УНИВЕРЗИТЕТ У БЕОГРАДУ
МАТЕМАТИЧКИ ФАКУЛТЕТ

Јелена Ј. Митровић

РАЗВОЈ АПЛИКАЦИЈЕ ЗА ВИЗУАЛИЗАЦИЈУ АКАДЕМСКЕ МРЕЖЕ МАСТЕР И ДОКТОРСКИХ ТЕЗА НА МАТЕМАТИЧКОМ ФАКУЛТЕТУ

мастер рад

Београд, 2026.

Ментор:

доц. др Мирјана Маљковић Ружичић, доцент
Универзитет у Београду, Математички факултет

Чланови комисије:

доц. др Ивана Танасијевић, доцент
Универзитет у Београду, Математички факултет

проф. др Јована Ковачевић, ванредни професор
Универзитет у Београду, Математички факултет

Датум одбране: _____

йородици

Наслов мастер рада: Развој апликације за визуализацију академске мреже мастер и докторских теза на Математичком факултету

Резиме: Овај рад представља развој веб апликације намењене визуелизацији менторских односа на Математичком факултету Универзитета у Београду. Апликација омогућава приказ повезаности између студената, ментора и чланова комисије, као и претрагу на основу различитих критеријума попут имена, смера и године одбране. Коришћењем система за управљање графовским базама података Neo4j, програмског језика Python и веб оквира Django за имплементацију серверске стране апликације, као и HTML и библиотеке D3.js за приказ података у корисничком интерфејсу, омогућено је интуитивно и ефикасно управљање и приказ академских односа. Циљ апликације је да олакша увид у академске повезаности и допринесе транспарентности академског процеса. Рад показује како се употребом савремених технологија могу решити специфични изазови у области визуализације података у академском контексту.

Кључне речи: графовске базе података, академска мрежа, neo4j, django, python, cypher, академско наслеђе, менторство, визуализација података, веб апликација

Садржај

Глава 1

Увод

Менторство има кључну улогу у академском напретку студената, посебно током израде мастер и докторских радова. Односи између ментора, студената и чланова комисије временом граде сложене структуре које је често тешко сагледати без адекватне визуелне подршке. Иако факултети воде евиденцију о овим подацима, они су најчешће представљени у облику текстуалних спискова, што отежава увид у целокупну мрежу академских веза и њихов развој.

Мотивисана потребом да се ова структура прикаже на интуитиван и визуелно прегледан начин, развијена је веб апликација која омогућава приказ и претрагу менторских односа на Математичком факултету. Користећи систем за управљање графовским базама података Neo4j, омогућен је природан модел повезивања чворова (студената и професора) и веза (менторства и чланства у комисији). Серверска страна је реализована у програмском језику Python, док је за клијентску страну коришћен HTML у комбинацији са савременим JavaScript библиотекама, међу којима је и библиотека D3.js, која је коришћена за визуелизацију графа и интерактивни приказ односа.

Апликација омогућава корисницима да претражују студенте по имену, смеру и години одбране, као и да визуелизују менторска стабла. Поред тога, омогућена је и претрага професора, уз приказ свих студената којима је професор био ментор или члан комисије, као и даље рекурзивно праћење академског наслеђа кроз генерације. Циљ овог рада је да прикаже како се помоћу савремених технологија могу решити реални проблеми у области управљања и приказа академских података.

У другом поглављу рада је описан развој графовских база података, њихова структура, предности и недостаци, као и примери примене у пракси.

Треће поглавље је посвећено систему за управљање графовском базом података Neo4j и упитном језику Cypher, при чему су приказане основне карактеристике и начини рада. Четврто поглавље садржи опис технологија које су коришћене у развоју апликације, укључујући програмски језик Python, оквир Django, као и библиотеку D3.js за интерактивне визуализације, као и начин њихове интеграције. Пето поглавље описује апликацију за визуализацију академске мреже, са акцентом на припрему података, креирање графовске базе и доступне функционалности апликације. У завршници рада дат је преглед и опис свих могућности које апликација пружа у циљу лакшег коришћења и даљег унапређења.

Глава 2

Графовске базе података

У овом поглављу дато је објашњење основних појмова везаних за графовске базе података, као и њихов значај и примена у савременим системима. ***Графовске базе података*** представљају савремен начин чувања и управљања подацима, који се ослања на принципе теорије графова. За разлику од традиционалних релационих база, које податке организују у табеле, графовски модел их представља помоћу чворова и веза, што омогућава природније представљање међусобних односа између ентитета. У графовском моделу, чворови представљају ентитете, попут особа, организација и појмова, док везе дефинишу односе између њих. И чворови и везе могу имати своје атрибуте, попут имена, типа или друге релевантне информације. Управо та структура чини графовске базе врло моћним алатом за анализу сложених и међусобно повезаних података. Захваљујући специјализованим језицима за упите над графовским базама података, попут језика Cypher [1] који се користи у системима за управљање графовским базама као што је Neo4j [2], рад са графовима је постао једноставан и ефикасан [3]. Претраге и анализе се могу извршавати брзо, чак и када су у питању комплексне мреже података. Овакав приступ је посебно користан у областима где су односи између података кључни, на пример, у анализи друштвених мрежа, системима за препоруке, откривању превара, па чак и у биомедицинским истраживањима. Графовске базе омогућавају не само боље разумевање података, већ и њихову визуелизацију на интуитиван начин.

2.1 Развој

Ране фазе

Идејни корени графовских база података потичу још из 18. века, када је Леонард Ојлер поставио темеље теорије графова кроз свој рад на познатом проблему мостова у Кенигсбергу [4]. Ипак, права примена ове теорије у информационим технологијама није започела све до средине двадесетог века, када су се појавили први рачунарски системи.

Током шездесетих година двадесетог века, базе података су углавном биле засноване на хијерархијском или мрежном моделу. Ови системи су омогућавали организацију података у облику стабала или мрежа, али нису били идеални за приказ и управљање сложеним везама између података. Теорија графова је тада још увек била претежно резервисана за академска истраживања, а не за практичну употребу у обради и анализи података [5].

Успон графовских база података

Почетком 21. века, експлозиван развој интернета, друштвених мрежа и система за препоруку (*eng. Recommendation Systems*) показао је ограничења традиционалних релационих база података, које нису биле у могућности да ефикасно обраде сложене и повезане податке. Овај период представља кључну прекретницу у развоју графовских база података. Графовске базе су убрзо постале популарне због својих способности да на природан и интуитиван начин моделују сложене структуре, као што су друштвене мреже и системи за препоруке. Овај успон технологија графовских база података настављен је појавом нових алата и решења који су одговарали на све веће захтеве за обрадом повезаних података. Систем за управљање графовским базама података Neo4j [2] представљен је 2007. године и сматра се једном од првих модерних графовских база података. Neo4j је омогућио ефикасно складиштење и претрагу података који су природно повезани, као што су корисници друштвених мрежа, трансакције или везе међу биолошким ентитетима.

Графовске базе података сада играју значајну улогу у многим индустријама, од финансија и биотехнологије до сајбер безбедности и анализе података. Поред Neo4j, најистакнутији актери на тржишту јесу ArangoDB [6], Amazon Neptune [7] и Microsoft Azure Cosmos DB [8], као и многи други специјализо-

вани системи.

2.2 Структура

Графови чине основу структуре графовских база података. Они се састоје од чворова (*eng. nodes*) и веза (*eng. edges*), док додатне информације о ентитетима и њиховим односима пружају својства (*eng. properties*) и ознаке (*eng. labels*) [5]. Овај модел база података омогућава природно представљање комплексних односа између података, што га чини погодним за примене као што су анализа друштвених мрежа, системи препорука, хијерархијске структуре и анализе великих скупова података.

Чворови

У графовским базама података, чворови представљају кључне елементе, то су основне јединице од којих се састоји граф. Сваки чвор има свој јединствени идентификатор (ID), а поред тога, може садржати различита својства, тј. атрибуте који га описују. На пример, ако чвор представља неку особу, могућа својства тог чвора су име, година рођења, адреса становања и слично. Оно што графовске базе чини посебно флексибилним јесте чињеница да чворови могу представљати практично било шта, нпр. људе, организације, пројекате, документе или чак апстрактне појмове. Управо та прилагодљивост структуре омогућава да се модел података лако прилагоди потребама домена у коме се користи.

Везе

У графовским базама података везе, тј. релације, повезују објекте. Оне описују односе између различитих ентитета и могу бити усмерене или неусмерене, у зависности од тога шта веза представља. Поред тога што повезују чворове, везе често садрже и додатне информације, тј. атрибуте, што омогућава прецизније описивање односа. Везе су подједнако важне као и сами чворови, јер кроз њих можемо анализирати односе и зависности унутар података. Ево неколико конкретних примера:

- Веза између чвора *Студент* и чвора *Курс* може означавати упис студента на одређени курс. Та веза може садржати додатне информације

као што су оцена коју је студент добио на курсу, као и семестар у коме је похађао курс.

- Веза између два чвора типа *Запослени* може представљати однос надређени - подређени у организационој структури фирме. Поред самог односа, веза може имати атрибут датум почетка сарадње или одговорности у односу.

Атрибути

Атрибути пружају додатне информације о чворовима и њиховим везама. Додавање атрибута омогућава софистицирану анализу података.

Ознаке

Ознаке се користе за категоризацију чворова у графу. Сваки чвор може имати једну или више ознака које га сврставају у одређену групу. На пример:

- Чвор са ознаком *Истраживач* може представљати особу укључену у научна истраживања.
- Чвор са ознаком *Институција* може означавати универзитет или истраживачки центар.

Ознаке омогућавају организацију података, чинећи их доступнијим за претраживање и анализу. На пример, могу се претраживати сви чворови са ознаком *Пројекат* који су повезани са одређеном институцијом, што поједностављује анализу скупова података.

2.3 Предности

Графовске базе података имају много предности, посебно у ситуацијама где је важно разумети и анализирати сложене односе међу подацима. Захваљујући својој специфичној структури и начину рада, оне се издвајају као одличан избор за апликације које захтевају ефикасност, флексибилност и интуитиван приступ обради информација.

За разлику од реалационих база података у којима се информације чувају у табелама које су повезане преко кључева, графовске базе користе чворове

и везе, што значајно поједностављује рад када су у питању повезани подаци. Представљање сложених односа између података у релационим системима често захтева вишеструко спајања табела, што може отежати рад са веома повезаним подацима и утицати на перформансе. Управо због ових ограничења релационог модела, графовске базе података се издвајају као погоднije решење за рад са подацима који имају сложене међусобне везе.

У наставку су наведене неке од најважнијих предности графовских база података [5]:

- ***Природно моделовање односа:*** графовске базе података омогућавају да се односи између ентитета моделују на интитиван и јасан начин. За представљање ентитета користе се чворови, а за представљење односа међу њима везе. Оваква струкура посебно је погодна у системима који се ослањају на сложене и променљиве мреже. Овакво моделовање односа смањује потребу за сложеним релационим моделима, што додатно олакшава прилагођавање базе података специфичним потребама.

- ***Ефикасност у претрагама сложених веза:*** графовске базе података су оптимизоване за брзо и ефикасно извођење упита који укључују вишеструке нивое односа, као што су претраге најкраћих путева (*eng. shortest path*) или идентификација сличности између ентитета. За разлику од релационих база, где сложени упити често захтевају спајање великог броја табела, у графовским базама овакви упити се директно изводе преко повезаних чворова и грана, чиме се значајно смањује време извршавања упита.

- ***Флексибилност и скалабилност шеме:*** за разлику од традиционалних релационих база података, графовске базе не захтевају унапред дефинисану шему. То значи да је могуће додавати нове типове чворова, атрибута и веза без потребе за преправком постојеће структуре базе. Ова флексибилност чини графовске базе идеалним за апликације са често променљивим захтевима или за системе у којима подаци и односи постају познати тек током времена.

- ***Побољшана визуелизација и анализа података:*** графовске базе података пружају могућности за јасну и прегледну визуелизацију података и њихових односа. Графички прикази омогућавају корисницима

лакше откривање и уочавање образаца, аномалија и веза у подацима, што може бити од кључног значаја у областима као што су истраживање тржишта, детекција превара или аналитика друштвених мрежа.

- ***Погодност за специфичне апликације:*** графовске базе података су посебно корисне у сценаријима где подаци природно формирају мреже, као што су системи за управљање препорукама, аналитика друштвених мрежа, оптимизација транспортних рута и биолошка истраживања. Њихова способност да директно раде са повезаним подацима пружа конкурентску предност у овим областима, јер омогућавају ефикасну и прилагођену обраду информација у стварном времену.

2.4 Недостаци

Иако графовске базе података нуде многе предности, оне нису увек најбоље могуће решење за сваку врсту апликације. У неким случајевима, графовске базе се суочавају са одређеним изазовима и недостацима, који морају бити узети у обзир при избору одговарајуће базе података за специфичне захтеве [9]. У наставку су приказани неки од главних недостатака графовских база података:

- ***Мањак стандардизације:*** Док релационе базе података користе стандардизовани и широко прихваћени упитни језик SQL, графовске базе података често користе сопствене упитне језике, као што је на пример Cypher у систему за управљање базама података Neo4j [2]. Мањак стандардизације доводи до проблема у интеграцији са другим системима или миграцији једног система на други. Програмери морају савладати специфичне упитне језике који се користе у различитим графовским базама података, па развој и одржавање апликације може бити изазовно.

- ***Ограничена скалабилност:*** Графовске базе података могу наићи на потешкоће када се користе за обраду веома великих количина података или обимних графова. Како број чворова и веза расте, постаје све теже ефикасно организовати податке, што може утицати на перформансе система. Управљање таквим графовима захтева додатне ресурсе и може представљати изазов, нарочито у ситуацијама када је потребна брза обрада или комплексне анализе.

- ***Мање погодне за трансакционе системе:*** Графовске базе нису увек оптималне за апликације које захтевају висок ниво трансакционе обраде и гаранције попут очувања ACID својстава [1]. Релационе базе података су прилагођене апликацијама које често обављају сложене трансакције, јер омогућавају ефикасно управљање великим бројем паралелних операција и чување података у табелама са унапред дефинисаним структурама. Са друге стране, графовске базе су мање ефикасне када се захтева брз одговор у системима са великим бројем паралелних упита.

- ***Ограничена ефикасност при анализи великих скупова података:*** Иако графовске базе података омогућавају ефикасну анализу веза и односа међу чворовима, њихова примена у контексту обраде великих количина података који нису у директној вези са графовском структуром може бити ограничена. Наиме, сложенији аналитички упити, као што су агрегације или операције које подразумевају обраду целокупног скупа података без ослањања на графове, чешће се успешније реализују у оквиру релационих база података. Релационе базе, захваљујући својој дугогодишњој оптимизацији за рад са великим скупом података и напредним могућностима индексирања и оптимизације упита, често представљају прикладнији избор у таквим случајевима. Специфична структура података у графовским базама може представљати ограничење у погледу перформанси приликом обраде обимних и комплексних задатака.

Ови недостаци указују на то да иако графовске базе података нуде изузетне предности у одређеним сценаријима, морамо бити свесни њихових ограничења и да у неким случајевима можда неће бити најбољи избор у односу на друге врсте база података. Стога је важно анализирати специфичне захтеве апликације пре него што се одлучи за имплементацију графовске базе података.

[1] ACID својства представљају скуп особина — Atomicity (атомичност), Consistency (конзистентност), Isolation (изолованост) и Durability (трајност) — које гарантују поузданост трансакција у базама података.

2.5 Примери употребе

Графовске базе података, због своје флексибилности и природне способности да моделују комплексне односе, налазе примену у различитим доменима. Њихова структура заснована на чворовима и гранама омогућава интуитивно представљање података и анализу сложених односа. У наставку су приказане најзначајније области примене графовских база података:

- ***Биомедицина*** [17]: Графовске базе података нашле су примену у области биомедицине, за анализу комплексних биолошких мрежа, попут мреже протеина, гена или метаболичких путева. Графови омогућавају откривање повезаности између гена који могу бити укључени у одређене болести, што је кључно за развој нових техника лечења.

- ***Криминалистика и форензика*** [18]: Графовске базе података олакшавају праћење веза између осумњичених, организација и догађаја. Захваљујући могућностима визуализације и анализе, лакше је уочити криминалне мреже, препознати кључне актере и чак предвидети могуће будуће активности. Овакав приступ чини графовске базе података изузетно корисним алатом у борби против организованог криминала.

- ***Финансије и детекција превара*** [19]: Графови се такође користе у финансијском сектору за откривање превара анализом трансакција и повезаности између корисника. Идентификација сумњивих активности, попут прања новца, олакшава се коришћењем графовских алгоритама који откривају проблеме и неправилности у подацима.

- ***Системи за препоруке*** [20]: Графовске базе података се користе за персонализацију препорука анализом повезаности корисника и садржаја. На пример, помоћу алгоритама као што су *PageRank* или *personalized walks*, могу се идентификовати релевантни производи, филмови или књиге које одговарају интересовањима корисника. Ови алгоритми се користе у компанијама као што су Netflix и LinkedIn — Netflix користи personalized walks за препоруку филмова и серија на основу корисничких преференција и понашања, док LinkedIn примењује PageRank варијанте за препоруке нових веза и садржаја корисницима. Овако добијене препоруке су значајно прецизније у поређењу са традиционалним приступима.

- ***Друштвене мреже*** [21]: Графовске базе података веома су погодне за моделирање друштвених мрежа, јер на природан начин осликавају односе међу корисницима. Сваки корисник представља један чвор, док су везе између њих различити облици међусобних односа – пријатељство, праћење или сличне интеракције. Оваква структура омогућава једноставнију и ефикаснију анализу, попут проналажења најутицајнијих корисника или предлагања нових пријатеља.

- ***Телекомуникације*** [22]: У овој индустрији, графовске базе података су корисне за анализу мрежа корисника и идентификацију загушења у систему. Омогућавају оптимизацију ресурса за пружање квалитетнијих услуга, као и боље планирање инфраструктуре.

- ***Образовање*** [23]: У образовању, графовске базе података могу да моделују односе између студената, предавача, курсева и ресурса. Овакве анализе могу пружити увид у обрасце учења, омогућити персонализацију образовних искустава и унапредити стратегије подучавања.

Уз овако широку примену и способност да ефикасно управљају сложеним подацима, графовске базе података постају незаменљив алат у савременом пословању и истраживању. Њихова популарност расте са све већим потребама за анализом великих количина повезаних података, што их чини идеалним избором за решавање многих изазова у савременом свету.

Глава 3

Neo4j

Neo4j [2] је један од најзначајнијих система за управљање графовским базама података. Дизајниран је тако да омогући ефикасно складиштење, претрагу и манипулацију подацима у графовском формату. Ова особина чини Neo4j изузетно погодним за апликације које се баве сложеним везама између података. Развијен је 2007. године од стране компаније која је тада носила име *Neo Technology*. Од тада је стекао велику популарност и данас се сматра једним од најзаступљенијих система за управљање за рад са графовима.

Архитектура Neo4j обухвата следеће кључне компоненте:

- ***Графовска база података***: Главни елемент система Neo4j је графовска база података која садржи чворове, гране, атрибуте и лабеле. Чворови представљају ентитете, док гране представљају везе између њих.
- ***Физичка организација података:*** Интерно, Neo4j организује податке тако да чворове, везе и атрибуте чува у посебним типовима датотека. Оваква структура омогућава ефикасно управљање великим количинама података и бржи приступ подацима.
- ***Индексирање и претрага:*** Користи напредне методе индексирања као што су B-стабло индекси за брзо претраживање по вредностима атрибута, индекси потпуног текста за претрагу текстуалних података, као и нативни графовски индекси који убрзавају обилазак веза између чворова. Ове методе омогућавају брзо претраживање чворова и веза, чиме се побољшавају перформансе система.

- ***Кеширање:*** Има уграђен механизам за паметно кеширање података у меморији ради бржег приступа најчешће коришћеним чворовима и везама.

- ***Клијент-сервер архитектура:*** Neo4j може бити инсталиран као самосталан сервер или као дистрибуирана архитектура, што омогућава скалабилност и лакшу интеграцију у шире системе. Корисници могу користити Neo4j путем REST API-ја, Java API-ја или других доступних клијената.

3.1 Упитни језик Cypher

Cypher [1] је упитни језик специјално дизајниран за рад са графовским базама података, развијен како би се поједноставио рад са графовима и омогућио брз и ефикасан начин за упите, манипулацију подацима и анализе. Својом синтаксом подсећа на SQL, при чему је прилагођен раду са везама и односима између различитих ентитета, што га чини погодним чак и за кориснике који нису нужно експерти у раду са базама података. Захваљујући томе, корисници могу ефикасно претраживати, анализирати и управљати повезаним подацима унутар графовских база.

У поређењу са упитним језиком SQL који манипулише табелама које садрже редове и колоне, Cypher манипулише графовима, односно чворовима и везама. Док SQL користи наредбе попут `SELECT`, `INSERT` и `UPDATE` за рад са подацима у табелама, Cypher користи сличне команде, али се фокусира на рад са графовским структурама. Синтаксно, Cypher је ближи природном језику и омогућава лакше представљање графова. На пример, упит за проналажење свих чворова који су повезани са одређеним чвором приказан је у Листингу 3.1.

Листинг 3.1: Проналажење повезаних чворова

```
MATCH (a)-[:POZNAJE]->(b)
RETURN a, b
```

Овакав приступ је једноставнији и читљивији у поређењу са SQL упитима који користе спајања табела.

Основна синтакса

У наставку су објашњене основне компоненте синтаксе језика Cypher и на конкретним примерима је приказано како се праве чворови, успостављају везе и извршавају једноставни упити.

Cypher упити се обично састоје од више клаузула које омогућавају претрагу, филтрирање, манипулацију и приказ података у граф бази. Најчешће коришћене клаузуле су:

- ***MATCH*** — клаузула за проналажење чворова и веза у бази података на основу задатог обрасца.
- ***WHERE*** — клаузула за филтрирање резултата добијених из `MATCH` клаузуле на основу дефинисаних услова.
- ***RETURN*** — клаузула која дефинише које податке упит враћа као резултат.
- ***CREATE*** — клаузула за креирање нових чворова и веза у графу.
- ***MERGE*** — клаузула за проналажење одређеног обрасца у бази или његово креирање уколико не постоји, чиме се избегава настанак дупликата.
- ***SET*** — клаузула за додавање нових или измену постојећих својстава чворова и веза.
- ***DELETE*** — клаузула за брисање чворова или веза из графа.
- ***WITH*** — клаузула за прослеђивање резултата из једног дела упита у други, уз могућност агрегирања, трансформације или додатног филтрирања података.
- ***ORDER BY*** — клаузула за сортирање резултата по једном или више својстава у растућем или опадајућем редоследу.
- ***LIMIT*** — клаузула за ограничавање броја резултата које упит враћа.

Чворови се креирају коришћењем синтаксе ***(:Label)***, где лабела одређује тип чвора. Везе се успостављају помоћу синтаксе ***-[:RELATION]***, где тип унутар угластих заграда дефинише природу везе између чворова.
Следећи примери показују основне упите у Cypher-у за претрагу и унос података:

- **Креирање чвора:** Наредба за прављење чвора са атрибутом ‘ime‘ приказана је у Листингу 3.2.

Листинг 3.2: Креирање чвора

```
CREATE (o:Osoba {ime: 'imeOsobe'})
```

- **Креирање везе између чворова:** Наредба за креирање везе између два чвора приказана је у Листингу 3.3.

Листинг 3.3: Креирање везе између чворова

```
CREATE (a:Osoba {ime: 'imeOsobe1'})
       -[:PRIJATELJ]->
       (b:Osoba {ime: 'imeOsobe2'})
```

- **Претрага чворова:** Упит за претрагу свих чворова са ознаком ‘Osoba‘ приказан је у Листингу 3.4.

Листинг 3.4: Претрага чворова

```
MATCH (n:Osoba)
RETURN n
```

- **Претрага са условом:** Упит за претрагу чворова са одређеним именом приказан је у Листингу 3.5.

Листинг 3.5: Претрага чворова са условом

```
MATCH (n:Osoba)
WHERE n.ime = 'imeOsobe'
RETURN n
```

- **Претрага веза између чворова:** Упит за претрагу веза између особа приказан је у Листингу 3.6.

Листинг 3.6: Претрага веза између чворова

```
MATCH (a:Osoba)-[:PRIJATELJ]->(b:Osoba)
RETURN a, b
```

Напредне функционалности

У овој секцији су размотрене напредне функционалности језика Cypher, укључујући агрегатне функције, филтрирање, спајање података, као и упите са вишеструким везама.

Агрегатне функције

Cypher подржава разне агрегатне функције које омогућавају анализу података у графу. Неке од најчешће коришћених функција су:

- **COUNT()** – враћа број чворова или веза који задовољавају услове упита.
- **AVG()** – израчунава просечну вредност за нумеричке податке.
- **SUM()** – израчунава збир нумеричких вредности.
- **MIN()** – враћа најмању вредност.
- **MAX()** – враћа највећу вредност.

Пример коришћења агрегатних функција приказан је у Листингу 3.7.

Листинг 3.7: Коришћење агрегатних функција

```
MATCH (n:Osoba)
RETURN COUNT(n) AS brojOsoba,
       AVG(n.godine) AS prosekGodina
```

Овај упит враћа број свих чворова типа `Osoba` у бази и просечну старост тих особа.

Филтрирање и спајање података

Cypher омогућава филтрирање података помоћу клаузуле `WHERE`, као и спајање услова помоћу логичких оператора. Најчешћи логички оператори су:

- `AND` - користи се за спајање више услова.
- `OR` - користи се за алтернативне услове.
- `NOT` - користи се за негирање услова.

Пример филтрирања са сложеним условима дат је у Листингу 3.8.

Листинг 3.8: Филтрирање са сложеним условима

```
MATCH (n:Osoba {godine: 30, grad: 'Beograd'})
RETURN n.ime, n.godine
```

Овај упит филтрира особе које су старије од 30 година и живе у Београду, а затим враћа њихова имена и старост.

Упити са вишеструким везама

Cypher омогућава рад са сложеним структурама графа, укључујући упите који укључују више веза између чворова. Упити са вишеструким везама могу бити корисни када је потребно анализирати сложене односе између чворова.

Пример упита са вишеструким везама приказан је у Листингу 3.9.

Листинг 3.9: Упит са вишеструким везама

```
MATCH (a:Osoba)-[:PRIJATELJ]->(b:Osoba)-[:PRIJATELJ]->(c:Osoba)
WHERE a.ime = 'Marko'
RETURN c.ime AS zajednickiPrijatelj
```

Овај упит тражи пријатеље особе `Marko`, а затим проналази пријатеље тих пријатеља. На крају, враћа имена особа које су заједнички пријатељи са особом `Marko`.

Упити са рекурзијом

Упити са рекурзијом у језику Cypher омогућавају да се извршавају претраге кроз граф са непознатим бројем нивоа веза између чворова. Овај тип упита је посебно користан када желимо да пронађемо све чворове који су повезани на било којем дубинском нивоу одређеним типом везе, односно када је потребно рекурзивно проћи кроз граф.

Cypher користи оператор * који означава варијабилан број понављања веза унутар једне путање. На пример, образац *[:PRIJATELJ*]* означава да се прати произвољан број узастопних веза типа *PRIJATELJ*.

Претпоставимо да желимо да пронађемо све особе које су директно или индиректно пријатељи особе `Marko`, без обзира на број корака у вези пријатељства. Такав упит приказан је у Листингу 3.10.

Листинг 3.10: Рекурзивна претрага пријатеља

```
MATCH (a:Osoba {ime: 'Marko'})-[:PRIJATELJ*1..]-(prijatelj)
RETURN DISTINCT prijatelj.ime AS sviPrijatelji
```

У делу упита `[:PRIJATELJ*1..]` ознака `PRIJATELJ` представља тип везе који се прати у графу, док симбол `*` означава да се та веза може поновити више пута. Број `1` дефинише минималан број понављања везе, односно да мора постојати бар једна веза типа `PRIJATELJ`, док запис `..` означава да горња граница дубине претраге није ограничена. На овај начин упит проналази све особе које су повезане са особом `Marko` једним или више корака преко веза типа `PRIJATELJ`.

Глава 4

Технологије коришћене у развоју апликације

У овом поглављу биће представљене технологије коришћене за развој апликације, са посебним освртом на њихову улогу, интеграцију и допринос крајњем решењу. Фокус овог поглавља је на технологијама, док ће детаљи о функционалностима саме апликације бити обрађени у наредном поглављу.

4.1 Програмски језик Python

У развоју система значајну улогу има програмски језик Python, који пружа подршку за рад са графовском базом података Neo4j и имплементацију функционалности на серверској страни. Захваљујући једноставној синтакси, великом броју доступних библиотека и могућности интеграције са различитим технологијама, Python представља погодан избор за развој система. За развој серверске стране апликације коришћен је Django оквир који омогућава брзо успостављање скалабилног система и ефикасно управљање свим процесима на серверској страни.

4.2 Оквир Django

Веб оквир Django [10] оквир одабран је као главна технологија за развој система због својих бројних предности које укључују брз, ефикасан и сигуран

развој апликација. Његов ORM [10] [1] олакшава рад са релационим базама података, а погодан је и за коришћење у комбинацији са графовском базом података.

Један од кључних аспеката Django оквира јесте безбедност. Django долази са уграђеним заштитама од најчешћих веб претњи, као што су SQL инјекције [2], крађа сесија на страни клијента, лажирање захтева и отмица кликова, што програмерима омогућава висок ниво сигурности без потребе за додатним сложеним имплементацијама. Поред безбедности, Django омогућава и брз развој захваљујући концепту „укључене свеобухватне функционалности“, који подразумева богат скуп уграђених алата и компоненти спремних за употребу.

Све ове карактеристике чине Django изузетно погодним избором за развој савремених веб система, јер омогућава ефикасно управљање подацима, одржавање високог нивоа сигурности и лако проширивање функционалности [10].

Архитектура Django оквира

Структура апликације имплементиране коришћењем оквира Django заснива се на принципу раздвајања одговорности, што омогућава чист и организован код. У самом језгру апликације налазе се модели, који дефинишу структуру података. Сваки **модел** представља један ентитет у систему. На тај начин, подаци у бази су повезани са Python објектима, што омогућава једноставније манипулисање и приступ подацима.

С друге стране, **погледи** представљају компоненту задужену за обраду корисничких захтева. Они примају захтеве од корисника, одређују који подаци су потребни и прослеђују их одговарајућем слоју за приказ. Понекад се ови подаци враћају у облику HTML [15] страница, а понекад као JSON [24] одговори, што омогућава комуникацију са клијентским апликацијама или другим системима. У том смислу, погледи служе као веза између базе података и корисничког интерфејса.

[1] ORM (Object-Relational Mapping) представља технику која омогућава програмерима да приступају и управљају подацима у релационим базама података користећи објектно-оријентисане парадигме.

[2] SQL инјекција представља безбедносни напад на базу података у којем се кроз улазне параметре апликације убацује злонамерни SQL код, чиме се може нарушити интегритет, поверљивост или доступност података.

Шаблони су одговорни за генерисање HTML страница. Када подаци стигну из погледа, шаблони их претварају у визуелни формат који је разумљив кориснику. Они омогућавају приказ динамичког садржаја и прилагођавање интерфејса у складу са подацима који се добијају из апликације.

4.3 D3.js: Интерактивне визуелизације у веб апликацијама

D3.js [11] је JavaScript библиотека која омогућава прављење динамичких и интерактивних визуализација података у веб апликацијама. Њена главна предност јесте способност да управља великим скуповима података, као и висок ниво прилагодљивости дизајна конкретним подацима. Библиотека D3.js омогућава детаљну манипулацију елементима у DOM [3] структури [12] кроз SVG (eng. Scalable Vector Graphics) [13], Canvas [14] и HTML [15], чиме се постиже висок ниво интерактивности и визуелне презентације података.

Једна од особина D3.js библиотеке је њена лака интеграција у веб апликације које користе оквир Django на страни сервера. У таквим апликацијама, подаци се обрађују у програмском језику Python на серверу и затим се прослеђују клијенту у JSON формату. Овај приступ омогућава библиотеци D3.js да преузме податке и генерише интерактивне графичке приказе директно у прегледачу. На пример, Django на серверској страни може генерисати JSON податке користећи Django REST Framework или кроз Django template систем, након чега D3.js користи те податке за приказ графикона, дијаграма и других визуализација. Ова интеграција омогућава високу динамику апликација и флексибилност у приказу података [16].

Предности коришћења библиотеке D3.js

Библиотека D3.js нуди многе предности приликом визуелизације података. Неке од њих су:

- **Флексибилност и прилагодљивост:** Омогућава потпуну контролу над бојама, облицима, анимацијама и распоредом елемената.

[3]DOM (eng. Document Object Model) је програмски интерфејс који представља HTML или XML документ као структуру објеката, омогућавајући скриптама да динамички мењају садржај, структуру и стил документа.

- **Интерактивност:** Омогућава различите облике корисничке интеракције, као што су задржавање миша изнад елемената, кликови, увећавање и померање приказа.
- **Ефикасност при раду са великим скуповима података:** Омогућава оптимизован и брз приказ података у реалном времену, чак и када се ради са великим количинама информација.
- **Једноставна интеграција са серверском страном апликације:** Омогућава лако преузимање и визуелизацију података из Django апликација путем REST API-ја.

4.4 Интеграција технологија

Комбинација D3.js библиотеке са оквиром Django и програмским језиком Python ствара одличан екосистем за развој модерних апликација које захтевају динамичне и интерактивне визуализације података. Библиотека D3.js омогућава висок ниво флексибилности и контроле, док оквир Django и програмски језик Python омогућавају ефикасно управљање и обраду података. Ова интеграција не само да побољшава корисничко искуство, већ такође омогућава развој апликација које могу да руководе великим количинама података са великим степеном интерактивности.

Глава 5

Апликација за визуализацију академске мреже

У овој глави је представљена апликација развијена у оквиру мастер рада која омогућава праћење и анализу менторства на Математичком факултету. Детаљно су описани подаци коришћени у раду, имплементација кључних делова система, као и могућности које апликација пружа.

5.1 Припрема података и прављење графовске базе података

Подаци коришћени за креирање базе података представљају евиденцију студената Математичког факултета који су одбранили мастер или докторску тезу по принципима Болоњског процеса. Подаци су иницијално били организовани у CSV формату, са пољима која укључују број индекса, име и презиме студента, студијски програм, наслов рада, тип завршног рада, датум одбране, ментора и чланове комисије. Да би се од података у овом облику могла направити графовска база података, било је неопходно извршити њихову обраду и конверзију у одговарајући облик.

У графовској бази података сваки студент представљен је као чвор типа *Студент* са следећим атрибутима:

- име,
- презиме,

- студијски програм,
- тип завршног рада (мастер или докторска дисертација),
- назив завршног рада,
- година одбране.

Ментор и чланови комисије представљени су као чворови типа *Професор*, са атрибутима:

- име,
- презиме,
- институција у којој је запослен.

Обрада података захтевала је више корака, укључујући:

- Екстракцију године одбране из датума, коришћењем регуларних израза;
- Парсирање и издвајање имена, презимена и институције ментора и чланова комисије;
- Нормализацију података, посебно у случају професора, где су академске титуле биле присутне у именима;

За издвајање и стандардизацију информација о професорима имплементирана је функција која користи регуларне изразе како би прецизно идентификовала име, презиме и институцију професора, узимајући у обзир могућност постојања академских титула и сложених презимена. Додатно, креиране су везе између професора који су истовремено били чланови комисије истом студенту.

Сви подаци из CSV фајла итеративно су обрађени и додати у базу података помоћу Cypher упита.

Процес креирања базе података укључује следеће кораке:

1. Екстракција и форматирање података о студентима и професорима.
2. Креирање чворова за студенте и професоре.
3. Додавање релација између студената и њихових ментора, као и између студената и чланова комисије.

4. Анализа и креирање додатних веза између професора који су заједнички учествовали у комисијама.

5. Чување свих података у бази података.

5.2 Визуализација менторства на Математичком факултету

Библиотека D3.js коришћена је за креирање различитих визуализација:

- **Приказ стабла односа ментора и студената**: Омогућена је визуализација у којој студент заузима коренски чвор стабла, док су професори представљени као листови. Овај приказ омогућава јасан увид у менторства и академске везе између студената и професора.

- **Граф приказа студената једног професора**: Имплементирана је визуализација која приказује све студенте који су имали одређеног професора као ментора или члана комисије.

- **Визуализација студената на основу критеријума**: Омогућен је приказ свих студената који задовољавају одређене критеријуме, омогућавајући динамичну анализу и филтрирање података у реалном времену.

Ове визуализације омогућавају бољу анализу и разумевање односа у академском окружењу, док интеграција са оквиром Django на серверској страни омогућава ефикасно управљање подацима и њихово ажурирање у реалном времену.

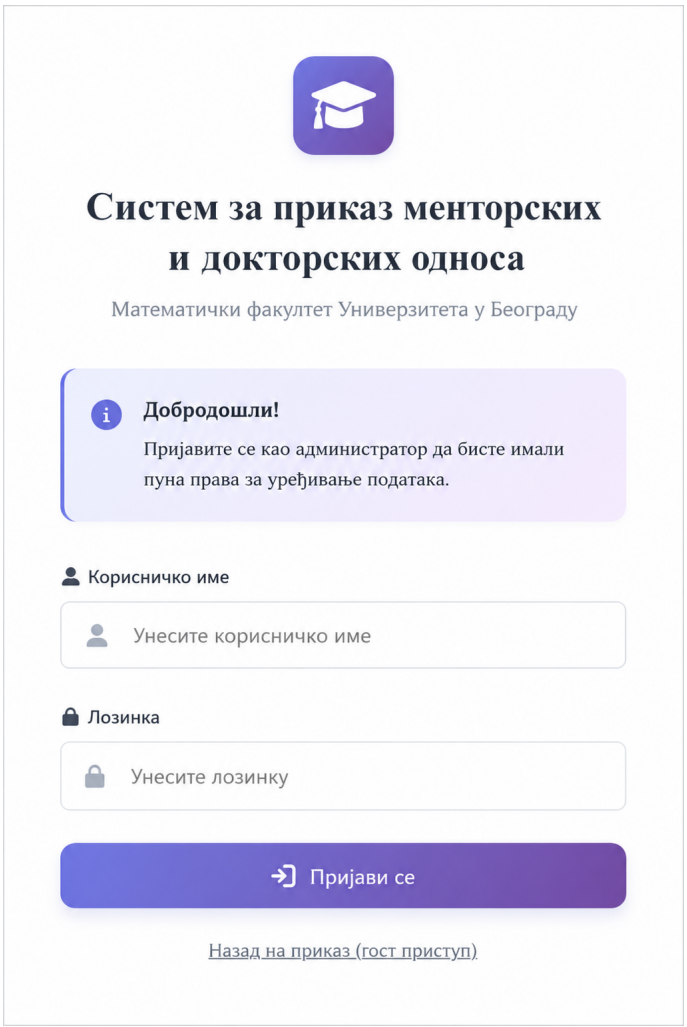
5.3 Могућности апликације

Апликација за праћење и анализу менторства на Математичком факултету пружа бројне могућности за ефикасно претраживање, управљање и визуелизацију података.

Корисничке улоге и приступ

У систему постоје два типа корисника: гост и администратор. По покретању апликације корисник је аутоматски пријављен као гост, што омогућава приступ свим функционалностима које не мењају податке, односно прегледу и претраживању садржаја.

Уколико корисник жели да приступи функционалностима које подразумевају измену података, као што су додавање, ажурирање или брисање, потребно је да се пријави као администратор. Тада се приказује страница за пријаву, приказана на слици 5.1, након чега администратор добија приступ свим административним функционалностима система.

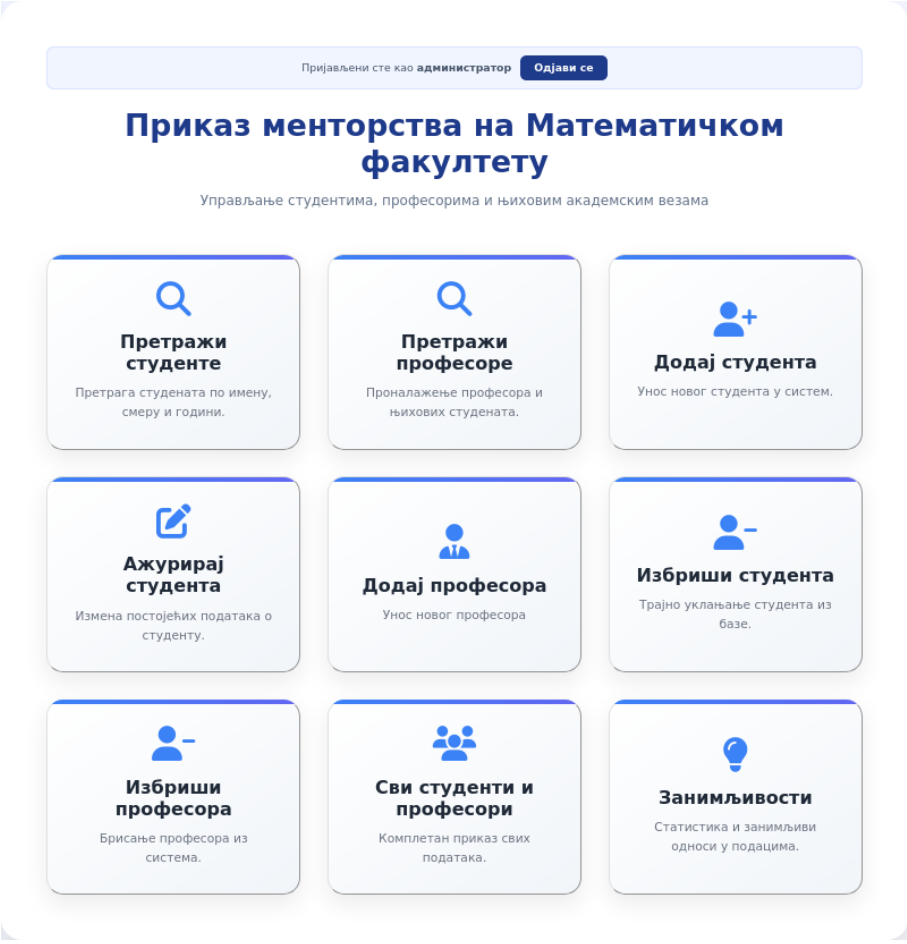


Слика 5.1: Страница за пријаву

Кориснику са налогом госта неће бити приказане опције које омогућавају мењање података у бази. Осетљиве операције које захтевају администраторски приступ су: додавање новог студента, додавање новог професора, ажурирање података о студенту, брисање студента и брисање професора.

Интерфејс апликације

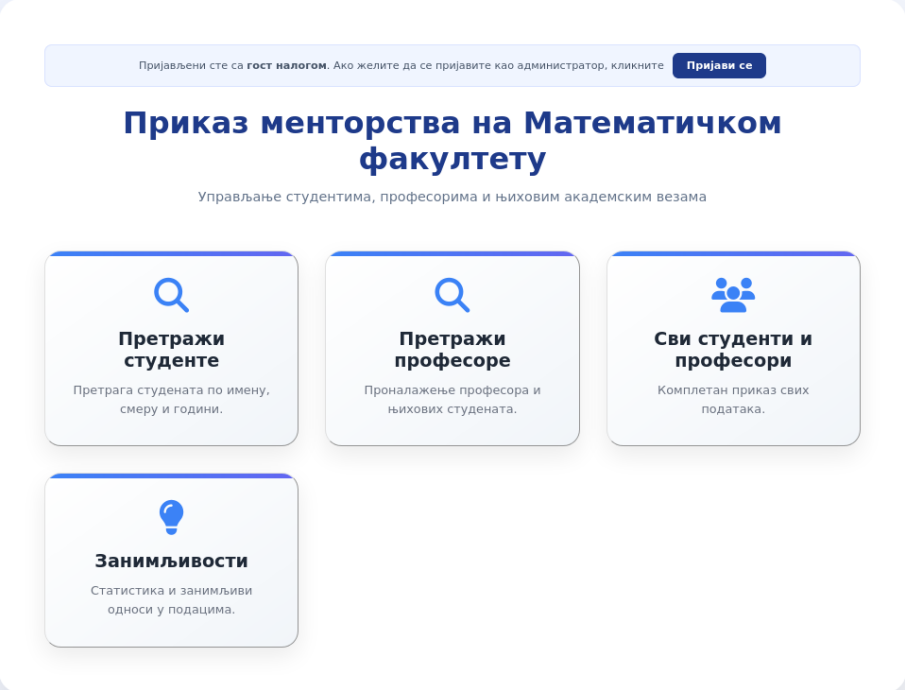
На слици 5.2 приказан је интерфејс апликације када се корисник пријави са администраторским правима. Администратору су доступне све опције, укључујући и оне које омогућавају измену података у бази.



Слика 5.2: Интерфејс апликације за приказ и визуелизацију менторства — администраторски налог

На слици 5.3 приказан је интерфејс апликације када се корисник пријави са налогом госта. Кориснику са налогом госта неће бити приказане опције

које омогућавају мењање података у бази, како би се спречиле неовлашћене измене.



Слика 5.3: Интерфејс апликације за приказ и визуелизацију менторства — налог госта

Функционалности везане за студенте

Апликација омогућава претрагу студената по имену и презимену, као и проналажење менторског графа одређеног студента. Поред основне претраге, корисницима је на располагању и напредна претрага по различитим критеријумима, као што су смер, тип тезе и други релевантни подаци. Резултати претраге могу се приказати на три начина: у облику табеле, графички кроз чворове који приказују повезаност професора, ментора и чланова комисије, или у облику круга у којем су професори представљени у унутрашњем прстену, а студенти у спољашњем, при чему је сваки студент повезан са својим ментором и члановима комисије. Администраторске функционалности обухватају додавање новог студента у базу података, ажурирање његових података, као и брисање студента уколико постоји у систему.

Функционалности везане за професоре

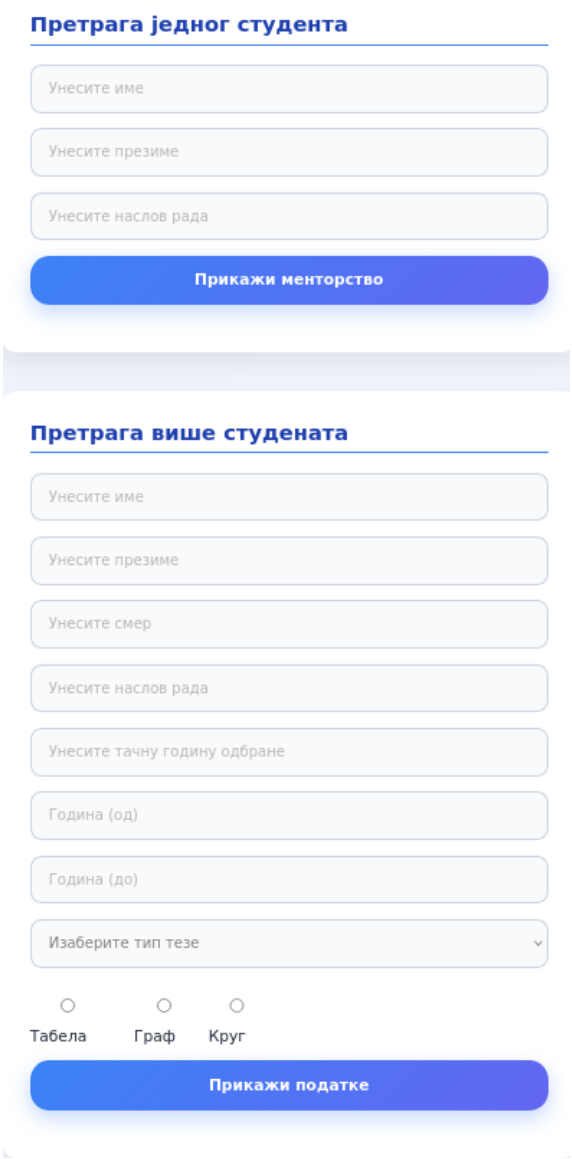
Професоре је могуће претраживати по имену и презимену, као и према институцији у којој су запослени. Након претраге, корисник може изабрати да ли жели да прикаже само студенте којима је одређени професор био ментор, само студенте у чијој комисији је учествовао, или обоје. Апликација подржава и рекурзивни приказ хијерархије ментор-студент — уколико су студенти изабраног професора и сами током каријере постали ментори или чланови комисије другим студентима, систем омогућава приказ и тих веза. За сваког професора доступна је статистика која обухвата број менторстава за мастер радове, број учешћа у комисијама за мастер радове, те број менторстава и учешћа у комисијама за докторске тезе. Администратор може и додавати нове професоре у базу, као и брисати постојеће.

Статистике и прегледи

За потребе анализе, апликација нуди преглед десет професора са највише менторстава на Математичком факултету, десет професора који су најчешће били чланови комисије, као и листу свих професора који су учествовали у менторствима или комисијама, а нису запослени на Математичком факултету. Апликација на тај начин пружа свеобухватан увид у менторство и рад комисија на Математичком факултету, подржавајући ефикасно управљање подацима и омогућавајући детаљне анализе у циљу бољег разумевања и унапређења менторског процеса.

Форма за претрагу студената

На слици 5.4 приказана је форма за претрагу студената, која омогућава корисницима да претражују податке о студентима на основу различитих критеријума.

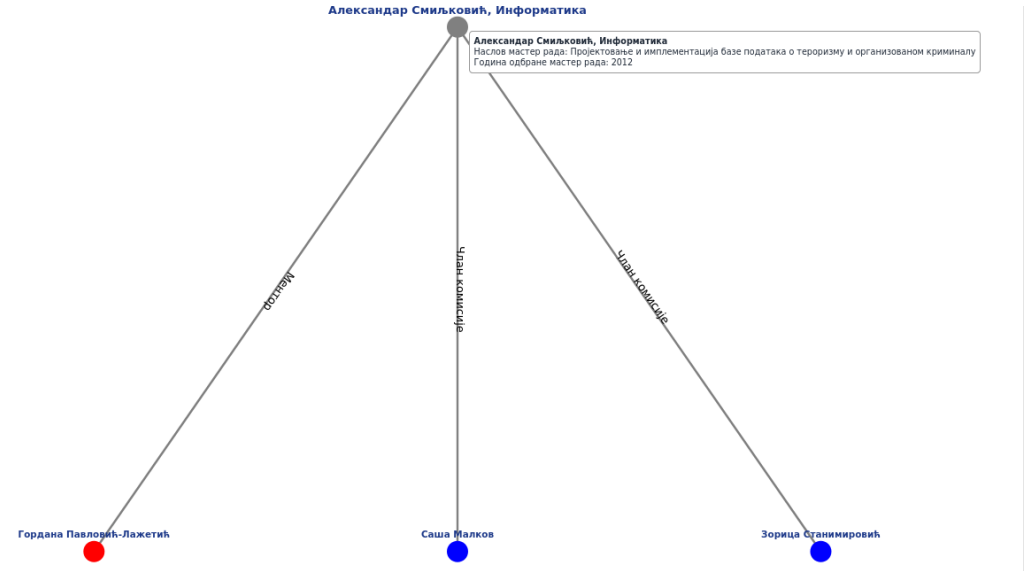


Слика 5.4: Форма за претрагу студената

Претрага једног студента

Корисник може унети име, презиме и наслов рада студента, а затим кликнути на дугме „Прикажи менторство“. У случају да постоји студент у бази података који одговара унетим критеријумима, у корену стабла ће бити приказан чвор који представља тог студента, док ће листови стабла представљати ментора студента и чланове комисије. Уколико неки од података недостаје или ако тражени студент не постоји у бази података, корисник ће бити обавештен одговарајућом поруком о грешци. На слици 5.5 приказан је пример стабла за

студента који одговара критеријумима претраге. Црвеном бојом су приказани ментори, а плавом бојом чланови комисије.



Слика 5.5: Приказ стабла за студента који постоји у бази података

У оквиру серверске стране апликације, имплементирани су упити за дохватање података неопходних за приказ и обраду података. Сваки упит је прилагођен типу тезе студента и омогућава приступ информацијама о студенту, његовим менторима и члановима комисије. Упити приказани у листингу 5.1 омогућавају ефикасан приступ релевантним подацима везаним за студента на основу његовог имена и презимена.

Листинг 5.1: Cypher упити за прикупљање информација о ментору и члановима комисије студента са датим именом и презименом

```
upiti = {
 ,,mentor_master'': (
        ''MATCH (s:Student {ime: $ime, prezime: $prezime, naslov_rada:
    $naslov_rada})
        -[r1:MENTOR]->(mentor:Profesor) ''
        ''WHERE s.tip_teze = 'Master rad' ''
        ''RETURN s, mentor''
    ),
 ,,mentor_doktorska'': (
```

```
9           ''MATCH (s:Student {ime: $ime, prezime: $prezime, naslov_rada:
        $naslov_rada })
10          -[r1:MENTOR]->(mentor:Profesor) ''
11          ''WHERE s.tip_teze = 'Doktorska teza' ''
12          ''RETURN s, mentor''
13      ),
14   ,,komisija_master'': (
15          ''MATCH (s:Student {ime: $ime, prezime: $prezime, naslov_rada:
        $naslov_rada }) ''
16          ''WHERE s.tip_teze = 'Master rad' ''
17          ''MATCH (s)-[r2:CLANOVI_KOMISIJE]->(k:Profesor) ''
18          ''RETURN s, COLLECT(k) AS komisija''
19      ),
20   ,,komisija_doktorska'': (
21          ''MATCH (s:Student {ime: $ime, prezime: $prezime, naslov_rada:
        $naslov_rada }) ''
22          ''WHERE s.tip_teze = 'Doktorska teza' ''
23          ''MATCH (s)-[r2:CLANOVI_KOMISIJE]->(k:Profesor) ''
24          ''RETURN s, COLLECT(k) AS komisija''
25      )
26  }
```

Подаци о студенту се враћају као JSON објекат. Код за визуелизацију података у облику стабла користи библиотеку `D3.js` за генерисање интерактивног графика. Главни делови кода укључују следеће:

1. **Припрема података:** У овој фази процеса, подаци добијени из базе података путем Cypher упита трансформишу се у JSON формат. Информације о студенту, ментору и члановима комисије за мастер рад или докторску тезу организују се у структуриран облик који омогућава ефикасан приступ, једноставну манипулацију у апликацији, као и даљу обраду и адекватан приказ у корисничком интерфејсу.

   У оквиру ове фазе, сваком чвору у графу додељује се одговарајућа улога (нпр. студент, ментор, члан комисије), што омогућава њихово јасно разликовање, као и даљу анализу и повезивање унутар апликације. Поред тога, сваки чвор садржи релевантне атрибуте, као што су име, презиме, институција и друге карактеристике од значаја.

   Подаци су организовани у облику стабла, при чему се у корену налази студент, док листове чине професори који су учествовали као ментори

или чланови комисије на мастер раду и/или докторској тези. Оваква структура омогућава јасан приказ односа и лаку визуализацију повезаности између актера.

У листингу 5.2 приказан је део кода који реализује овакву структуру.

Листинг 5.2: Пример креирања JSON чворова који представљају студента, менторе и чланове комисије за потребе визуализације графовске структуре

```
const podaci = {
''ime'': ''{{ podaci_stablo_student.ime_studenta }}
        {{podaci_stablo_student.prezime_studenta }},
        {{podaci_stablo_student.smer_studenta }}'',
''uloga'': ''student'',
''grupa'':0,
''naslov_master_rada'': ''{{ podaci_stablo_student.naslov_master_rada}}'',
''godina_odbrane_master_rada'': ''{{podaci_stablo_student.
    godina_odbrane_master_rada}}'',
''naslov_doktorske_teze'': ''{{podaci_stablo_student.naslov_doktorske_teze}}'',
            ''godina_odbrane_master_rada'': ''{{ podaci_stablo_student.
    godina_odbrane_master_rada}}'',
            ''godina_odbrane_doktorske_teze'': ''{{ podaci_stablo_student.
    godina_odbrane_doktorske_teze}}'',
            ''tip_teze'': ''{{ podaci_stablo_student.tip_teze }}'',
            ''children'': [
    {% if podaci_stablo_student.mentor_master_rad %}
    {
        ''ime'': ''{{ podaci_stablo_student.mentor_master_rad.ime }} {{
    podaci_stablo_student.mentor_master_rad.prezime }}'',
        ''veza'': ''mentor_master_rad'',
        ''uloga'': ''mentor_master_rad'',
        ''institucija'': ''{{podaci_stablo_student.mentor_master_rad.
    institucija}}'',
        ''grupa'': 1
    }{% if podaci_stablo_student.komisija_master_rad or
    podaci_stablo_student.komisija_doktorska_teza %},{% endif %}
    {% endif %}
    {% for profesor in podaci_stablo_student.komisija_master_rad %}
    {
        ''ime'': ''{{ profesor.ime }} {{ profesor.prezime }}'',
        ''veza'': ''komisija_master_rad'',
        ''uloga'': ''komisija_master_rad'',
```

```
28          ''institucija'': ''{{profesor.institucija}}'',
29          ''grupa'':1
30      }{% if not loop.last or podaci_stablo_student.komisija_doktorska_teza
        %},{% endif %}
31      {% endfor %}
32
```

2. **Генерисање стабла:** Коришћењем функције `d3.hierarchy`, подаци се конвертују у хијерархијску структуру која омогућава лакшу манипулацију и приказ чворова. Затим се функција `d3.tree` из D3.js библиотеке користи за израчунавање и постављање позиција чворова у стаблу, чиме се обезбеђује визуелно организован и читљив приказ структуре.

3. **Визуелизација веза између чворова:** Везе између чворова (студент, ментор, комисија) се приказују линијама које повезују чворове. Свака веза има одговарајући текстуални опис који означава улогу повезану са сваким чвором (нпр. „ментор мастер рада“). Пример кода који омогућава додавање називе везама приказан је у Листинту 5.3.

Листинг 5.3: Код за визуелизацију и етикетирање веза у графу

```
1  svg.selectAll(''.veza'')
2      .data(koren.links())
3      .enter().append(''line'')
4      .attr(''class'', ''link'')
5      .attr(''x1'', d => d.source.x)
6      .attr(''y1'', d => d.source.y)
7      .attr(''x2'', d => d.target.x)
8      .attr(''y2'', d => d.target.y)
9      .attr(''stroke'', ''#999'')
10     .attr(''stroke-width'', 2);
11 svg.selectAll(''.veza-label'')
12     .data(koren.links())
13     .enter().append(''text'')
14     .attr(''class'', ''veza-label'')
15     .attr(''x'', d => (d.source.x + d.target.x) / 2)
16     .attr(''y'', d => (d.source.y + d.target.y) / 2)
17     .style(''text-anchor'', ''middle'')
18     .attr(''transform'', d => {
19           const ugao = Math.atan2(d.target.y - d.source.y, d.target.x - d.
       source.x) * 180 / Math.PI;
```

```
20          return `rotate(${ugao}, ${(d.source.x + d.target.x) / 2}, ${(d.
    source.y + d.target.y) / 2})`;})
21      .text(d => {
22              if (d.target.data.veza === ''mentor_master_rad'') {
23                  return ''Mentor master rada'';
24              }else if(d.target.data.veza === ''mentor_doktorska_teza''){
25                  return ''Mentor doktorske disertacije'';
26              } else if (d.target.data.veza === ''komisija_master_rad'') {
27                  return ''Clan komisije master rad'';
28              }else if (d.target.data.veza === ''komisija_doktorska_teza''){
29                  return ''Clan komisije doktorske disertaciej'';
30              }
31                  return '''';
32              });
```

4. **Интерактивност:** Преласком миша преко чвора који представља студента, корисник може добити детаљне информације о врсти тезе, њеном наслову и години одбране, што такође можемо погледати на слици 5.5. Такође, преласком миша преко чвора који представља ментора или члана комисије, корисник може видети назив институције на којој је професор тренутно запослен. Кликом на чвор који представља ментора или члана комисије, корисник добија приступ статистици о професору, која укључује број радова у којима је био ментор мастер рада, број радова у којима је био члан комисије за мастер рад, као и сличне информације у вези са докторском тезом.

5. **Визуелни аспекти:** Боје чворова се одређују на основу улоге особе у стаблу. На пример, ментори су црвени, а чланови комисије плави.

Претрага више студената

Апликација омогућава и претрагу више студената који задовољавају одређене критеријуме. Критеријуми за претрагу могу укључивати име, презиме, смер, годину или интервал одбране, тип тезе или наслов рада. Ови критеријуми се могу комбиновати, што омогућава да се на крају прикажу само они студенти који испуњавају све задате услове, уколико такви постоје.

Поред тога, апликација нуди различите начине приказа резултата претраге. Студенти који задовољавају критеријуме могу бити приказани у та-

беларном формату, као повезани граф између студената и професора, или у форми динамичког „круга", где су професори приказани у спољашњем кругу, а студенти у унутрашњем. Сваки студент у овом кругу је повезан са својим ментором и члановима комисије, чиме се визуализује однос између студента и професора на веома интуитиван начин. Без обзира на начин приказа који је изабран, у свим случајевима користи се иста функција у делу за логичку обраду захтева да би се издвојили релевантни подаци. Једина разлика јесте у томе што када се подаци приказују у табеларном облику, тада се не прикупљају подаци о менторима и члановима комисије. У додатку 7.1 је дат део функције која се користи за дохватање релевантних информација. У наставку су описани различити облици приказа података.

- **Табеларни приказ**
  Подаци добијени у логичком делу апликације прослеђују се шаблону, где их приказујемо у табеларном облику помоћу једноставног HTML кода. Поред основних информација о студентима, имплементирана је и могућност приказа њиховог менторског стабла кликом на дугме „Прикажи граф". На слици 5.6 приказан је део табеле са студентима који испуњавају критеријум да су на смеру Математика, да је година одбране у распону од 2019. до 2021. године и да је тип тезе Докторска теза.

РЕДНИ БРОЈ	ИМЕ	ПРЕЗИМЕ	НАСЛОВ	СМЕР	ТИП ТЕЗЕ	ГОДИНА ОДБРАНЕ	ПРИКАЖИ ГРАФ
1	Марија	Цупарић	Тестови сагласности засновани на L² и L[∞] растојањима и њихова асимптотска ефикасност	Математика	Докторска теза	2021	Прикажи граф
2	Милан	Лазаревић	Неједнакости Коши-Шварца и Грис-Ландауа за елементарне операторе и трансформере типа унутрашњег производа на Q и Q[*] идеалима компактних оператора	Математика	Докторска теза	2020	Прикажи граф
3	Khola	Ajlidi	Средње вредности мултипликативних аритметичких функција више променљивих зависних од НЗД и НЗС аргумената	Математика	Докторска теза	2020	Прикажи граф
4	Јелена	Ивановић	Кохеренција и прости политопи	Математика	Докторска теза	2020	Прикажи граф
5	Марија	Јелић Милутиновић	Комбинаторна топологија и графовски комплекси	Математика	Докторска теза	2021	Прикажи граф
6	Александар	Јовић	Услови екстремума за једну класу проблема оптимизације са непрекидним временом	Математика	Докторска теза	2021	Прикажи граф
7	Маја	Рославцев	Гребнерове базе за коначно генерисане идеале над неким класама нетериних прстена	Математика	Докторска теза	2021	Прикажи граф
8	Милан	Перић	Полиномијална ентропија за Морсове градијентне системе и логистичко пресликавање	Математика	Докторска теза	2021	Прикажи граф
9	Маринко	Тимотијевић	Ауто-дуални симплицијални комплекси, њихова генерализација и примене у комбинаторици и геометрији	Математика	Докторска теза	2019	Прикажи граф
10	Марко	Пешовић	Комбинаторика уопштених пермутоедара	Математика	Докторска теза	2021	Прикажи граф

Слика 5.6: Табеларни приказ студената који задовољавају услове

- **Приказ података у облику графа**
  У овом сегменту апликације генерише се структура података која се

користи за визуелизацију односа између студената, ментора и чланова комисије. Најважнији део кода односи се на итерацију кроз податке и њихову конверзију у формат прилагођен за приказ помоћу D3.js библиотеке. Прво се прави празан објекат *podaci*, који садржи два кључна елемента:

1. чворове, представљају студенте, менторе и чланове комисије, при чему сваки чвор има јединствени ID, име и додатне атрибуте (смер, институција итд.)

2. везе, представљају односе између студената и њихових ментора, као и између студената и чланова комисије.

```
const podaci = {
''cvorovi'': [],
''veze'': []
};
```

Приликом итерације кроз податке осигурава се да се исти чвор не додаје више пута у граф, користећи структуру скуп. Сваки студент се повезује са својим ментором и члановима комисије, при чему је поступак додавања студента у граф и провере јединствености чвора илустрован у листингу 5.4.

Листинг 5.4: Додавање чвора студента у граф и провере постојања чвора помоћу скупа

```
studentId = ''{{ student.ime }} {{ student.prezime }}'';
if (!postojeciCvorovi.has(studentId)) {
    podaci.cvorovi.push({
        ''id'': studentId,
        ''ime'': ''{{student.ime}} {{student.prezime}}'',
        ''uloga'': ''student'',
        ''smer'': ''{{student.smer}}'',
        ''naslov_rada'': ''{{student.naslov}}'',
        ''godina_odbrane'': ''{{student.godina_odbrane}}'',
        ''tip_teze'': ''{{ student.tip_teze }}''
    });
    postojeciCvorovi.add(studentId);
}
```

Ментор и чланови комисије се такође додају у граф, уколико већ нису додати, односно ако не постоје у структури података, пример кода можемо погледати у листингу 5.5.

Листинг 5.5: Пример додавања чвора ментора у граф уз проверу да ли већ постоји у структури података

```
mentorId = ''{{ student.mentor.ime }} {{ student.mentor.prezime }}'';
if (!postojeciCvorovi.has(mentorId)) {
    podaci.cvorovi.push({
        ''id'': mentorId,
        ''uloga'': ''mentor'',
        ''ime'': ''{{ student.mentor.ime }} {{ student.mentor.prezime }}'',
        ''institucija'': ''{{student.mentor.institucija}}''
    });
    postojeciCvorovi.add(mentorId);
}
```

На крају се додају везе између студената и њихових ментора, као и чланова комисије, како је приказано у листингу 5.6.

Листинг 5.6: Додавање веза између студента и његовог ментора у структуру података

```
podaci.veze.push({
    ''source'': mentorId,
    ''target'': studentId,
});
```

Имплементација обухвата следеће кључне елементе:

1. Дефинисање платна

Платно на којем се исцртава граф дефинише се заједно са легендом која објашњава боје чворова (ментори – црвено, чланови комисије – плаво, студенти – сиво), што је приказано у листингу 5.7.

Листинг 5.7: Дефинисање платна и додавање легенде која објашњава боје чворова у графу

```
var legenda = svg.append(''g'')
    .attr(''class'', ''legend'')
    .attr(''transform'', ''translate(100, 100)'');
```

```
var podaciLegende = [
    { label: ''Profesor mentor'', color: ''red'' },
    { label: ''Profesor clan komisije'', color: ''blue'' },
    { label: ''Student'', color: ''gray'' }
];
```

2. Дефинисање физичких сила

Библиотека D3 користи симулацију физичких сила како би осигурала природно распоређивање чворова у простору и спречила њихово преклапање. Функција *d3.forceSimulation* креира динамичку симулацију која управља кретањем чворова на основу дефинисаних сила. Током симулације, положаји чворова се континуирано ажурирају, што омогућава да граф изгледа природно и прегледно. Снага симулације лежи у комбинацији различитих сила: *d3.forceLink* моделује везе између чворова и одржава оптималну удаљеност између њих, *d3.forceManyBody* примењује одбијајуће или привлачне силе како би се спречило груписање чворова на једном месту, док *d3.forceCenter* поставља централну тачку око које се организује цела структура. На овај начин, визуелизација графа добија јасан и уредан распоред, који је лако пратити и анализирати. Пример кода можемо погледати у листингу 5.8.

Листинг 5.8: Дефинисање силе и симулације

```

var simulacija = d3.forceSimulation(podaci.cvorovi)
    .force(''link'', d3.forceLink(podaci.veze)
    .id(d => d.id)
     .distance(50))
    .force(''charge'', d3.forceManyBody().strength(-60))
    .force(''center'', d3.forceCenter(sirina / 2, visina / 2))
    .force(''collision'', d3.forceCollide().radius(68));
```

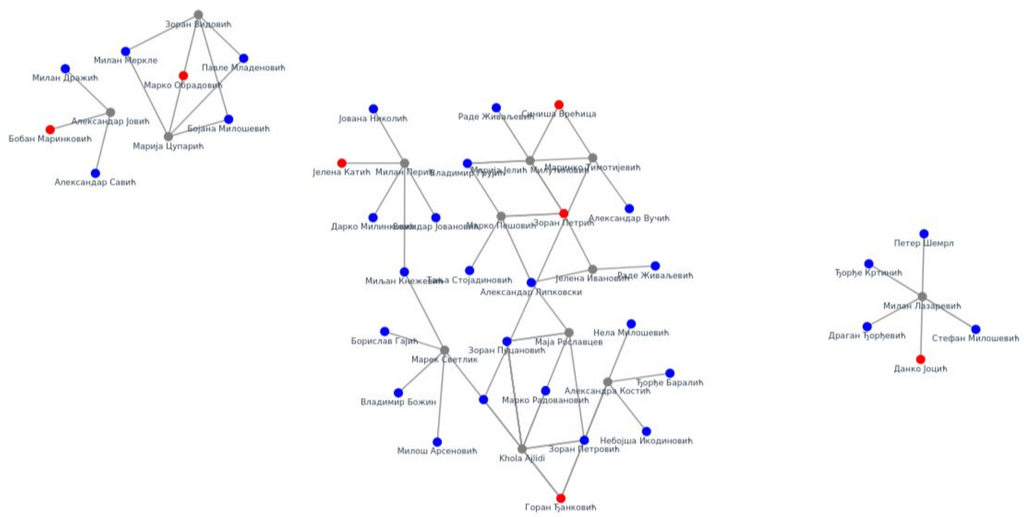
3. Додавање чворова и веза

Подаци о студентима, менторима и члановима комисије користе се за генерисање чворова (круг) и веза (линија) које повезују ентитете.

4. Интерактивност

Имплементирана је функционалност приказа додатних информација о ентитетима када корисник мишем пређе преко чвора. Такође, омогуће-

но је повлачење чворова како би корисник могао да реорганизује приказ према потреби. На слици 5.7 приказан је граф са студентима који испуњавају критеријум да су на смеру Математика и да је година одбране у распону од 2019. до 2021. године и чији је тип тезе Докторска теза.



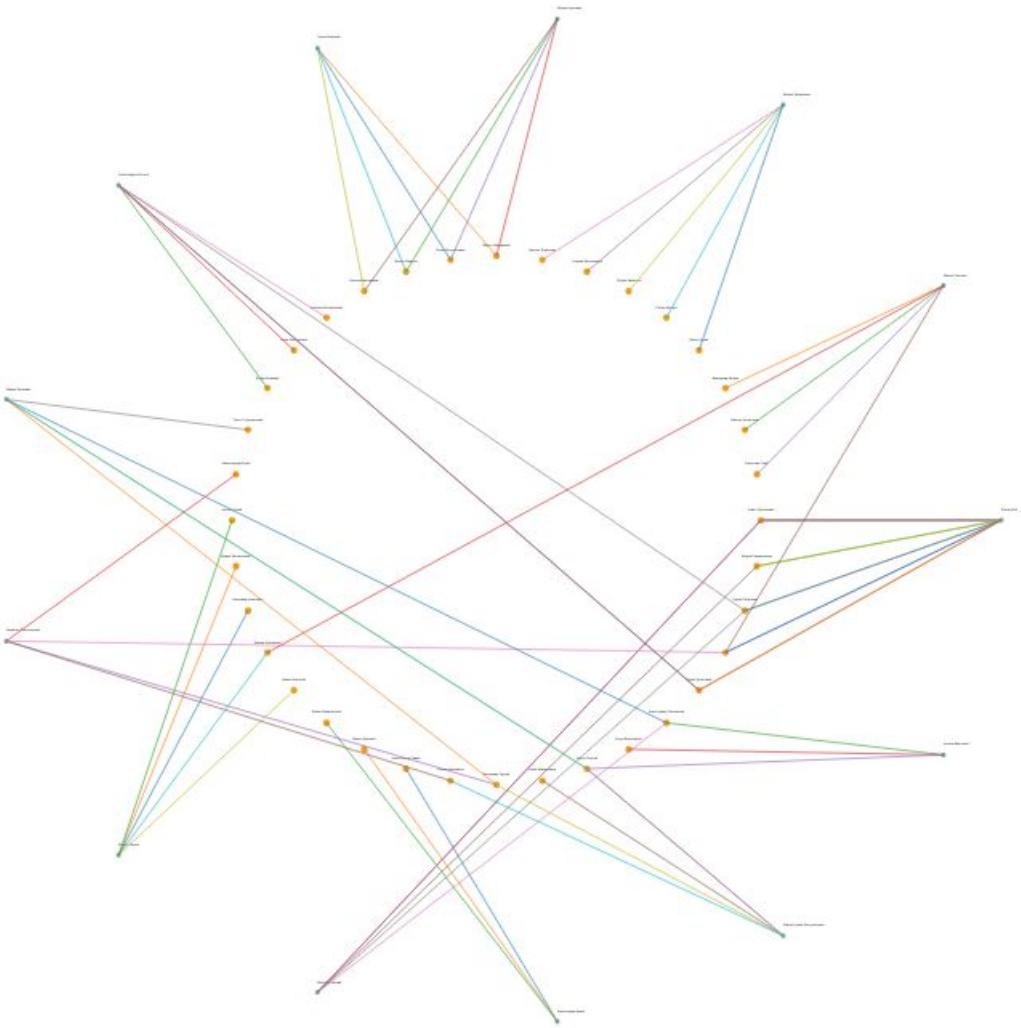
Слика 5.7: Графовски приказ студената који задовољавају услове

- **Приказ података у облику круга**

  Реализована је функционалност која омогућава избор приказа података у облику два круга, при чему су у спољашњем кругу студенти, а у унутрашњем чланови комисије и ментори. Сви студенти су повезани са својим менторима и члановима комисије.

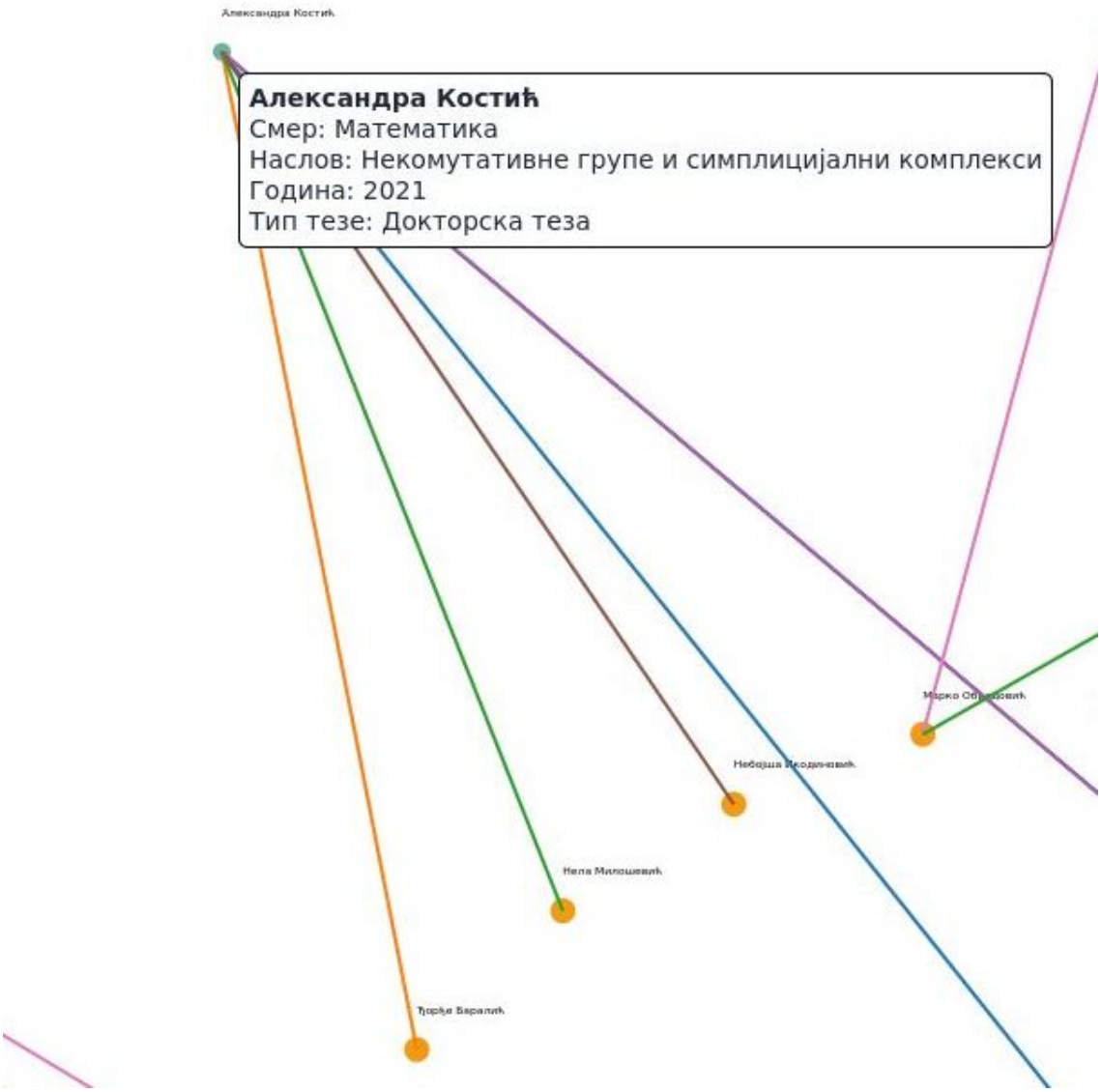
  Студенти се постављају у кружну формацију око центра графика. Прво се израчунава угао између сваког чвора у зависности од броја чворова који испуњавају услове за приказивање, а затим се користе тригонометријске функције како би се равномерно позиционирали. Коришћењем *Math.cos* и *Math.sin*, свака тачка се поставља на одговарајућу локацију на кружници. На сличан начин, професорски чворови се распоређују у унутрашњост круга, при чему се успостављају везе између студентских чворова и професорских чворова који представљају њихове менторе и чланове комисије. Циљ ове визуализације је да омогући прегледан и ин-

туитиван приказ односа између студената и њихових ментора/чланова комисије у формату који лако може да прикаже везе и повезаност између академских чланова. У додатку 7.2 можемо погледати код који се односи на распоређивање чворова. На слици 5.8 налазе се студенти који испуњавају критеријум да су на смеру Математика, да је година одбране у распону од 2019. до 2021. године и да је тип тезе Докторска теза.



Слика 5.8: Приказ студената који задовољавају услове у облику круга.

На слици 5.9 приказан је издвојени и увећани део графа, на коме су ознаке чворова јасно видљиве.



Слика 5.9: Увећани приказ издвојеног дела графа студената који задовољавају задате услове.

Претрага професора

Претрага професора представља једну од кључних функционалности система и омогућава кориснику да лако приступи подацима о менторским и комисијским улогама појединих професора. Претрага се врши уношењем имена и презимена професора. Корисницима је омогућена претрага студената једног професора, приказ статистике о изабраном професору, као и приказ свих

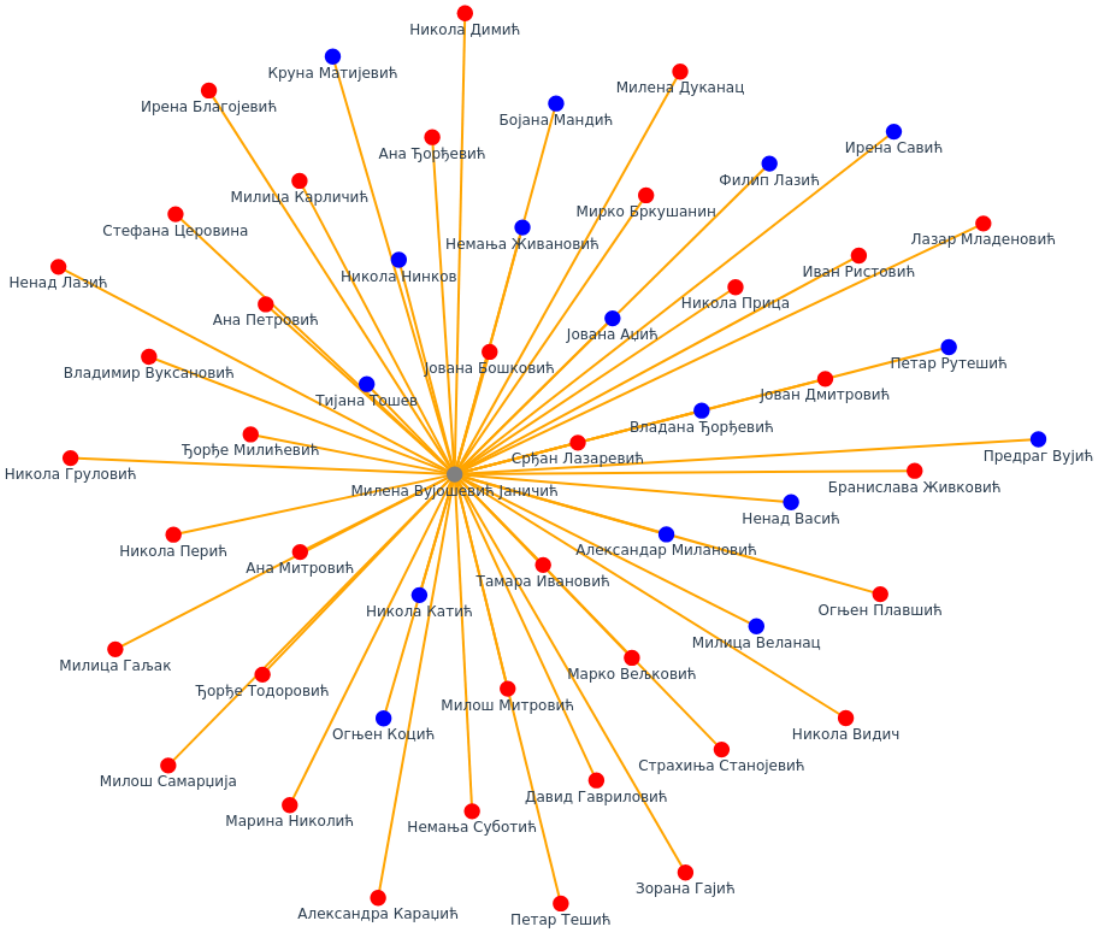
професора са дате институције. Ове функционалности пружају корисницима интуитиван и прегледан начин за анализу академских веза, и може бити посебно корисна приликом планирања менторстава, састављања комисија или истраживања академске историје факултета.

Претрага студената једног професора

Уколико унети подаци одговарају неком од постојећих професора у бази, кориснику се нуди избор између три различите опције приказа података. Прва опција омогућава приказ свих студената којима је дати професор био ментор. Друга опција приказује списак студената у чијим је комисијама професор учествовао као члан. Трећа, најобухватнија опција, пружа могућност да се прикажу сви студенти повезани са изабраним професором – како они којима је био ментор или члан комисије, тако и студенти његових студената, уколико су и они у међувремену стекли звање професора. На тај начин добија се проширено менторско стабло које илуструје везе између генерација студената и професора.

Сви подаци се визуелно представљају у облику графа. Чворови у графу представљају студенте и професоре, док везе између њих указују на улогу коју је одређени професор имао у процесу менторства или чланства у комисији. Преласком мишем преко било ког чвора који представља студента, појављују се детаљне информације о изабраном студенту – укључујући назив и тему рада, смер студија, годину одбране, као и друге релевантне податке. Додатно, кликом на сам чвор приказује се комплетно менторско стабло тог студента, омогућавајући дубље истраживање односа унутар академске заједнице.

На слици можемо видети 5.10 можемо видети студенте којима је професорка Милена Вујошевић Јаничић била ментор, односно члан комисије.

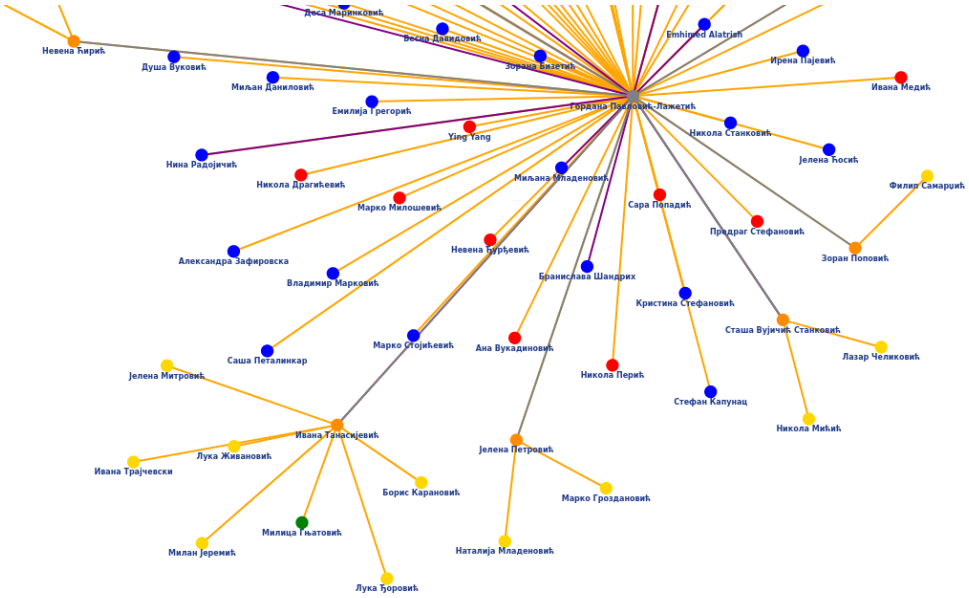


Слика 5.10: Студенти којима је професорка Милена Вујошевић Јаничић била ментор (црвеном бојим) и члан комисије (плавом бојом)

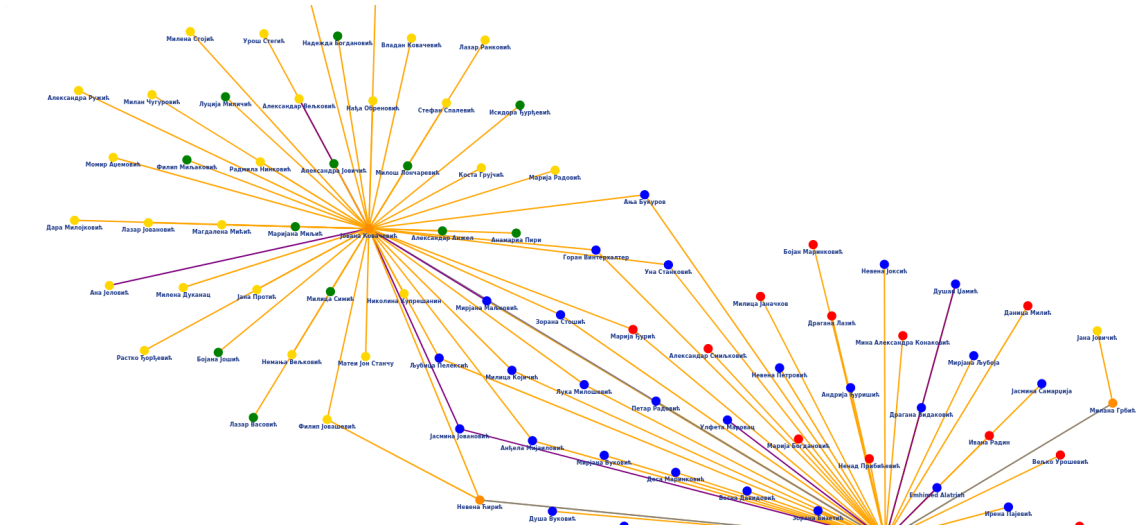
На сликама 5.11 и 5.12 приказани су студенти професорке Гордане Павловић Лажетић. Изабрана је опција рекурзивног приказа, што значи да су, поред студената професорке Гордане, приказани и студенти њених студената који су у међувремену постали професори на Математичком факултету. У графу су чворови студената обојени плавом бојом уколико је професорка била члан њихове комисије, односно црвеном бојом уколико им је била ментор. Наранџаста боја означава студенте који су у међувремену постали професори и имају своје студенте.

У рекурзивном приказу, зелена боја означава менторство, док жута боја означава чланство у комисији. Боја везе између чворова додатно означава тип

тезе — наранџаста боја представља мастер радове, а љубичаста докторске тезе.



Слика 5.11: Приказ студената којима је професорка Гордана Павловић-Лажетић била ментор или члан комисије, као и менторских односа њених бивших студената који су касније постали професори и њихових студената.



Слика 5.12: Приказ студената којима је професорка Гордана Павловић-Лажетић била ментор или члан комисије, као и менторских односа њених бивших студената који су касније постали професори и њихових студената.

Важни делови имплементације укључују иницијализацију објекта графа који садржи два низа: чворове и везе. Пример овакве иницијализације и додавања студената са одговарајућим везама може се видети у листингу 5.9. У зависности од тога која је од три поменуте опције изабрана, издвајају се подаци о потребним студентима, на пример само оних којима је изабрани професор био ментор, и додају се у структуру заједно са везама између њих.

Листинг 5.9: Иницијализација објекта графа и додавање студената и веза у зависности од изабране опције

```
if (ime && prezime) {
    const podaci = {
        ''cvorovi'': [
            {
                ''id'': ime + '' '' + prezime,
                ''grupa'': 0,
                ''uloga'': ''profesor''
            }
        ],
        ''veze'': []
    };
    let studentiSet = new Set();
    if (mentorstvo) {
        {% for student in podaci_stablo_profesor.mentorstvo_master_rad %}
            var studentId = ''{{ student.ime }} {{ student.prezime }}'';
            if (!studentiSet.has(studentId)) {
                studentiSet.add(studentId);
                podaci.cvorovi.push({
                    ''id'': studentId,
                    ''grupa'': 1,
                    ''uloga'': ''master_student'',
                    ''naslov_master_rad'': ''{{ student.naslov }}'',
                    ''smer'': ''{{ student.smer }}'',
                    ''godina_odbrane_master_rad'': ''{{ student.godina_odbrane }}''
                });
            }
            podaci.veze.push({
                ''source'': ime + '' '' + prezime,
                ''target'': studentId,
                ''tip'': ''master''
            });
        {% endfor %}
```

Затим се иницијализује SVG елемент како би се омогућио приказ потенцијално великог броја чворова и веза између њих. Симулација користи *forceSimulation* функцију из D3.js библиотеке, која аутоматски распоређује позиције чворова применом физичких сила — привлачења, одбијања и сударања. Ово омогућава читљив распоред без преклапања чворова, што је илустровано у листингу 5.10.

Листинг 5.10: Покретање симулације са подешавањем сила за распоред чворова

```
var simulacija = d3.forceSimulation(podaci.cvorovi)
    .force(''link'', d3.forceLink(podaci.veze).id(d => d.id).distance(20))
    .force(''charge'', d3.forceManyBody().strength(-800))
    .force(''center'', d3.forceCenter(sirina/2 , visina/2 ))
    .force(''collision'', d3.forceCollide().radius(80));
```

Додавањем *click* и *mouseover* догађаја омогућена је интеракција са графом. Кликом на чворове који представљају студенте омогућава се приказ менторског стабла изабраног студента, док кликом на професорски чвор добијамо информације о статистици професора. Преласком миша корисник добија додатне информације о студенту (смер, наслов рада, година одбране). Пример имплементације дат је у листингу 5.11.

Листинг 5.11: Додавање *click* и *mouseover* догађаја ради омогућавања интеракције са графом

```
.on(''mouseover'', function (event, d) {
    ispisPodataka.style(''visibility'', ''visible'')
    .html(`Ime: ${d.ime}<br>Prezime: ${d.prezime}<br>...`)
    .style(''left'', (event.pageX + 10) + ''px'')
    .style(''top'', (event.pageY + 10) + ''px'');
})
.on(''click'', function(event, d) {
    if (d.grupa === 0) {
        prikaziGraf(d.ime, d.prezime);
    } else {
        prikaziStatistiku(d.ime, d.prezime);
    }
});
```

Чворови и везе у графу стилизовани су различитим бојама, у зависности од њиховог значења. Везе се приказују у наранџастој, љубичастој или сивој боји, зависно од тога да ли представљају мастер, докторску или рекурзивну везу. Чворови се боје у складу са својом групом: ментори су црвени, чланови комисије плави, а студенти сиви. Део кода који реализује ову логику налази се у Листингу 5.12.

Листинг 5.12: Стилизовање чворова и веза према њиховом типу и значењу

```
link.attr(''stroke'', function(d) {
    if (d.tip === ''master'') return ''orange'';
    if (d.tip === ''doktorska'') return ''purple'';
    return ''gray'';
});
cvor.append(''circle'')
    .attr(''fill'', function(d) {
        if(d.grupa == 1) return ''red'';
        if(d.grupa == 2) return ''blue'';
        return ''gray'';
    })
```

Да би се омогућила додатна флексибилност, имплементирана је функција за превлачење чворова. На тај начин корисник може ручно прилагодити положај чворова ради боље прегледности. Ова класификација олакшава кориснику разумевање структуре графа и улоге сваког чвора у оквиру ње.

Приказ статистике о професору

Апликација такође пружа могућност приказа проширених информација о професору, укључујући податке о броју мастер радова на којима је био ментор, учешћу у комисијама за одбрану мастер радова, као и броју докторских теза које је менторисао и комисијама за њихову одбрану у којима је учествовао.

Приказ професора са изабране институције

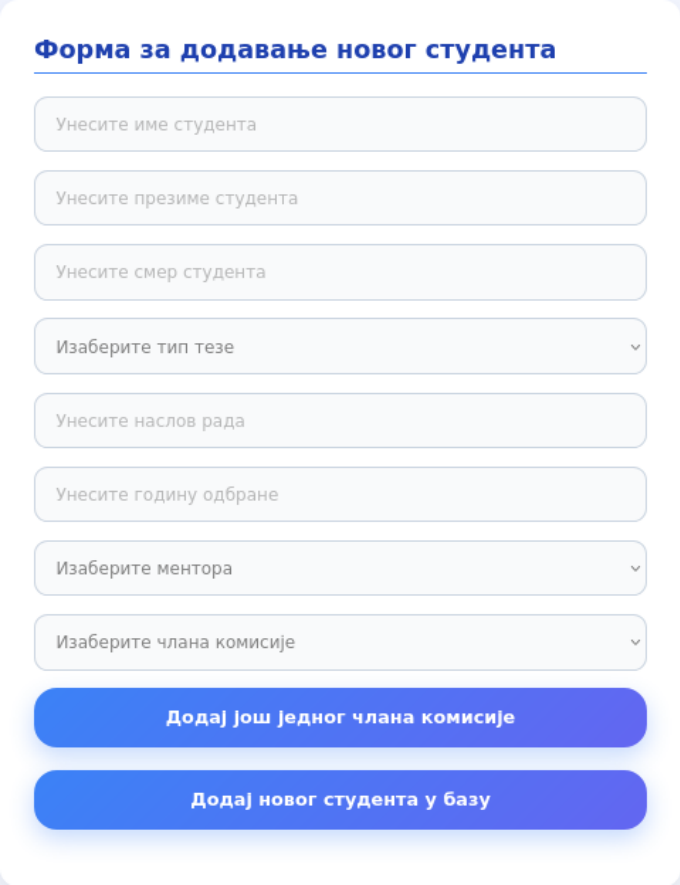
Апликација такође омогућава претрагу професора према институцији у којој су запослени. На слици 5.13 можемо видети све професоре са Матаматичког института САНУ.

ИМЕ	ПРЕЗИМЕ	ИНСТИТУЦИЈА
Марко	Марковић	МИ САНУ
Петар	Лабан	МИ САНУ
Слободан	Симић	научни саветник МИ САНУ
Ненад	Младеновић	МИ САНУ
Бранко	Драговић	МИ САНУ
Борислав	Гајић	МИ САНУ
Марко	Стошић	МИ САНУ
Татјана	Давидовић	МИ САНУ
Драгош	Цветковић	МИ САНУ
Ђорђе	Баралић	МИ САНУ

Слика 5.13: Професори са институције Математички институт САНУ

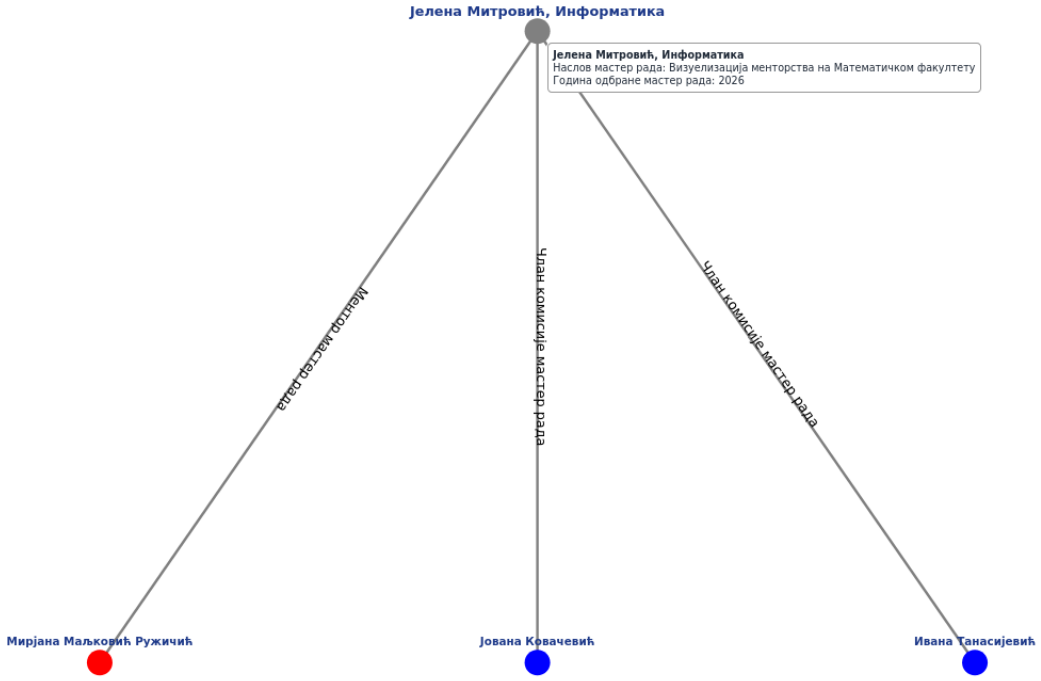
Додавање нових студената

Апликација омогућава додавање студената у базу података уз одређена ограничења. Приликом додавања студента, корисник бира ментора и чланове комисије из листе већ постојећих професора. На овај начин је онемогућено да студент буде повезан са професорима који не постоје у бази података. Због тога је потребно да сви професори буду унети у систем пре него што се приступи додавању студената и повезивању са менторима и члановима комисије. На слици 5.14 приказана је форма за унос новог студента.



Слика 5.14: Форма за унос новог студента

Уколико исправно унесемо податке о студенту, његовом ментору и члану комисије, нови студент ће исправно бити додат у базу података, што можемо видети на слици 5.15.



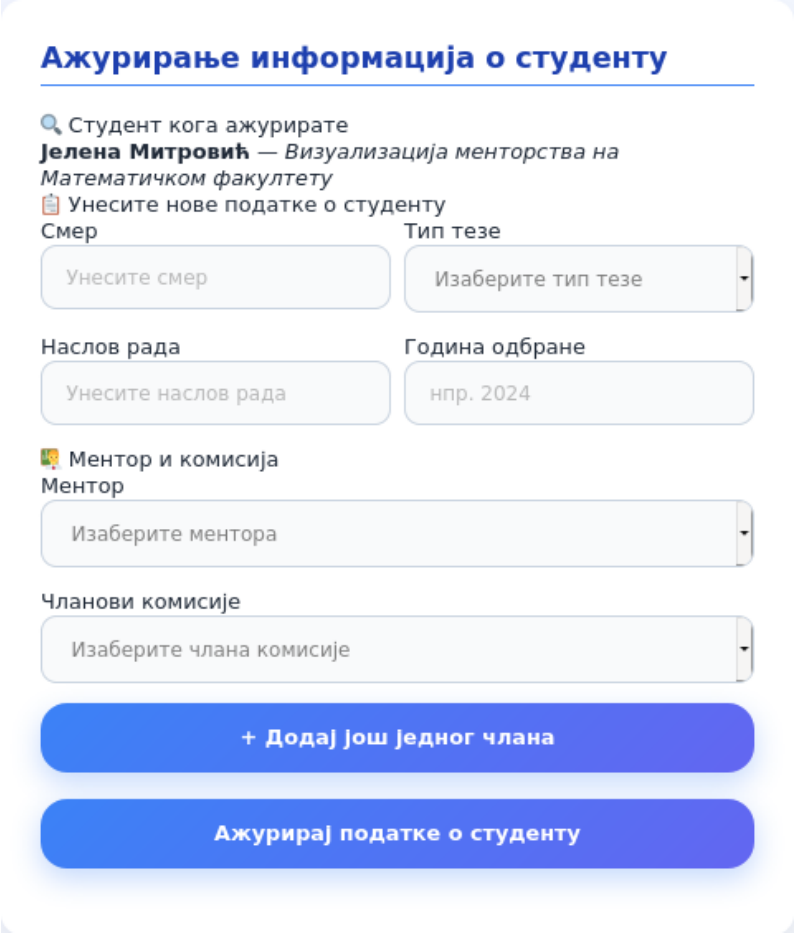
Слика 5.15: Претрага новог студента након што смо га успешно додали у базу података.

Приликом имплементације логичке обраде захтева, подаци унети путем форме преузимају се у оквиру функције типа views. На основу тих података формира се одговарајућа Cypher наредба која има за циљ да у базу унесе новог студента. Уколико су сви подаци валидни, прави се чвор за новог студента на основу унетих података. Након тога, успостављају се везе између студента и његовог ментора, као и са члановима комисије. На тај начин се обезбеђује да се у базу уносе само валидне и повезане информације.

Ажурирање информација о студенту

Поступак ажурирања информација о студенту започиње избором опције за ажурирање, након чега се приказује форма за претрагу студента. Корисник уноси име и презиме студента чије податке жели да измени. На основу унетих података, систем приказује све студенте који одговарају задатим критеријумима. Након избора жељеног студента кликом на дугме „Ажурирај“ отвара се форма за ажурирање његових података.

На слици 5.16 приказан је изглед форме за ажурирање информација о студенту. Поља која садрже име, презиме и наслов рада изабраног студента аутоматски су попуњена подацима из базе. Поред тога, форма садржи поља у која корисник уноси нове вредности које жели да ажурира за изабраног студента.

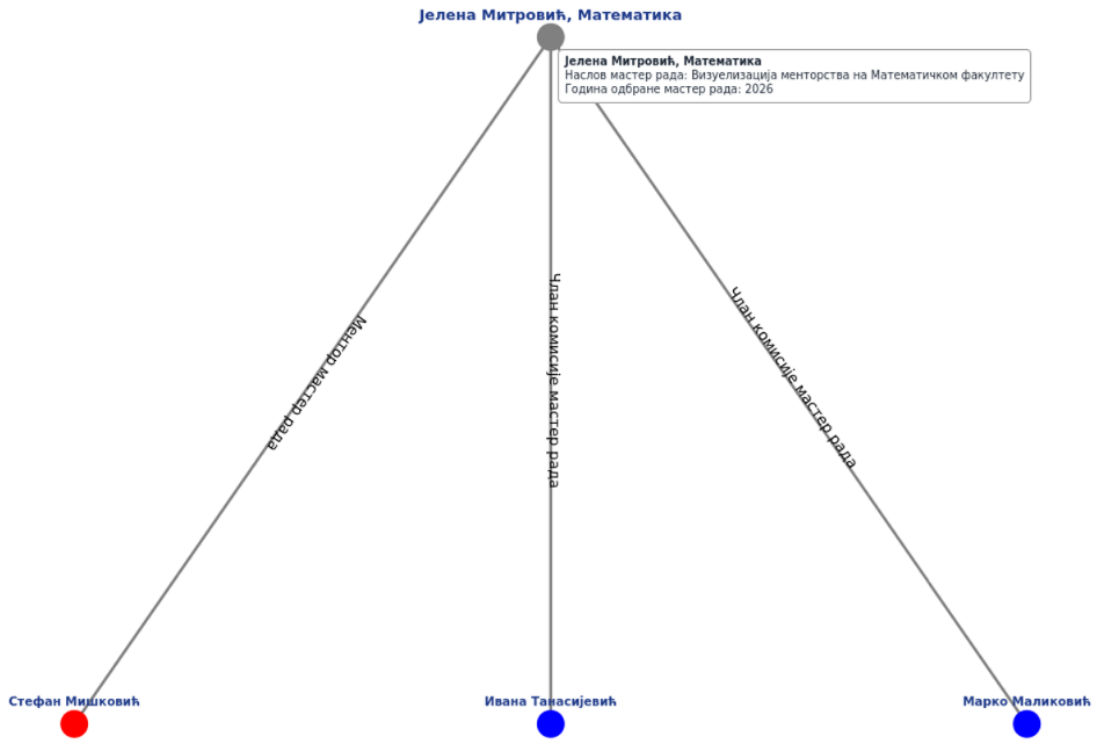


Слика 5.16: Форма за ажурирање информација о студенту

Ажурирање информација о студентима у апликацији могуће је само уколико студент већ постоји у бази података. Након потврде измена, систем ажурира постојеће податке о студенту. Уколико су измењени ментор или чланови комисије, постојеће релације се уклањају, а затим се формирају нове релације које одговарају ажурираним подацима.

На пример, желимо да ажурирамо ментора, смер и члана комисије студенту који већ постоји у бази података. Менторско стабло студента пре ажурирања приказано је на слици 5.15. Након успешно извршеног ажурирања,

приказ информација о студенту и његово менторско стабло одражавају нове податке, што је приказано на слици 5.17.

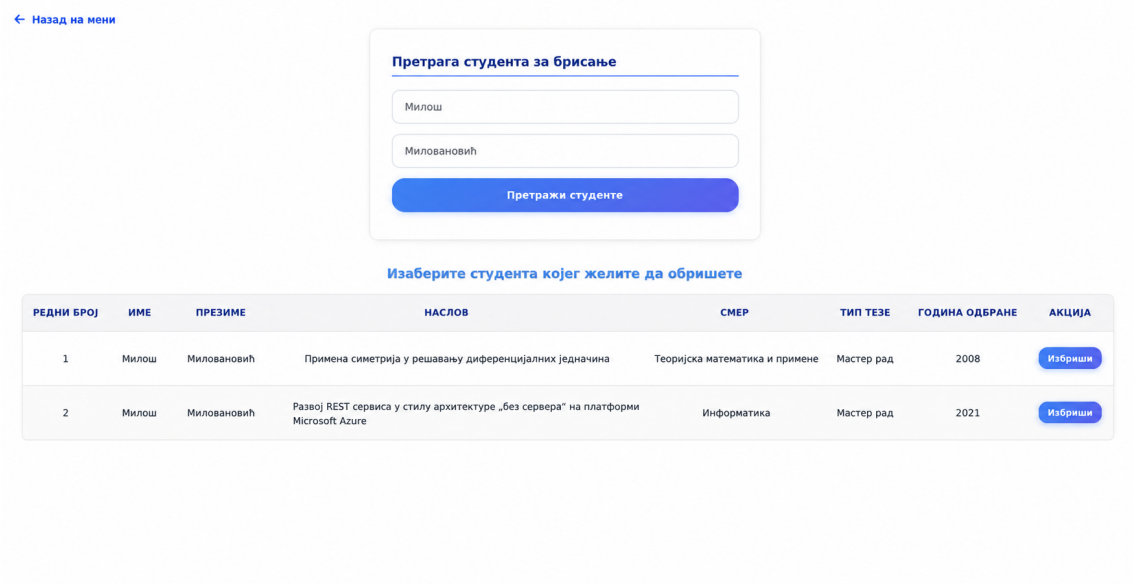


Слика 5.17: Приказ информација о студенту након успешног ажурирања података

Избриши студента из базе

Форма за брисање студента служи за уклањање студента из базе података. Приликом брисања потребно је унети име и презиме студента, након чега систем претражује базу података и приказује све студенте који одговарају унетим подацима. Корисник затим бира студента којег жели да обрише, а изабрани студент ће бити успешно уклоњен кликом на дугме „Избриши“. На слици 5.18 приказан је пример претраге студента који постоји у бази података.

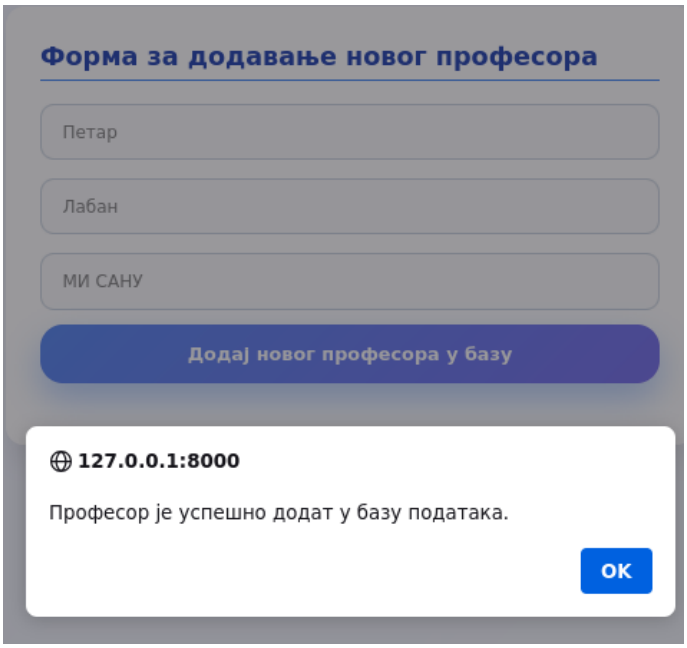
Уколико у бази података не постоји ниједан студент са унетим именом и презименом, брисање неће бити могуће, а корисник ће бити обавештен одговарајућом поруком.



Слика 5.18: Приказ студената пронађених на основу унетог имена и презимена

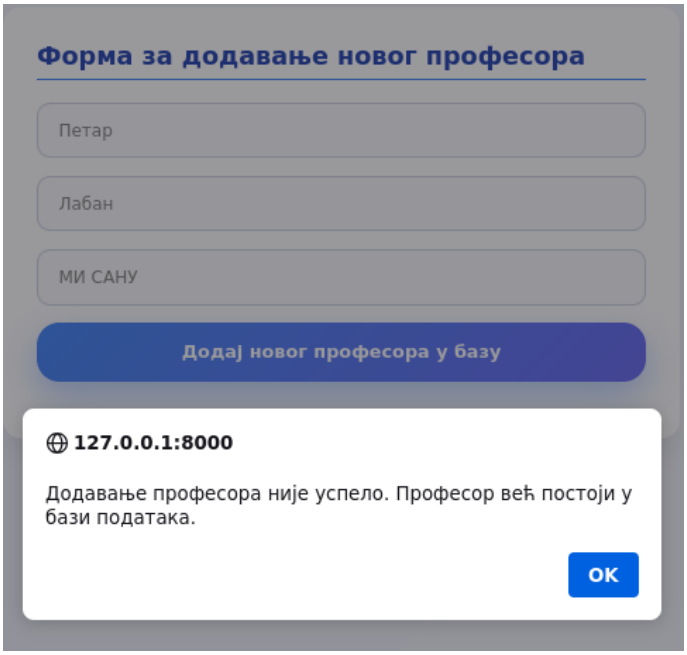
Додавање новог професора

Форма за унос новог професора омогућава додавање професора у базу података уношењем имена, презимена и институције у којој је запослен. На слици 5.19 приказан је пример успешног додавања професора који до тада није постојао у бази података.



Слика 5.19: Успешно додавање новог професора у базу података

Није могуће унети професора који већ постоји у бази података са истим именом, презименом и институцијом, што можемо видети на слици 5.20, где је приказан покушај поновног додавања истог професора.

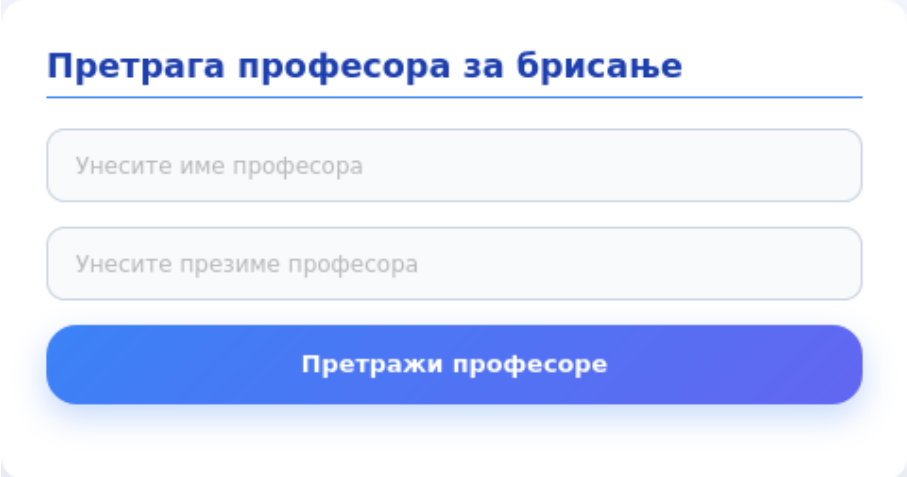


Слика 5.20: Неуспешан покушај додавања већ постојећег професора

Избриши професора из базе

Брисање професора из базе података врши се путем форме у коју је потребно унети име и презиме професора. Након претраге, систем приказује све професоре са унетим именом и презименом у облику табеле. Корисник затим бира професора којег жели да обрише, а изабрани професор ће бити успешно уклоњен из базе података кликом на дугме „Избриши“.

На слици 5.21 може се видети изглед саме форме за брисање професора.



Слика 5.21: Форма за брисање професора

Прикажи све студенте и професоре

Кликом на дугме „Прикажи све студенте и све професоре“ добија се преглед свих студената и професора. Студенти су распоређени у облику спољашњег круга, при чему су приказани само њихови иницијали. Када се курсором миша пређе преко неког студента, појављују се додатне информације као што су име, презиме, смер, тип тезе, година одбране и наслов рада. Уколико се кликне на чвор који представља студента, биће приказано стабло менторства за тог студента. У унутрашњем делу круга налазе се чворови који представљају професоре. Професори су међусобно повезани линијама уколико су заједно били чланови комисије или ментори истом студенту. Кликом на чвор који представља професора, приказује се статистика везана за тог професора.

Прикажи занимљивости

Кликом на дугме „Прикажи занимљивости" биће приказане три интересантне табеле. Прва табела приказује професоре са Математичког факултета који су најчешће били ментори, друга приказује професоре који су највише пута били чланови комисије, док трећа приказује професоре који су били ментори или чланови комисије студентима Математичког факултета, а запослени су у некој другој институцији. На сликама 5.22, 5.23 и 5.24 , можемо видети поменуте табеле.

ИМЕ	ПРЕЗИМЕ	ИНСТИТУЦИЈА	БРОЈ МЕНТОРСТАВА
Мирослав	Марић	МАТФ: Катедра за рачунарство и информатику	100
Миодраг	Живковић	МАТФ	73
Александар	Липковски	МАТФ: Катедра за алгебру и математичку логику	60
Милан	Божић	МАТФ	57
Владимир	Филиповић	МАТФ: Катедра за рачунарство и информатику	53
Бојана	Милошевић	МАТФ: Катедра за вероватноћу и статистику	43
Саша	Малков	МАТФ: Катедра за рачунарство и информатику	42
Ненад	Митић	МАТФ: Катедра за рачунарство и информатику	39
Небојша	Икодиновић	МАТФ: Катедра за алгебру и математичку логику	38
Милена	Вујошевић Јаничић	МАТФ: Катедра за рачунарство и информатику	36

Слика 5.22: Професори са Математичког факултета који су највише пута били ментори

ИМЕ	ПРЕЗИМЕ	ИНСТИТУЦИЈА	БРОЈ ЧЛАНСТАВА У КОМИСИЈИ
Филип	Марић	МАТФ: Катедра за рачунарство и информатику	139
Саша	Малков	МАТФ: Катедра за рачунарство и информатику	139
Александар	Липковски	МАТФ: Катедра за алгебру и математичку логику	94
Владимир	Филиповић	МАТФ: Катедра за рачунарство и информатику	90
Марко	Обрадовић	МАТФ: Катедра за вероватноћу и статистику	88
Милан	Божић	МАТФ	84
Зоран	Петровић	МАТФ: Катедра за алгебру и математичку логику	82
Миљан	Кнежевић	МАТФ: Катедра за реалну и комплексну анализу	71
Небојша	Икодиновић	МАТФ: Катедра за алгебру и математичку логику	65
Мирослав	Марић	МАТФ: Катедра за рачунарство и информатику	59

Слика 5.23: Професори са Математичког факултета који су највише пута били чланови комисије

ИМЕ	ПРЕЗИМЕ	ИНСТИТУЦИЈА
Александар	Перовић	Саобраћајни факултет,Универзитет у Београду
Раде	Живаљевић	МИ САНУ
Јозеф	Кратица	МИ САНУ
Слободан	Симић	научни саветник МИ САНУ
Христос	Краваритис	Технички универзитет Минхен

Слика 5.24: Професори који су били ментори и чланови комисија студентима Математичког факултета, а који нису професори Математичког факултета.

Упити којима се подаци о професорима који су најчешће били чланови комисије извлаче из базе приказани су у додатку 7.3.

Глава 6

Закључак

У оквиру овог рада развијена је веб апликација која омогућава визуелизацију менторских односа на Математичком факултету, са циљем да се омогући јаснији и прегледнији увид у повезаност између студената, ментора и чланова комисија. Апликација пружа интуитиван приказ менторских стабала и омогућава корисницима лаку претрагу и преглед релевантних података. Кроз употребу графовских база података и савремених веб технологија, приказани су начини на које се могу организовати и визуелизовати сложени односи у академском окружењу. Иако су основне функционалности успешно имплементиране, постоји простор за даљи развој, као што је унапређење корисничког интерфејса, додавање напреднијих опција претраге и детаљнија анализа менторских веза. Посебан правац развоја може обухватити дубинску анализу односа између студената и професора, што би омогућило боље разумевање образаца менторства и академског напретка.

У будућности, апликација би се могла проширити и на друге институције, чиме би се омогућило шире поређење академских структура и мрежа. Додатно, увођење статистичких и визуелних приказа допринело би јаснијем сагледавању менторских токова и повезаности унутар академске заједнице.

Такође, значајно проширење система подразумевало би укључивање података о студентима који су студирали пре увођења Болоњског процеса, чиме би се омогућила свеобухватнија анализа кроз различите периоде образовног система и добио потпунији увид у развој академских односа током времена.

Глава 7

Додатак

Листинг 7.1: Функција која проналази све студенте који задовољавају унете критеријуме

```
def nadji_studente(tx, godina_odbrane=None, smer=None, naslov = None,godina_od=
    None, godina_do=None,tip_teze = None,ime_studenta = None,prezime_studenta =
    None,mentor_i_clanovi_komisije=False):
    upit = ''MATCH (s:Student) WHERE ''
    uslovi = []
    if godina_odbrane is not None:
        uslovi.append(''s.godina_odbrane = $godina_odbrane'')
    if smer:
        uslovi.append(''s.smer = $smer'')
    if naslov:
        uslovi.append(''(s.naslov STARTS WITH $naslov OR s.naslov = $naslov)'')
    if godina_od is not None:
        uslovi.append(''s.godina_odbrane >= $godina_od'')
    if godina_do is not None:
        uslovi.append(''s.godina_odbrane <= $godina_do'')
    if tip_teze is not None:
        uslovi.append(''s.tip_teze = $tip_teze'')
    if ime_studenta is not None:
        uslovi.append(''s.ime = $ime_studenta'');
    if prezime_studenta is not None:
        uslovi.append(''s.prezime = $prezime_studenta'');
    if uslovi:
        upit += '' AND ''.join(uslovi)
    else:
        upit = upit.rstrip('' WHERE'')
```

```
    if mentor_i_clanovi_komisije:
        upit += """
        MATCH (s)-[r2:CLANOVI_KOMISIJE]->(clan:Profesor)
        MATCH (s)-[r:MENTOR]->(mentor:Profesor)
        RETURN
            s AS student,
            s.ime AS ime,
            s.prezime AS prezime,
            s.naslov AS naslov,
            s.smer AS smer,
            s.godina_odbrane AS godina_odbrane,
            s.tip_teze AS tip_teze,
            COLLECT(clan {ime: clan.ime, prezime: clan.prezime, institucija: clan.institucija}) AS komisija,
            mentor {ime: mentor.ime, prezime: mentor.prezime, institucija: mentor.institucija} AS mentor
        """
    else:
        upit += """
        RETURN s.ime AS ime, s.prezime AS prezime, s.naslov AS naslov,
               s.smer AS smer, s.godina_odbrane AS godina_odbrane, s.tip_teze AS tip_teze
        """
```

Листинг 7.2: Распоређивање студентских чворова у кругу помоћу D3.js библиотеке

```
const ugaoStudentskiCvorovi = (2 * Math.PI) / studenti.length;
const studentskiCvorovi = svg.selectAll(''.student-node'')
    .data(studenti)
    .enter()
    .append(''g'')
    .attr(''class'', ''student-node'')
    .attr(''transform'', (d, i) => {
        const ugao = i * ugaoStudentskiCvorovi;
        const x = Math.cos(ugao) * spoljasnjiUgao;
        const y = Math.sin(ugao) * spoljasnjiUgao;
        return `translate(${x}, ${y})`;
    });
```

Листинг 7.3: Функција која прикупља податке о 10 професора са највише менторстава

```
def top_10_mentorstva(tx):
    upit = (
        ''MATCH (p:Profesor)<-[r:MENTOR]-(s:Student) ''
        ''WITH p, COUNT(s) AS broj_mentorstava ''
        ''ORDER BY broj_mentorstava DESC ''
        ''LIMIT 10 ''
        ''RETURN p.ime, p.prezime, p.institucija, broj_mentorstava''
    )
    rezultati = tx.run(upit)
    return [rezultat for rezultat in rezultati]
```

Глава 8

Употреба алата заснованих на вештачкој интелигенцији

Током израде овог мастер рада коришћени су алати засновани на вештачкој интелигенцији, пре свега Microsoft Copilot и ChatGPT, као помоћни алати у процесу развоја програмске имплементације и писања рада.

Microsoft Copilot је коришћен првенствено током развоја програмске имплементације, за генерисање и унапређивање појединих делова програмског кода, помоћ при отклањању грешака, као и за разматрање одговарајућих имплементационих решења. ChatGPT је коришћен као помоћ при формулисању, преформулисању и језичком уједначавању појединих делова текста.

Садржај и предлози добијени коришћењем ових алата нису преузимани без додатне провере. Исправност генерисаног и измењеног програмског кода проверена је тестирањем имплементираних функционалности и поређењем добијених резултата са очекиваним резултатима. Теоријске тврдње и објашњења проверени су на основу релевантне стручне литературе наведене у библиографији, док су сви садржаји укључени у рад додатно проверени, кориговани и прилагођени теми и циљевима мастер рада.

Литература

[1] Neo4j. *Cypher Query Language*. Доступно на: https://neo4j.com/docs/cypher-manual/current/ (приступ јул 2025).

[2] Neo4j. *Official website*. Доступно на: https://neo4j.com/ (приступ јул 2025).

[3] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč. *Foundations of Modern Query Languages for Graph Databases*. ACM Comput. Surv. 50, 5, 2018. Доступно на: https://doi.org/10.1145/3104031 (приступ јул 2025).

[4] Зоран Станић. *Дискретне структуре 2*. Математички факултет, Београд, 2020.

[5] Ian Robinson, Jim Webber, and Emil Eifrem. *Graph Databases*. 2nd edition, O'Reilly Media, 2015. Доступно на: https://neo4j.com/books/graph-databases/ (приступ јул 2025).

[6] ArangoDB GmbH. *ArangoDB Documentation*. Доступно на: https://www.arangodb.com/docs/stable/ (приступ јул 2025).

[7] Amazon Web Services. *Amazon Neptune – Fast, reliable graph database service*. Доступно на: https://aws.amazon.com/neptune/ (приступ јул 2025).

[8] Microsoft Azure. Azure Cosmos DB documentation. Доступно на: https://learn.microsoft.com/en-us/azure/cosmos-db/ (приступ јул 2025).

[9] Neo4j. Comparing relational to graph database. Доступно на: https://neo4j.com/developer/graph-db-vs-rdbms/ (приступ јул 2025).

[10] Django Software Foundation. *Django Documentation*. Доступно на: https://docs.djangoproject.com/en/stable/ (приступ јул 2025).

[11] Mike Bostock. *D3.js Documentation*. Доступно на: `https://d3js.org/` (приступ јул 2025).

[12] MDN Web Docs. *Document Object Model (DOM)*. Доступно на: `https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model` (приступ јул 2025).

[13] MDN Web Docs. *Scalable Vector Graphics (SVG)*. Доступно на: `https://developer.mozilla.org/en-US/docs/Web/SVG` (приступ јул 2025).

[14] MDN Web Docs. *Canvas API*. Доступно на: `https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API` (приступ јул 2025).

[15] World Wide Web Consortium (W3C). *HTML 5.2 Specification*. Доступно на: `https://www.w3.org/TR/html52/` (приступ јул 2025).

[16] Scott Murray. *Interactive Data Visualization for the Web*. O'Reilly Media, 2013.

[17] Neo4j. *Life Sciences Use Cases*. Доступно на: `https://neo4j.com/use-cases/life-sciences/` (приступ јул 2025).

[18] Neo4j. *Fraud Detection Use Cases*. Доступно на: `https://neo4j.com/use-cases/fraud-detection/` (приступ јул 2025).

[19] Neo4j. *Financial Services Use Cases*. Доступно на: `https://neo4j.com/use-cases/financial-services/` (приступ јул 2025).

[20] Neo4j. *Real-Time Recommendation Engine Use Case*. Доступно на: `https://neo4j.com/use-cases/real-time-recommendation-engine/` (приступ јул 2025).

[21] Neo4j. *Social Network Use Case*. Доступно на: `https://neo4j.com/use-cases/social-network/` (приступ јул 2025).

[22] Neo4j. *Telecommunications Use Cases*. Доступно на: `https://neo4j.com/use-cases/telecommunications/` (приступ јул 2025).

[23] Neo4j. *Building an Educational Platform with Neo4j*. Доступно на: `https://neo4j.com/developer-blog/building-educational-platform-neo4j/` (приступ јул 2025).

[24] ECMA International. *ECMA-404 The JSON Data Interchange Standard*. Доступно на: `https://www.ecma-international.org/publications-and-standards/standards/ecma-404/` (приступ јул 2025).

Биографија аутора

Јелена Митровић рођена је 8. фебруара 2001. године у Лесковцу. Основну школу и гимназију завршила је у Косовској Каменици. Године 2019. уписала је Основне академске студије на Математичком факултету Универзитета у Београду, на студијском програму Информатика. Дипломирала је у јуну 2024. године са просечном оценом 8,65.

Исте године уписала је Мастер академске студије на Математичком факултету Универзитета у Београду, на студијском програму Информатика.