazu podataka kao middleware za komunikaciju.

5.2.1. Polling

Ideja je da front end pokrene dugo računanje na način da pošalje običan HTTP request servisu koji računa. Servis odmah vraća response „202 accepted“ zajedno sa URL-om na kojem će front end moći pratiti rezultate računanja. Jednom kada je servis vratio HTTP response, kreće u računanje.

Front end čim dobije URL kreće u tzv. „polling“, odnosno iterira po tom URL-u i u svakoj iteraciji šalje HTTP request u kojem servis šalje response sa trenutnim rezultatima računanja. Koliko dugo će front end iterirati i kojim tempom će slati request-ove ovisi od situacije do situacije, ali obično je normalno da pošalje request čim dobije response (ili svakih N sekundi) i da prestane iterirati kada je računanje gotovo (ili kada je neki drugi proizvoljni uvjet zadovoljen).

5.2.2. Pohranjivanje privremenih rezultata

Dalje se postavlja pitanje kako će servis u sred računanja obavijestiti front end o trenutnim rezultatima. Odgovor je sljedeći - neće direktno kontaktirati front end, ali će pohranjivati svaki ažurni rezultat u bazu podataka pod kolekcijom privremenih rezultata. Postojati će end point nad kojim će front end vršiti polling koji će dohvaćati rezultate iz te kolekcije i slati ih nazad.

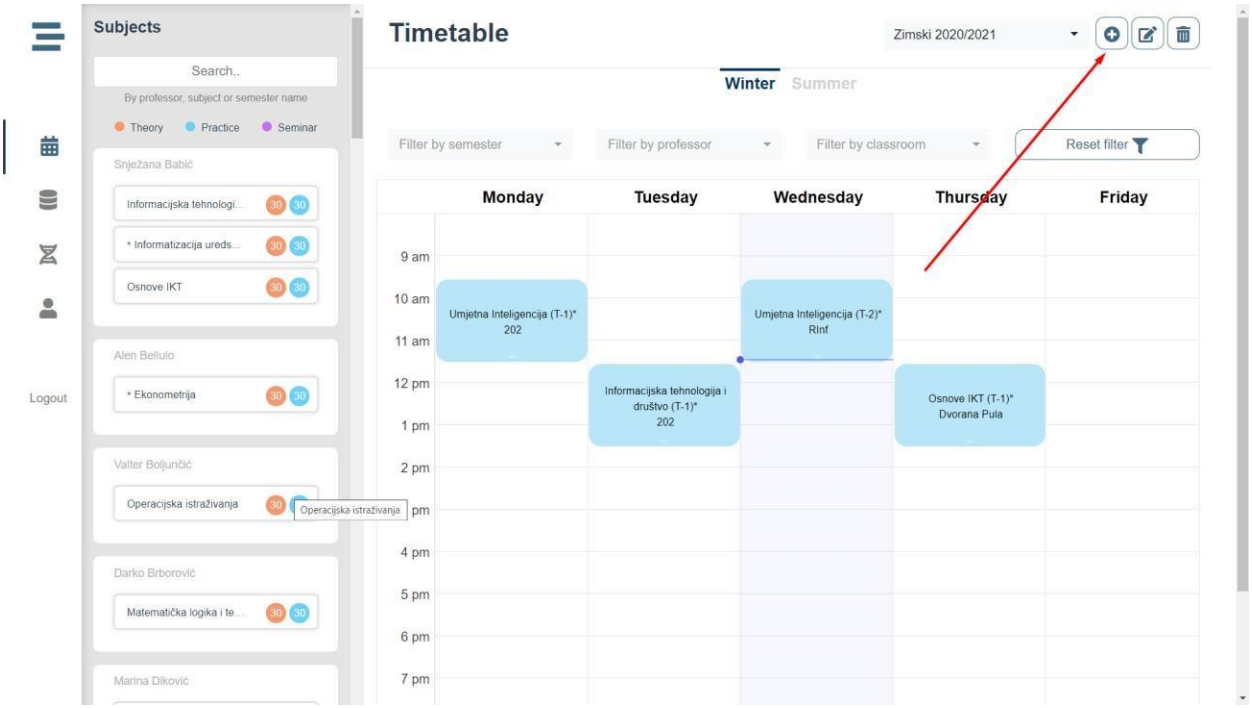
5.2.3. Zaustavljanje računanja prirodnom konvergencijom

Još je ostalo jedno neistraženo pitanje, a to je kako će se računanje zaustaviti na način da front end dobije završne rezultate, prestane iterirati po URL-u i da servis prestane računati. Odgovor je – ovisi u kojem trenutku se računanje zaustavi. Ukoliko se radi u najblažem slučaju koji je zaustavljanje računanja kada je računanje prirodno konvergiralo, onda će servis jednostavno zapisati u bazu privremenih rezultata da je proces završio (svi privremeni rezultata računanja dijele isti ID računanja) i pohraniti će konačni rezultat u novu kolekciju konačnih rezultata. Front end cijelo vrijeme provjerava da li mu se u response-u nalazi upravo taj dokument koji označava kraj računanja. Ukoliko je to slučaj, prekida stari polling, dobija novi URL za novi polling na kojem se nalaze konačni rezultati (novi polling je potreban radi delay-a spremanja konačnih rezultata). Jednom kada front end dohvati konačne rezultate, proces je gotov.

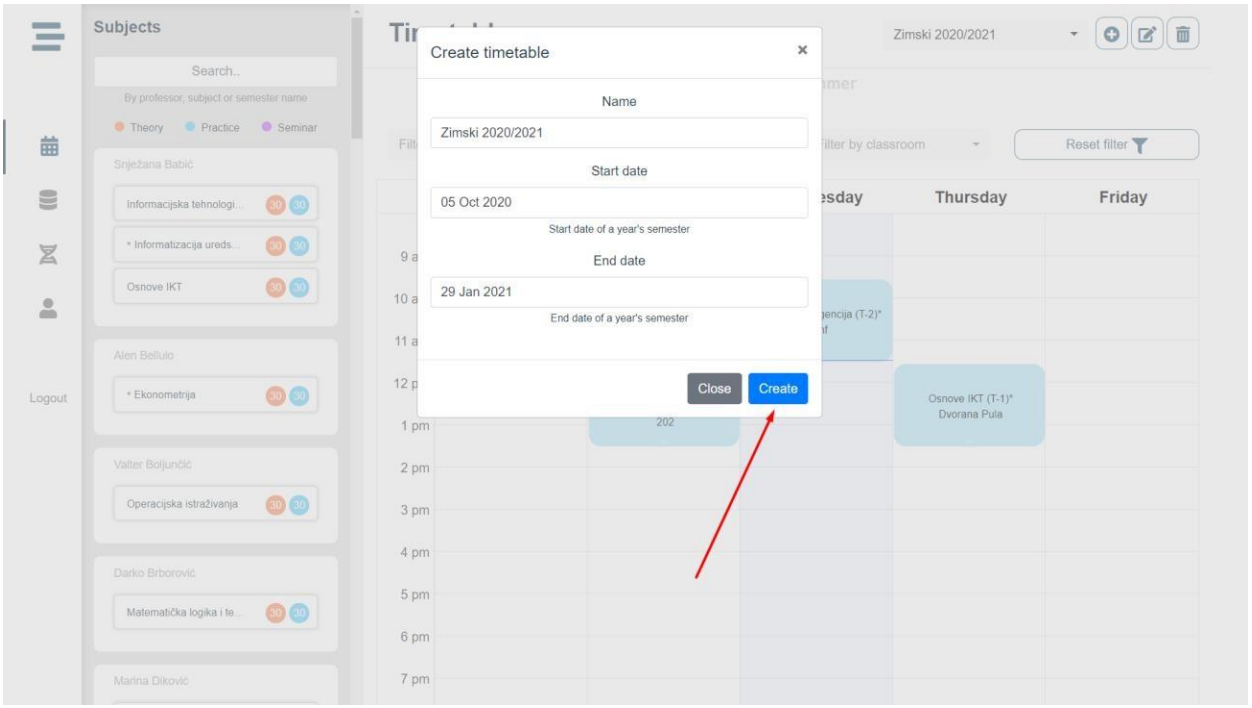
5.2.4. Zaustavljanje računanja proizvoljnim uvjetom

Međutim, što ako korisnik želi u proizvoljnom trenutku zaustaviti računanje bez da je ono prirodno konvergiralo pomoću gumba „STOP“, kako mu to omogućiti? Odgovor je - front end će prilikom pritiska gumba „STOP“ poslati zahtjev za stvaranje novog dokumenta u bazi pod kolekcijom prekida računanja. Taj dokument će sadržavati ID računanja i korisnikov ID. Servis računanja prije svake iteracije provjerava kolekciju prekida računanja i traži dokument sa relevantnim ID-om računanja i ID-om korisnika te ukoliko ga pronađe, prekida sljedeće iteracije i odmah upisuje trenutne rezultate kao konačne rezultate (u kolekciju konačnih rezultata), zatim zapisuje u kolekciju trenutnih rezultata da je proces gotov. Front end dobija response da je proces gotovo, zaustavlja stari polling i kreće u novi polling za konačne rezultate. Jednom kada front end dohvati konačne rezultate, proces je gotov.

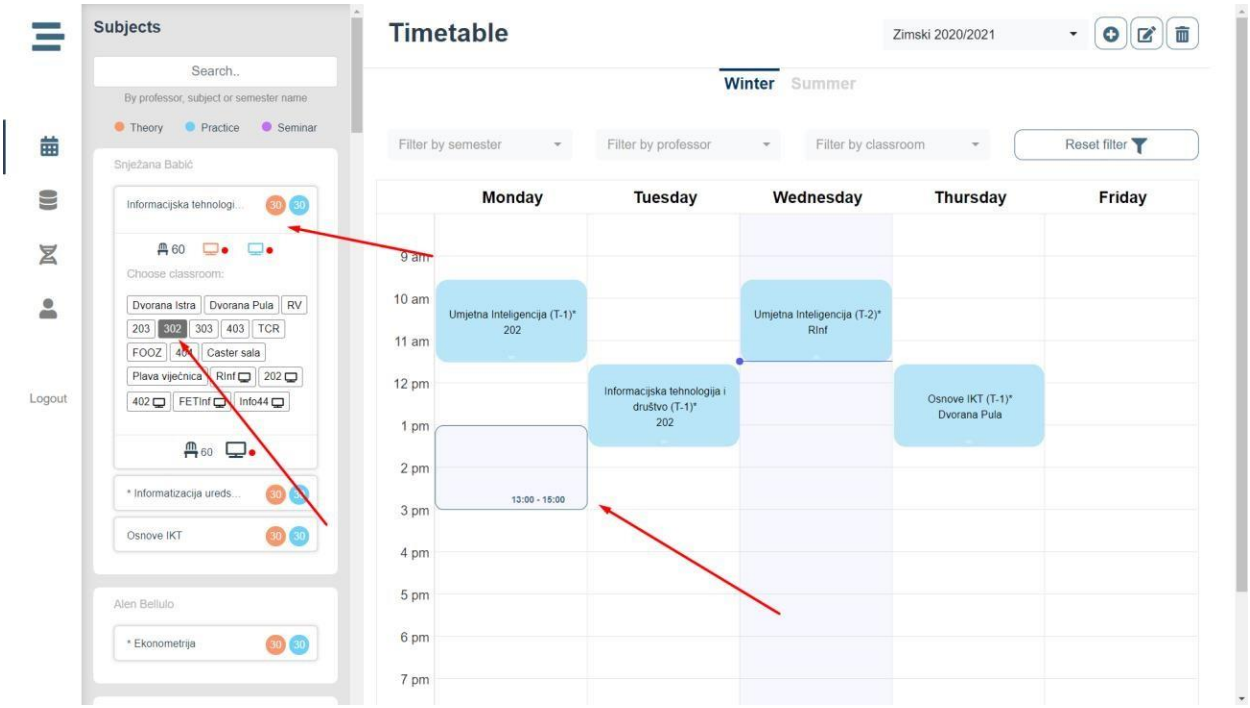
6. Korisničke upute



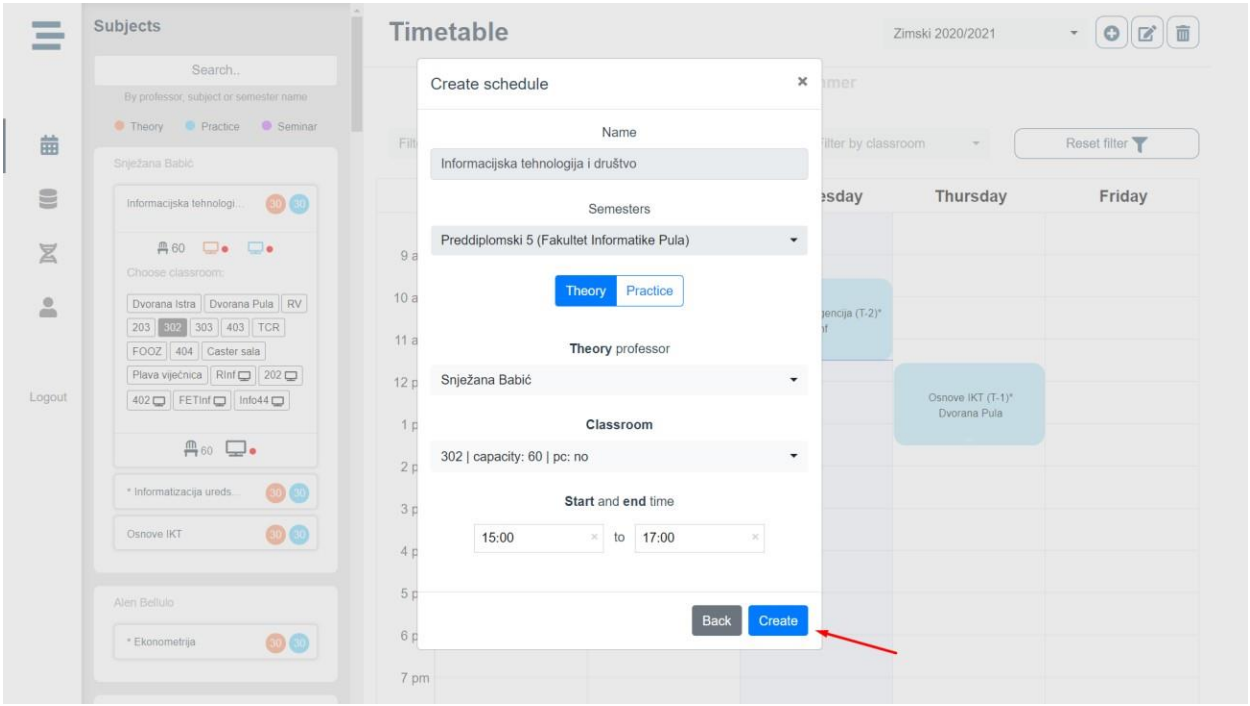
Slika 6 - Stvori raspored, Izvor: Vlastita aplikacija



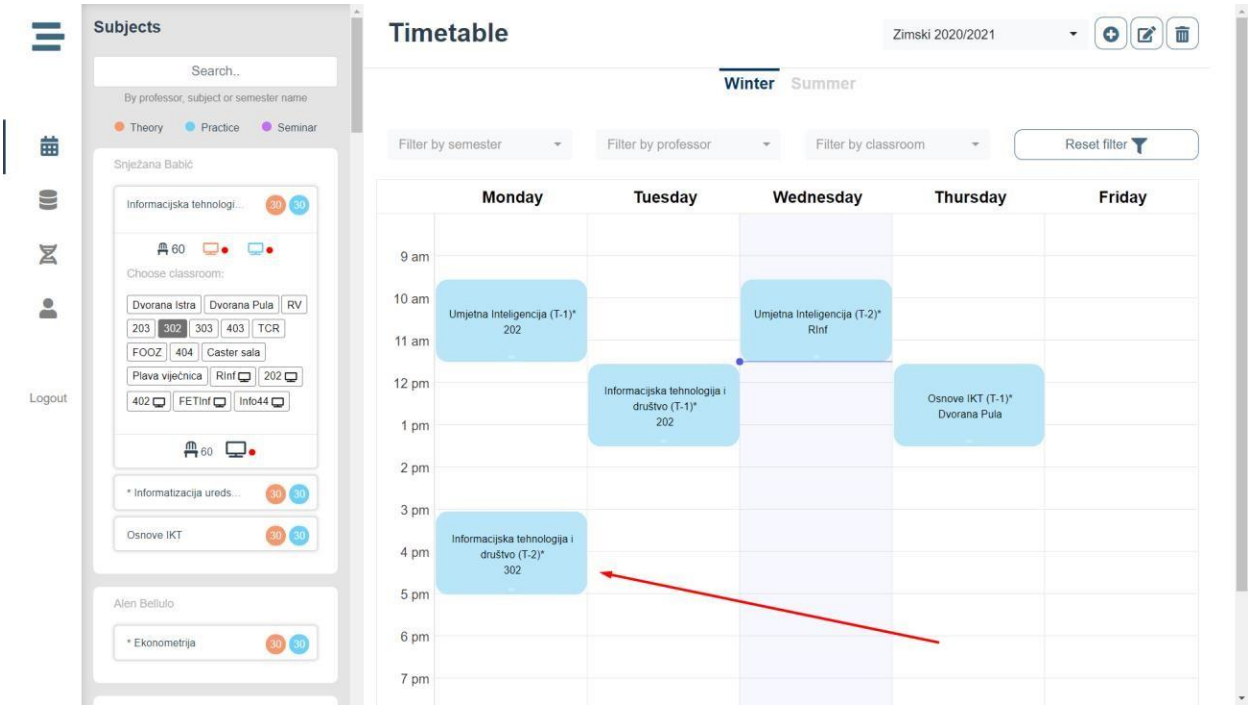
Slika 7 - Stvori raspored prozor, Izvor: Vlastita aplikacija



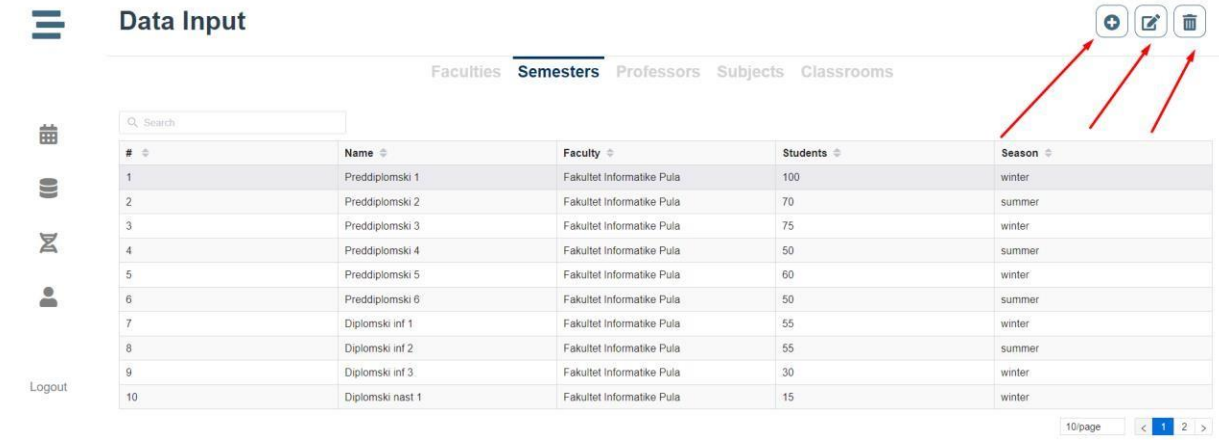
Slika 8 - Stavi predmet na raspored
Izvor: Vlastita aplikacija



Slika 9 - Stavi predmet na raspored prozor
Izvor: Vlastita aplikacija

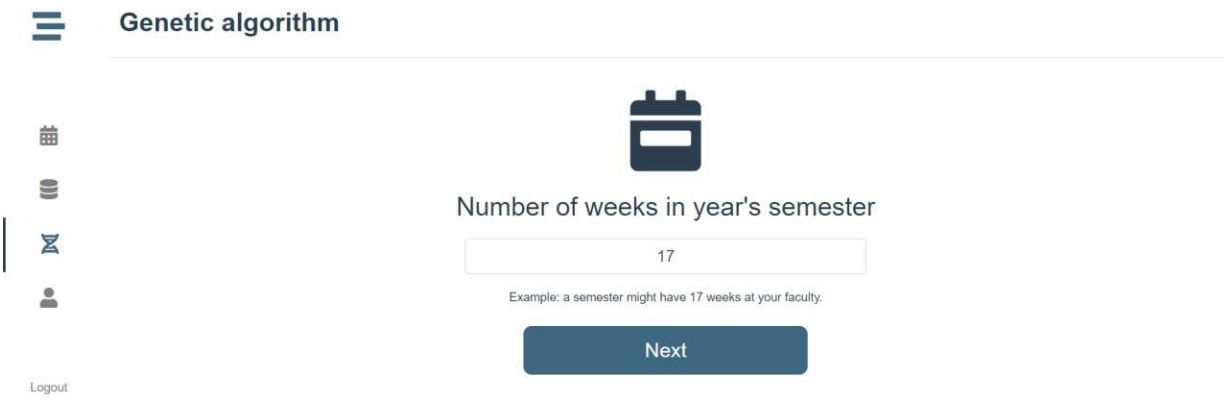


Slika 10 - Predmet stavljen na raspored (drag svih predmeta moguć), Izvor: Vlastita aplikacija

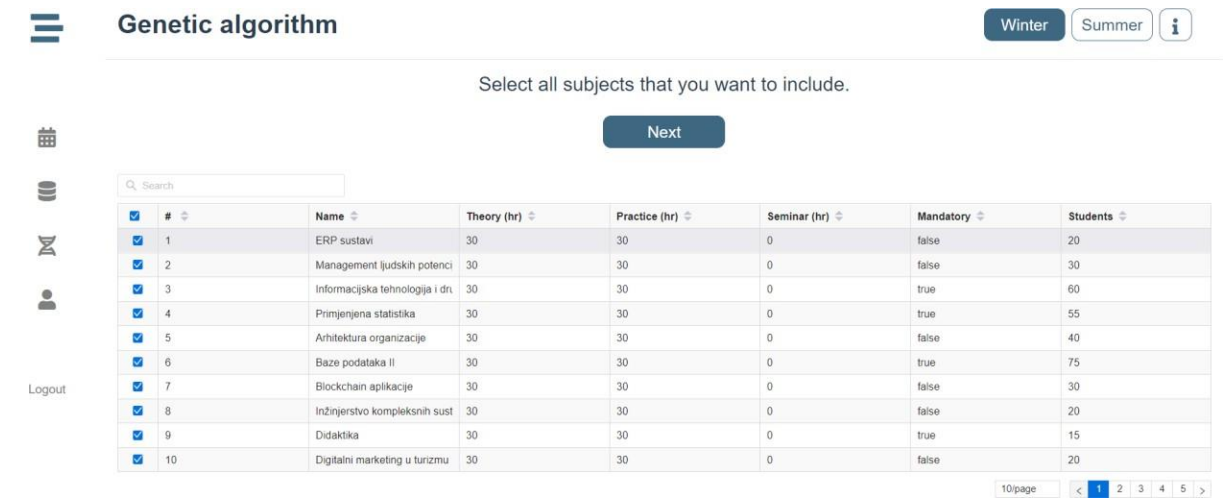


#	Name	Faculty	Students	Season
1	Preddiplomski 1	Fakultet Informatike Pula	100	winter
2	Preddiplomski 2	Fakultet Informatike Pula	70	summer
3	Preddiplomski 3	Fakultet Informatike Pula	75	winter
4	Preddiplomski 4	Fakultet Informatike Pula	50	summer
5	Preddiplomski 5	Fakultet Informatike Pula	60	winter
6	Preddiplomski 6	Fakultet Informatike Pula	50	summer
7	Diplomski inf 1	Fakultet Informatike Pula	55	winter
8	Diplomski inf 2	Fakultet Informatike Pula	55	summer
9	Diplomski inf 3	Fakultet Informatike Pula	30	winter
10	Diplomski nast 1	Fakultet Informatike Pula	15	winter

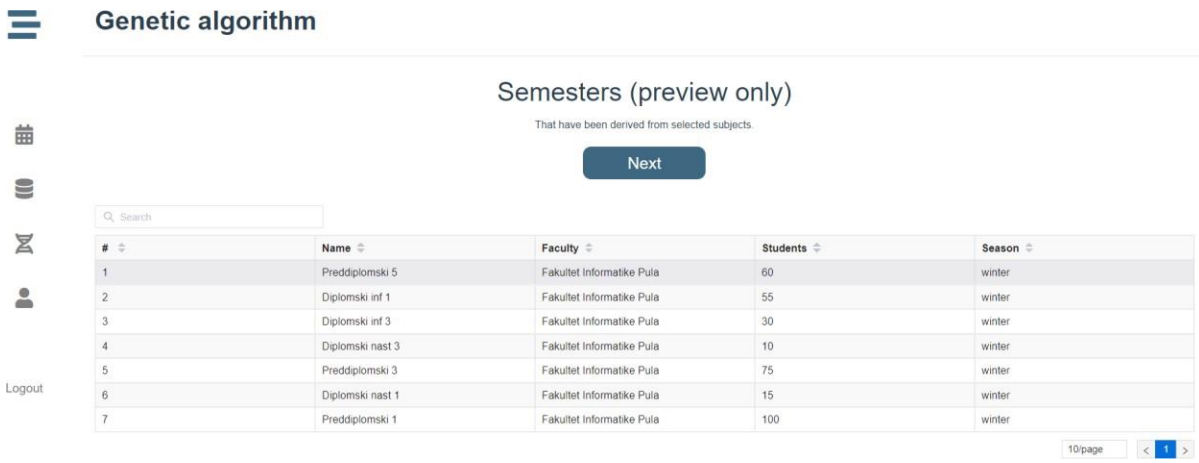
Slika 11 - Unos/Uređivanje/Brisanje podataka, Izvor: Vlastita aplikacija



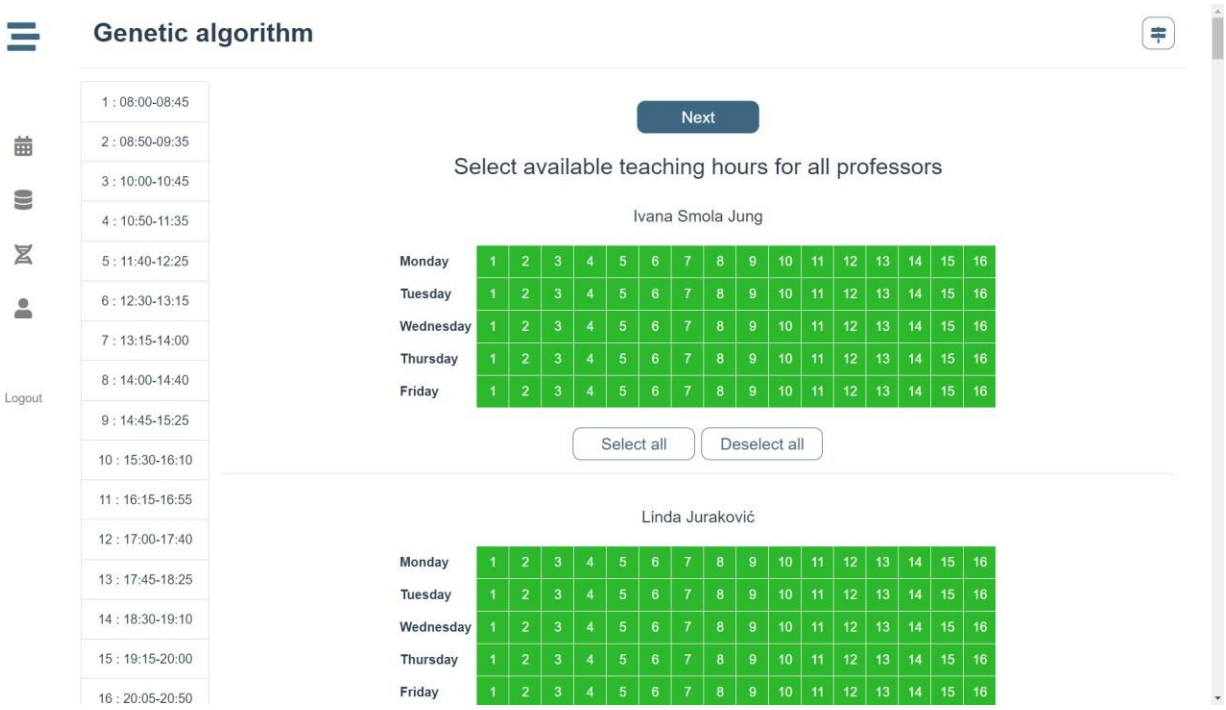
Slika 12 - Genetski algoritam unos tjedana
Izvor: Vlastita aplikacija



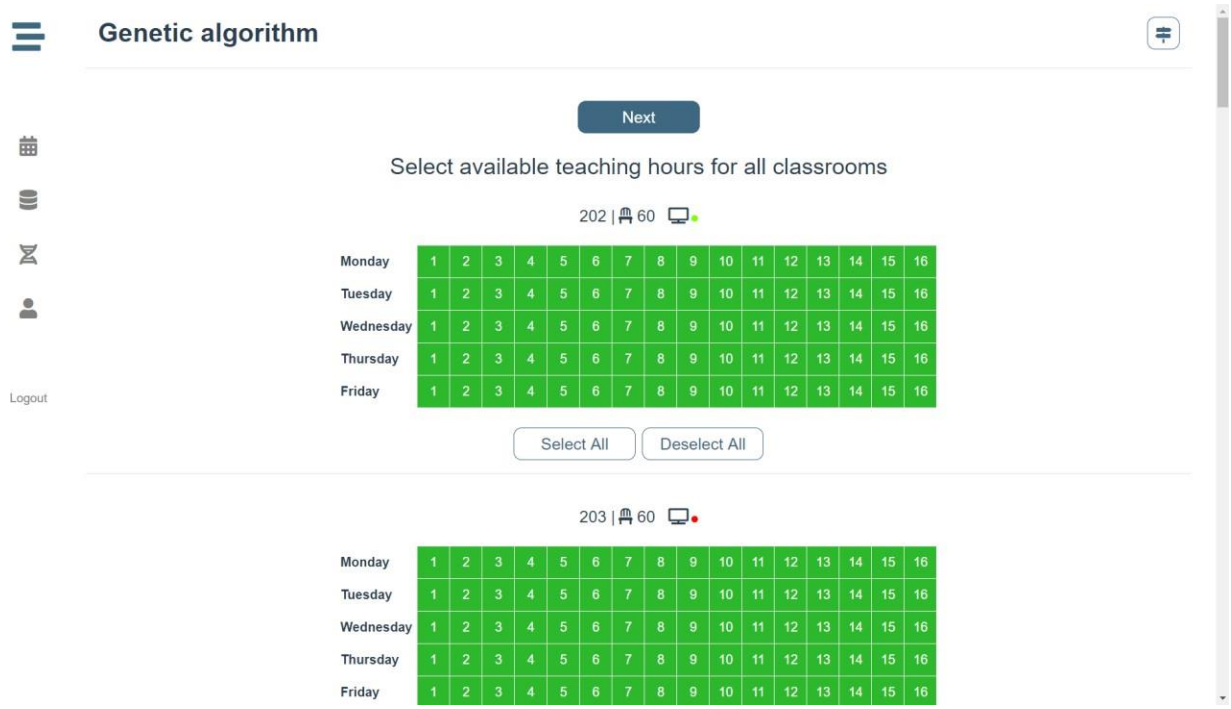
Slika 13 - Genetski algoritam unos predmeta
Izvor: Vlastita aplikacija



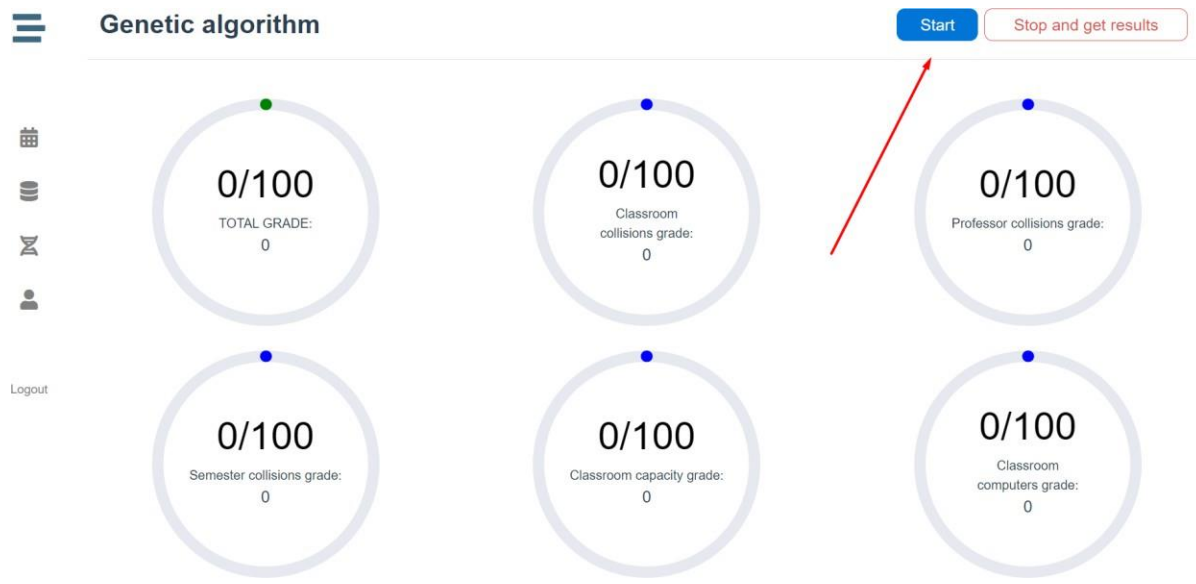
Slika 14 - Genetski algoritam, pregled semestara
Izvor: Vlastita aplikacija



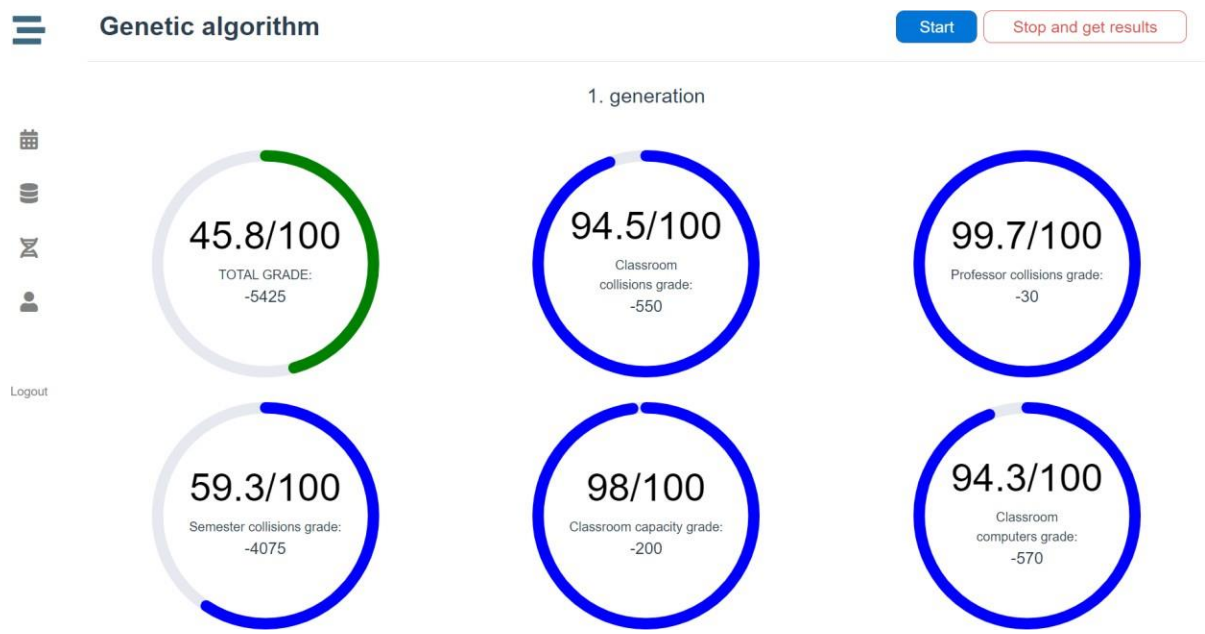
Slika 15 - Genetski algoritam, unos slobodnih sati profesora
Izvor: Vlastita aplikacija



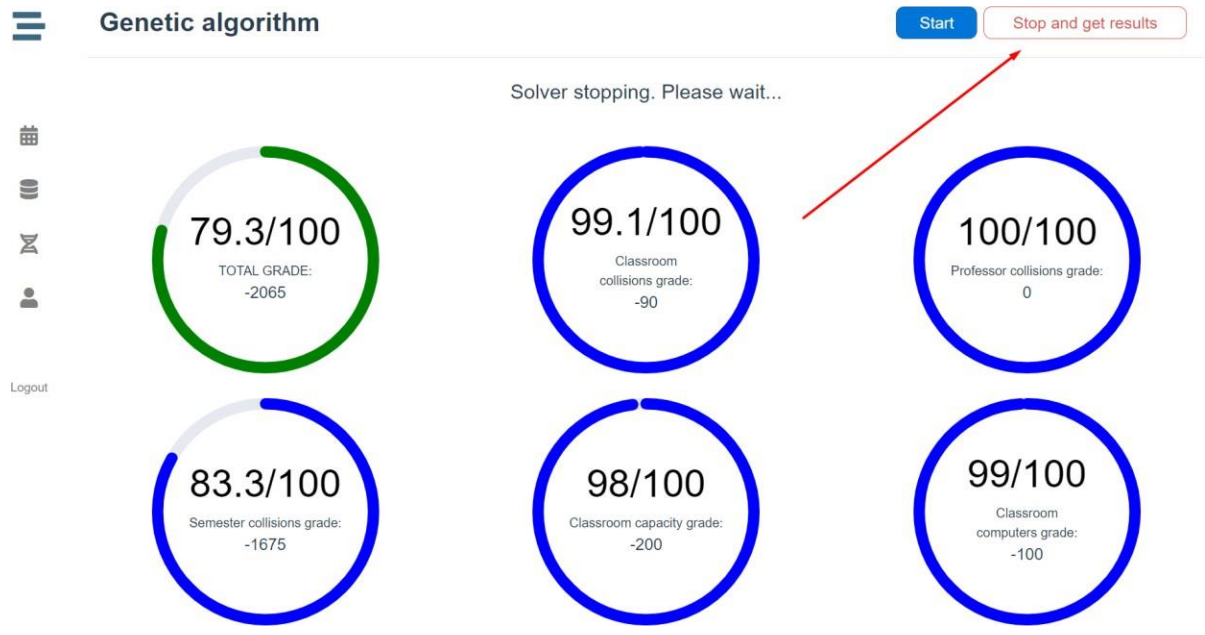
Slika 16 - Genetski algoritam, unos slobodnih sati dvorana
Izvor: Vlastita aplikacija



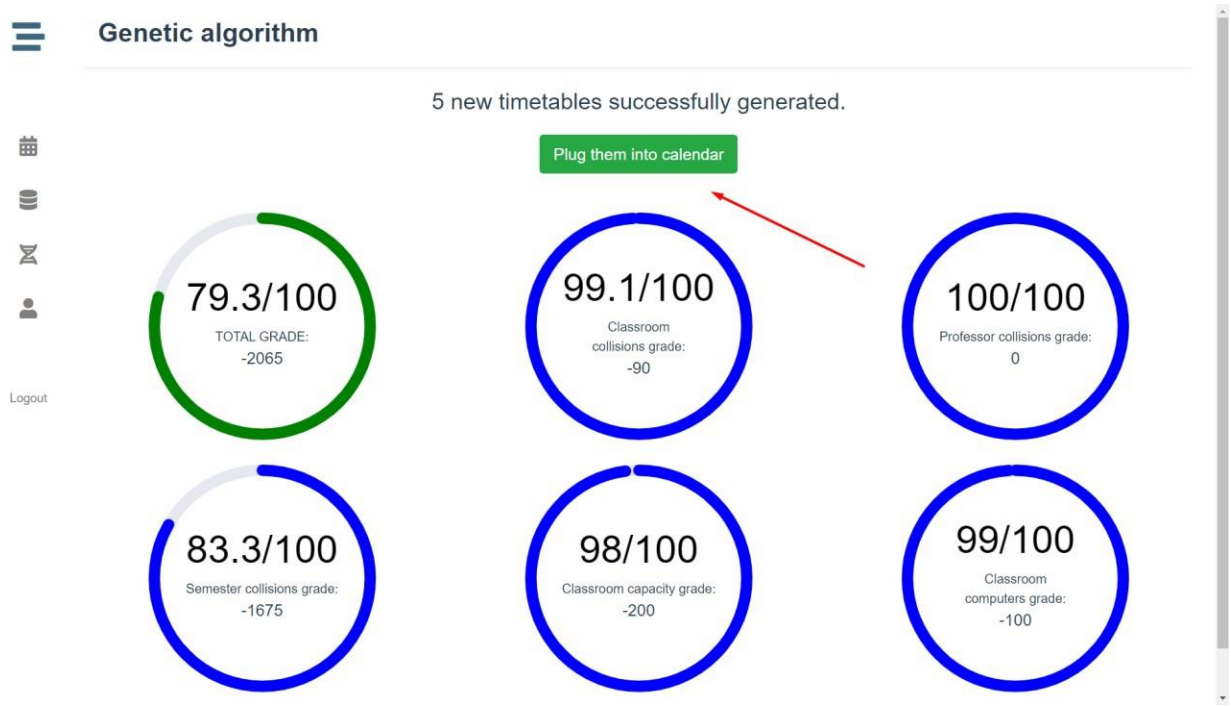
Slika 17 - Genetski algoritam, start
Izvor: Vlastita aplikacija



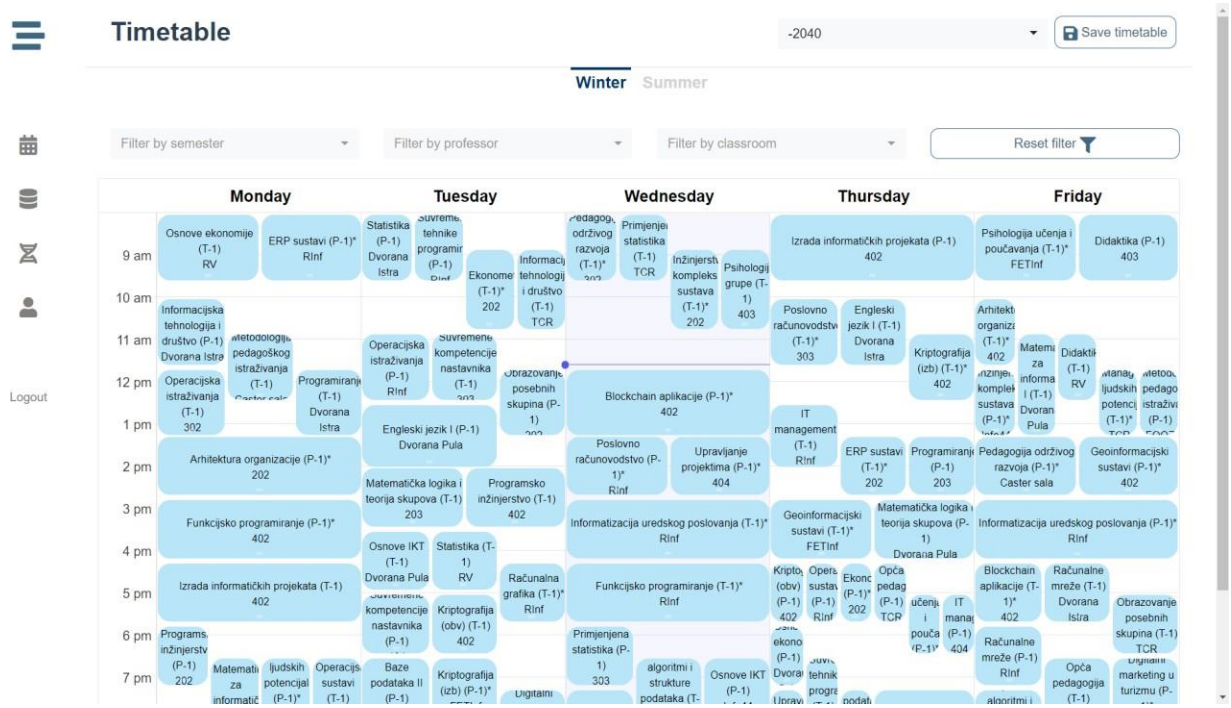
Slika 18 - Genetski algoritam, prvi rezultati
Izvor: Vlastita aplikacija



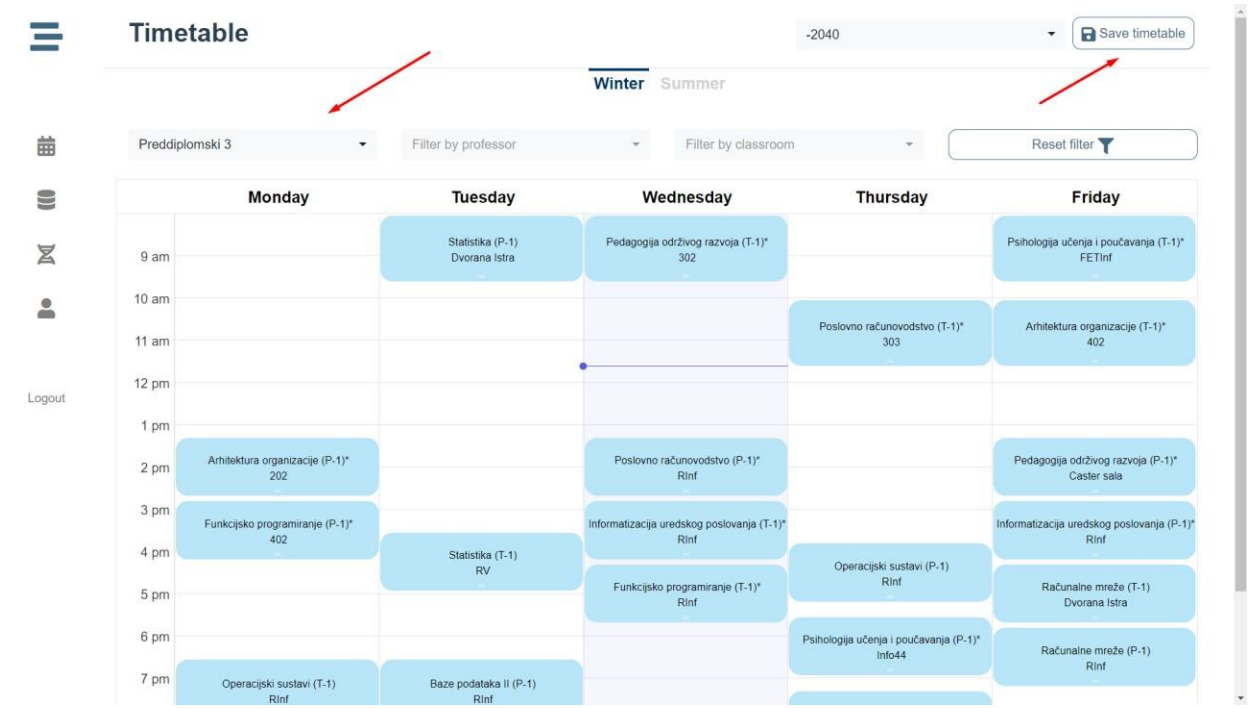
Slika 19 - Genetski algoritam, stop
Izvor: Vlastita aplikacija



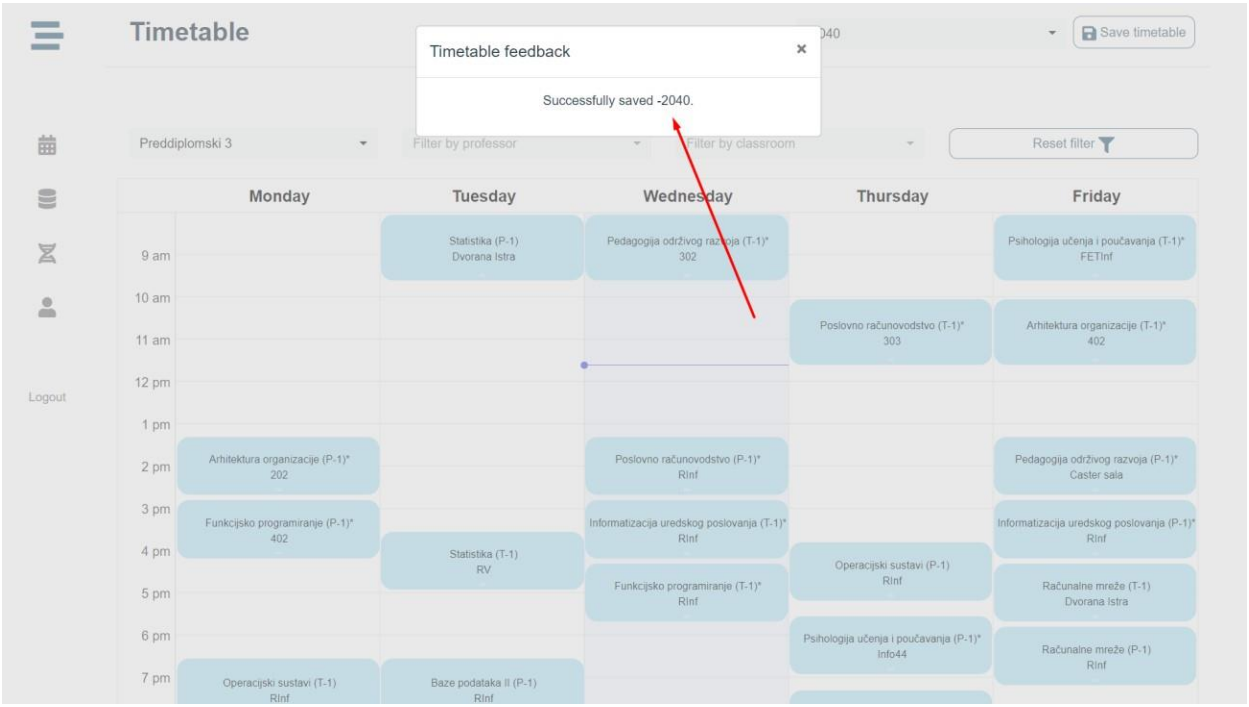
Slika 20 - Genetski algoritam, stavljanje u raspored
Izvor: Vlastita aplikacija



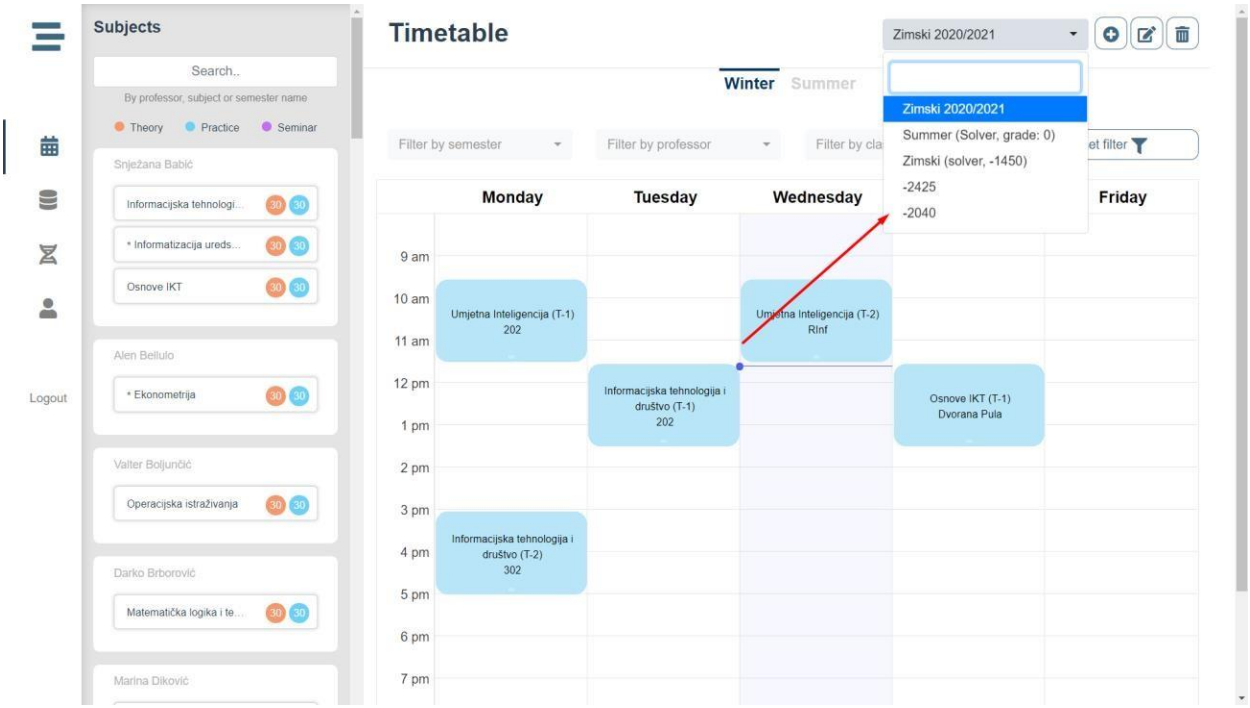
Slika 21 - Genetski algoritam, cijeli raspored
Izvor: Vlastita aplikacija



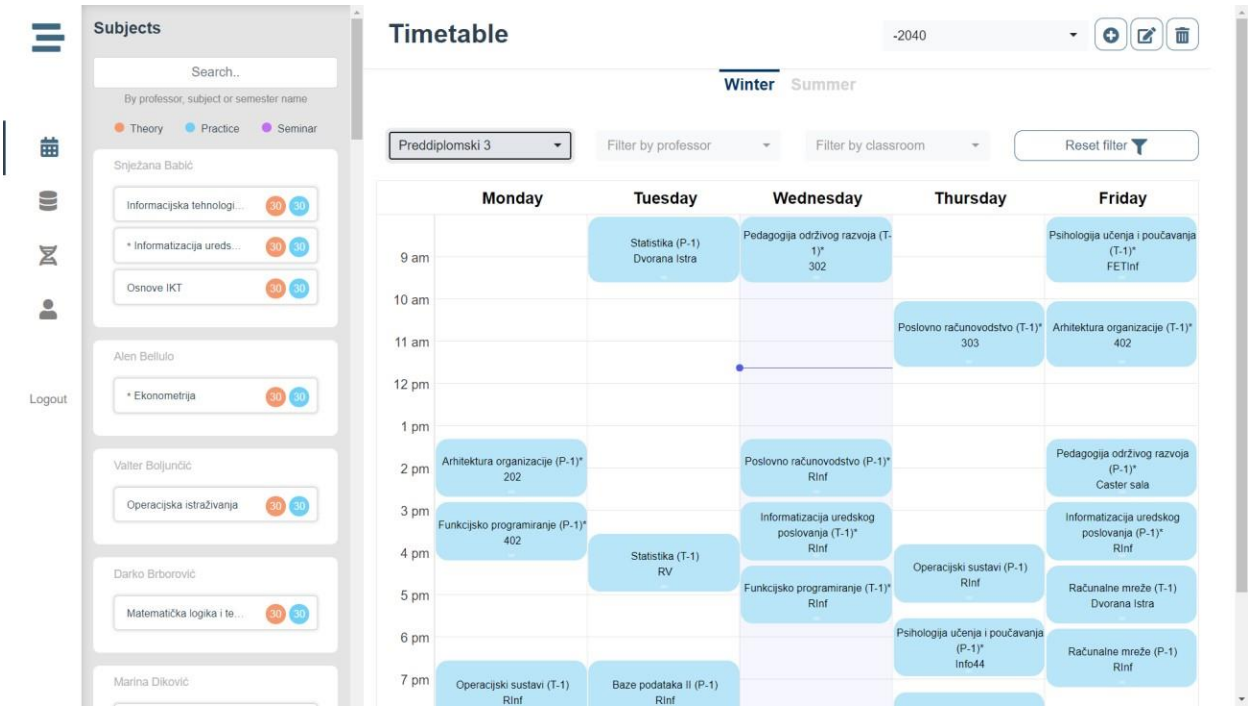
Slika 22 - Genetski algoritam, raspored filtriran po semestru
Izvor: Vlastita aplikacija



Slika 23 - Genetski algoritam, spremanje izgeneriranog rasporeda u bazu
Izvor: Vlastita aplikacija



Slika 24 - Početna stranica, odabir spremljenog izgeneriranog rasporeda kao početnog
Izvor: Vlastita aplikacija



Slika 25 - Početna stranica, filtriranje spremljenog izgeneriranog rasporeda po semestru
Izvor: Vlastita aplikacija

7. Zaključak

Trenutno nije poznat polinomni algoritam koji rješava probleme pronalaska optimalnog rješenja sa velikim brojem ograničenja koja su kvantitativno nepredvidiva. Razlog je taj što različite kombinacije ograničenja zahtijevaju različite puteve i pristupe, a tih kombinacija ima previše da se ispitaju u realnom vremenu. Genetski algoritam takve probleme rješava na način da stvori N kopija podataka, u iteracijama ih mijenja, ocjenjuje i u iduću iteraciju šalje samo K najbolje ocijenjenih. Sam po sebi ne garantira optimalnost, ali pametnim postavljanjem ograničenja i funkcije ocjenjivanja moguće je dobiti optimalna ili blizu optimalna rješenja nakon nekog vremena (koje ovisi o snazi servera). U ovoj aplikaciji implementiran je servis genetskog algoritma koji je povezan sa korisničkim sučeljem preko kojega korisnici mogu unijeti vlastita ograničenja i zatražiti rješenje. Također je implementirano i područje za ručno uređivanje rasporeda na način da korisnik može filtrirati, dodati, urediti, premjestiti ili brisati predmete s njega.

8. Popis literature i alata

Okviri

Vue JS: https://vuejs.org/v2/guide/ (9.9.2020.)

Express JS: https://expressjs.com/ (9.9.2020.)

Flask: https://flask.palletsprojects.com/en/1.1.x/ (9.9.2020.)

Vue komponente

Toast UI Calendar: https://ui.toast.com/tui-calendar/ (9.9.2020.)

Moment JS: https://momentjs.com/ (9.9.2020.)

Axios: https://github.com/axios/axios (9.9.2020.)

Vue dynamic table: https://github.com/TheoXiong/vue-table-dynamic (9.9.2020.)

Vue ellipse progress: https://github.com/setaman/vue-ellipse-progress (9.9.2020.)

Vue timepicker: https://github.com/phoenixwong/vue2-timepicker (9.9.2020.)

Vue datepicker: https://github.com/charliekassel/vuejs-datepicker (9.9.2020.)

Deployment

Free Vercel's 'now': https://vercel.com/ (9.9.2020.)

Free Heroku: http://heroku.com/ (9.9.2020.)

9. Prilozi

Github

Korisničko sučelje: https://github.com/jjurinci/schedulerui (9.9.2020.)

Servis za DML: https://github.com/jjurinci/scheduler_dml (9.9.2020.)

Servis za genetski algoritam: https://github.com/jjurinci/scheduler_solver (9.9.2020.)

10. Sažetak

Aplikacija rješava problem rasporeda koji se pojavljuje na fakultetima gdje je potrebno uzeti u obzir veliki broj činjenica i pravilno rasporediti predmete po dvoranama i vremenu. Sastoji se od korisničkog sučelja na kojem korisnik može unijeti sve podatke, ručno napraviti raspored i zatražiti rješenje genetskog algoritma za dana ograničenja. Genetski algoritam vraća raspored skupa sa njegovom ocjenom gdje gora ocjena predstavlja veće kršenje ograničenja. Moguće je izgenerirane rasporede spremiti u bazu podataka te ih u bilo kojem trenutku dalje ručno modificirati što uvelike olakšava proces izrade rasporeda imajući na umu da je dobar dio ograničenja već zadovoljen. Za kraj, korisnik može filtrirati raspored po semestru, profesoru ili dvorani i dobiti uvid u svaki individualni raspored na fakultetu.

Ključne riječi: raspored, ograničenja, genetski algoritam, baza podataka, ručna modifikacija, filtriranje

Application solves academic timetable problems where it's necessary to take into account a large number of constraints and optimally allocate classroom and time resources to respective subjects. It has an user interace where user can insert data, manually create timetables and request a generated timetable from genetic algorithm for given constraints. Genetic algorithm returns a timetable together with its grade where worse grade represents greater violation of constraints. It's possible to save generated timetables to database and later manually edit them which greatly eases the process of constraint satisfaction. User can also filter their timetable by semester, professor or classroom and get insight into any individual timetable on their faculty.

Keywords: timetable, constraints, genetic algorithm, database, manually edit, filter