Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nermin Jukan

Razvoj spletne platforme za vizualizacijo in deljenje medicinskih algoritmov

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Miha Moškon

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Razvoj spletne platforme za vizualizacijo in deljenje medicinskih algoritmov

Tematika naloge:

Kandidat naj v svojem delu pregleda obstoječe stanje spletnih podatkovnih baz in z njimi povezanimi platformami za deljenje diagnostičnih in ostalih medicinskih algoritmov. Predlaga naj svojo rešitev, ki bi omogočala gradnjo vizualizacij medicinskih algoritmov in deljenje teh vizualizacij s klinično in znanstvenoraziskovalno skupnostjo. Rešitev naj svoje funkcionalnosti končnim uporabnikom nudi preko spletnega brskalnika.

Zahvaljujem se doc. dr. Mihi Moškonu za mentorstvo in pomoč pri izvedbi diplomske naloge, članom projektne skupine DiaGenKri za strokovno pomoč pri analizi obstoječega stanja in formiranju potreb znanstvenoraziskovalne stroke ter Davidu Zagoršku za ideje, nasvete in aktivno spremljanje razvoja platforme.

Zahvaljujem se tudi svoji družini za potrpežljivost in vse, kar so mi v življenju dali.

Posebna zahvala gre moji punci Petri za vso motivacijo, podporo, ljubezen in vse radosti življenja.

Mojim prijateljem in družini,
tistim, ki so blizu, in
tistim, ki so daleč, daleč stran.

Kazalo

Povzetek

Abstract

1 Uvod **1**

2 Medicinski algoritmi **3**

2.1 Uporaba medicinskih algoritmov 3

2.2 Pregled obstoječih platform za deljenje medicinskih algoritmov 4

2.2.1 Platforma American Academy of Family Physicians (AAFP) 4

2.2.2 Platforma The Medical Algorithm Project (MEDAL) . 5

2.3 Dobre prakse in problemi obstoječih platform 7

3 Opis rešitve **9**

3.1 Opis uporabljenih tehnologij 10

3.1.1 Spletni strežnik Apache 10

3.1.2 PHP 10

3.1.3 Podatkovna baza MySQL 11

3.1.4 MVC 11

3.1.5 HTML 5 11

3.1.6 CSS 12

Bootstrap 12

3.1.7 JavaScript 13

jQuery 13
Raphaël.js 13
Notify.js 14
3.1.8 AJAX 14
3.1.9 Format JSON 14
3.1.10 Format SVG 15
3.1.11 Format PNG 15
3.1.12 Git 15
3.2 Opis strukture platforme 15
3.2.1 Model platforme 15
3.2.2 Podatkovna baza 16
3.2.3 Neregistriran uporabnik 17
3.2.4 Registriran uporabnik 18
Uporabniške pravice 19
Orodje za gradnjo vizualizacij 19
Filter za iskanje algoritmov in pregledovalnik vizualizacij 20
Uporabniški profil 21
Urejanje uporabniških pravic 21
Potrjevanje algoritmov 22
3.3 Implementacija platforme 22
3.3.1 Postavitev razvojnega okolja 22
3.3.2 Implementacija modela MVC 23
3.3.3 Postavitev podatkovne baze 23
3.3.4 Izdelava sistema za registracijo in prijavo 23
3.3.5 Izdelava uporabniškega profila 25
3.3.6 Izdelava orodja za gradnjo vizualizacij 25
3.3.7 Prikaz, filtriranje in urejanje algoritmov 30
3.3.8 Uporabniške pravice in administracija uporabnikov . . 32
3.3.9 Potrjevanje algoritmov 33

4 Primer uporabe **35**
4.1 Izdelava novega algoritma 35

4.1.1 Registracija in prijava 35
4.1.2 Gradnja algoritma 36
4.2 Prikaz algoritma 37

5 Zaključek **39**

Literatura **41**

Seznam uporabljenih kratic

kratica	angleško	slovensko
MVC	Model–view–controller	model-pogled-krmilnik
PHP	Hypertext Preprocessor	skriptni programski jezik
SQL	Structured Query Language	strukturirani povpraševalni jezik
HTML	Hyper Text Markup Language	jezik za označevanje nadbesedila
CSS	Cascading Style Sheets	kaskadne stilske predloge
AJAX	Asynchronous JavaScript and XML	asinhroni JavaScript in XML
HTTP	Hypertext Transfer Protocol	protokol za prenos hiperteksta
API	application programming interface	aplikacijski programski vmesnik
XML	Extensible Markup Language	razširljiv označevalni jezik
JSON	JavaScript Object Notation	objektna notacija JavaScript
SVG	Scalable Vector Graphics	umerljiva vektorska grafika
PNG	Portable Network Graphics	prenosljiva spletna grafika
PDO	PHP Data Object	podatkovni objekt PHP

Povzetek

Naslov: Razvoj spletne platforme za vizualizacijo in deljenje medicinskih algoritmov

Avtor: Nermin Jukan

Vizualizacija postaja vse bolj pomemben aspekt moderne znanosti. Medicina, tako kot ostale veje znanosti, zelo hitro napreduje in količina informacij, ki jih je potrebno obdelati, da ohranimo stabilno in produktivno delovno okolje, narašča iz dneva v dan.

Vizualizacija medicinskih algoritmov je zelo uporabna metoda za pospeševanje odločitvenega procesa diagnosticiranja in zdravljenja bolnikov. Zdravnikom in ostalim končnim uporabnikom omogoča boljše razumevanje zapletenih procesov, ki jih algoritmi prikazujejo. Cilj te diplomske naloge je bil izdelati vizualizacijsko orodje, ki uporabnikom omogoča preprost način izgradnje vizualizacije medicinskih algoritmov, ter spletno platformo za deljenje teh vizualizacij znotraj znanstvenoraziskovalne stroke in znotraj širše javnosti.

Platforma je razvita po modelu MVC, kjer zaledni del temelji na jeziku PHP in uporablja podatkovno bazo MySQL. Začelni del izvršujejo tehnologije kot so HTML5, CSS, JavaScript in AJAX. Naloga je nadaljevanje dela, nastalega v okviru projekta Genetska diagnostika krvnih bolezni (DiaGen-KRI), ki je bil sofinanciran v okviru razpisa Študentski inovativni projekt za družbeno korist (ŠIPK), in je nadgradnja predhodno preizkušenih funkcionalnosti. Predstavlja prvi poizkus formiranja spletne skupnosti za ustvarjanje in deljenje vizualizacij medicinskih algoritmov.

Ključne besede: vizualizacija, medicinski algoritmi, spletne tehnologije, zdravstvo in medicina.

Abstract

Title: Development of web platform for visualization and sharing of medical algorithms

Author: Nermin Jukan

Visualization has become a very important aspect in modern science. Medicine, like all other branches of science, is rapidly progressing and the amount of information that needs to be analysed in order to maintain a stable and productive environment is increasing day by day.

Visualization of medical algorithms can be very useful for speeding up the decision making process of diagnosing and treating patients. It also helps doctors and other end-users to better understand the complex processes that the algorithms represent. The goal of this thesis was to develop a visualization tool that would allow for a simple way of building visualizations of medical algorithms and to develop a web-based platform that would support the sharing of these visualizations amongst scientists, researchers and the general public.

The platform is designed upon a MVC model, where the back end was created using the PHP language and the MySQL database. The front end is executed by technologies such as HTML5, CSS, JavaScript and AJAX. This thesis is a continuation of the work done on the project Genetic diagnosis of blood disorders, funded by Študentski inovativni projekt za družbeno korist (ŠIPK), and is an upgrade of the already tested functionalities. It presents a first atempt of forming a web comunity for creating and sharing visualisations of medical algorithms.

Keywords: visualization, medical algorithms, web technologies, health and medical sciences.

Poglavje 1

Uvod

V sodobnem svetu znanosti je vse težje pridobiti pomembne informacije iz različnih podatkovnih virov. Vizualizacija ima pri tem ključno vlogo, saj nam omogoča, da podatkom dodamo kontekts. S pomočjo dodanega konteksta omogočimo bolj poglobljeno razumevanje podatkov in tvorjenje informacij, hkrati pa pri tem ne popačimo kakovosti podatkov in ne izgubimo prvotne podatkovne vrednosti. Preko boljšega razumevanja pridobljenih podatkov lahko sprejemamo hitrejše ter bolj učinkovite odločitve.

Zdravniki, laboratorijski raziskovalci in znanstveniki se srečujejo s problematiko posredovanja podatkov in deljenja informacij tako z drugimi strokovnjaki kot tudi s pacienti. V širši uporabi je namreč več možnih načinov zapisovanja in prikazovanja podatkov, ki so opisani v [1] in [2]. Zaenkrat sistemska rešitev, ki bi omogočala tako deljenje informacij kot posodabljanje podatkov, hkrati pa bi bila uporabljena tako s strani zdravnikov, znanstvenikov in raziskovalcev kot tudi s strani širše javnosti, še ne obstaja.

Velik delež strokovnih vsebin je še vedno dostopen le preko člankov objavljenih v znanstvenih revijah. Znanstveni članki so v najboljšem primeru razpršeni po različnih spletnih straneh in nimajo poenotenega načina predstavitve podatkov. Ravno zaradi vse večjega števila podatkov na medicinskih ter znanstvenih področjih nasploh pa postajajo članki vse manj primerni za deljenje tovrstnih vsebin. Članki niso dostopni kot na primer spletna stran,

niso primerni za vsebine, ki se neprestano spreminjajo in posodabljajo ter ne podpirajo interaktivnosti med posredovanimi podatki in uporabnikom.

V literaturi se za ponazoritev izvedbe sekvenčnih procesov pogosto pojavljajo diagrami poteka (angl. *flow charts*). S pomočjo takšnih diagramov lahko na vizualen način predstavimo tudi zapletene medicinske ali znanstvene procese. Zadevo še nekoliko izpopolnimo, če te procese zapišemo v obliko algoritmov, torej kot natančno določeno, nedvoumno in izvedljivo zaporedje končnih korakov, ki nas običajno pripelje do nekega rezultata.

Diplomska naloga je bila usmerjena v izdelavo spletne platforme za oblikovanje in deljenje medicinskih algoritmov. S pomočjo takšne spletne platforme bi omogočili znanstveni skupnosti ter širši javnosti dostop do velike količine vizualno predstavljenih podatkov, ki bi se lahko pogosto posodabljali. Platforma bi podpirala tudi interaktivnost med posredovanimi podatki in uporabnikom, kar pomeni, da vizualna predstavitev ne bi bila statična, ampak bi nad vizualizacijo lahko vršili različne ukaze, ki bi vsebino še dodatno obogatili.

Platforma, ki jo predlagamo v tem delu, je sestavljena iz dveh delov, tj. dela namenjenega neregistriranemu in dela namenjenega registriranemu uporabniku. Neregistrirani uporabnik lahko dostopa do podatkovnih vizualizacij, ki so po vsebini namenjene in dostopne javnosti. Registrirani uporabnik ima določene uporabniške pravice, ki definirajo, katere funkcionalnosti platforme lahko koristi. V splošnem lahko vsi registrirani uporabniki dostopajo do vseh dodanih vizualizacij ter dodajajo in urejajo lastne vizualizacije. Določeni uporabniki imajo nadzor nad urejanjem in potrjevanjem vizualizacij ter imajo možnost spreminjanja uporabniških pravic ostalih uporabnikov.

Predstavljena platforma omogoča poenoteno, centralizirano ter široko dostopno rešitev za prikazovanje ter hranjenje medicinskih algoritmov. Pričakujemo, da bo kot taka v bližnji prihodnosti omogočila enostavno deljenje medicinskega znanja tako znotraj znanstvenoraziskovalne skupnosti kot tudi z bolniki.

Poglavje 2

Medicinski algoritmi

Algoritmi so natančno določena, nedvoumna in izvedljiva zaporedja končnih korakov, ki nas v določenem številu korakov pripeljejo do nekega končnega rezultata. Uporabljamo jih za reševanje računskih problemov, procesiranja podatkov in avtomatičnega sklepanja. Algoritme lahko zapišemo oziroma predstavimo na več načinov: v naravnem jeziku, s pomočjo psevdokode, kot program v enem od programskih jezikov ali pa grafično s pomočjo diagrama poteka.

Vsaka izmed omenjenih metod prikazov algoritmov ima določene prednosti in pomanjkljivosti. V naravnem jeziku lažje predstavimo potek dogodkov, vendar imamo probleme z ohranjanjem nedvoumnosti. Zapis kode v enem izmed programskih jezikov lahko natančno določi posamezne korake, vendar zamegli splošno razumevanje postopka. S pomočjo diagramov poteka lahko ohranimo tako nedvoumnost posameznih korakov, kakor tudi celostno razumevanje postopka.

2.1 Uporaba medicinskih algoritmov

V medicini so algoritmi potrebni za prikaz kompleksnih procesov na področju diagnostike, zdravljenja in preprečevanja bolezni ali bolezenskih stanj. Medicinski algoritmi, zaradi jasne, vizualne predstavitve podatkov in predaje

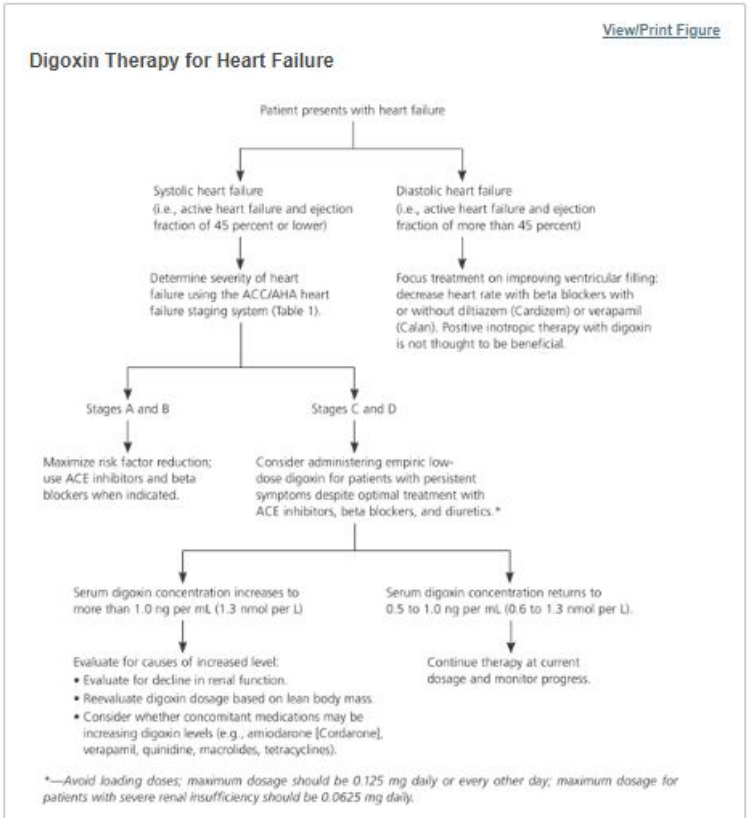
informacij, pripomorejo k bistveno hitrejši interpretaciji nekega medicinskega pojava in hitrejšemu odločanju o nadaljnem ukrepanju. Namen medicinskih algoritmov je tudi izboljšati in standardizirati odločitve, ki jih zdravstveno osebje sprejema ob izvajanju zdravstvene oskrbe. S pomočjo algoritmičnega avtomatizma pa se lahko prepreči oziroma zazna tudi marsikatera potencialna človeška napaka. Medicinski algoritmi so uporabljeni tudi kot pripomočki za izobraževanje in raziskovalno delo in pripomorejo k nižanju stroškov zdravljenja, saj omejijo uporabo nepotrebnih testiranj in terapij.

2.2 Pregled obstoječih platform za deljenje medicinskih algoritmov

V literaturi in na spletu lahko najdemo veliko virov in platform, ki ponujajo medicinske algoritme v različnih oblikah predstavitve. Mnoge izmed teh platform so ponujene v obliki mobilnih aplikacij, so plačljive ali pa ne omogočajo centraliziranega shranjevanja algoritmov. Osredotočili se bomo na dve razširjeni platformi, ki sta dostopni preko spletnega brskalnika.

2.2.1 Platforma American Academy of Family Physicians (AAFP)

Na spletni strani American Academy of Family Physicians (AAFP) [3] lahko najdemo obsežen seznam povezav do specifičnih znanstvenih člankov, ki vsebujejo medicinske algoritme. Ob kliku na določeno povezavo smo preusmerjeni na novo stran, kjer se prikaže znanstveni članek, ki vsebuje izbrane algoritme. Algoritmi so predstavljeni v obliki diagramov poteka in so statični ter nimajo poenotene vizualne predstavitve. Iz primera na sliki 2.1 je razvidno, da lahko s pomočjo večje količine besedila dobro predstavimo vsebino algoritma. Če je vsebina algoritma zelo obsežna in posledično vsebuje veliko besedila, vizualizacija izgubi na preglednosti.



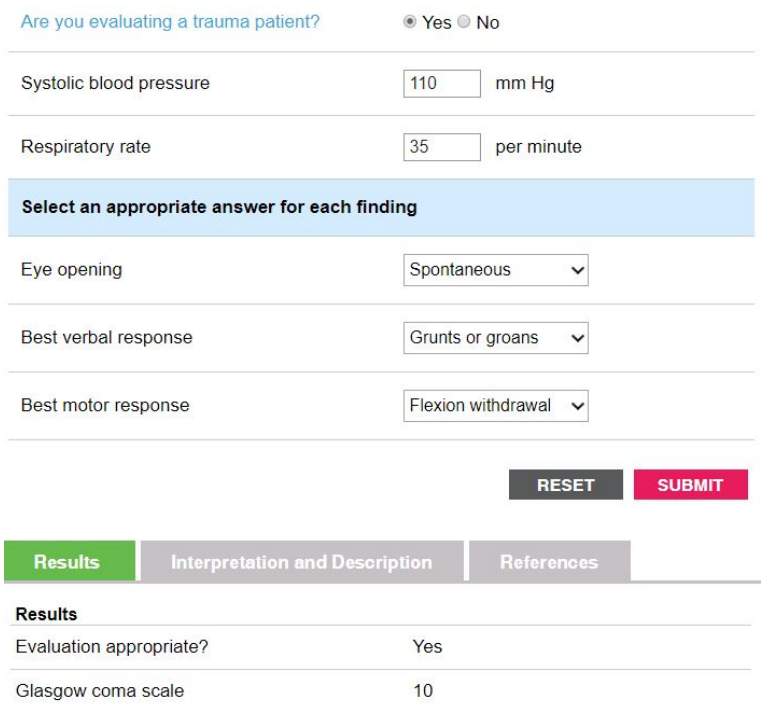
Slika 2.1: Primer medicinskega algoritma iz članka [4] na strani AAFP.

Spletna stran nudi dostop do algoritmov le preko člankov, ki so bili v preteklosti objavljeni v znanstvenih revijah. Določeni članki niso javno dostopni, kar onemogoča splošen dostop do algoritmov, ki so v njih vsebovani. Poleg tega vizualizacije niso poenotene, saj različni avtorji prispevkov vizualizacije tudi različno oblikujejo. Obstoječih algoritmov ni možno urejati oziroma posodabljati, prav tako pa ni možno dodajati lastnih algoritmov.

2.2.2 Platforma The Medical Algorithm Project (MEDAL)

The Medical Algorithm Project [5] je drugi primer spletne platforme za deljenje medicinskih algoritmov. Algoritmi so predstavljeni v obliki interak-

tivnega vprašalnika (primer take vizualizacije prikazuje slika 2.2) ali pa kot spletni kalkulatorji. Oba načina predstavitve imata skupno pomanjkljivost, saj uporabnik dejanskega algorima ne vidi. Uporabnik lahko spremlja le vstopne parametre algoritma ter končni rezultat.



Slika 2.2: Primer medicinskega algoritma iz strani [6] projekta MEDAL.

Pomankljivost te platforme je tudi v tem, da se mora uporabnik registrirati in prijaviti, če želi dostopati do algoritmov. Med drugim je platforma v procesu uvajanja licenc, kar pomeni, da določene funkcionalnosti v bližnji prihodnosti ne bodo več brezplačne. Kot v prvem primeru tudi tu običajni uporabniki ne morejo prispevati svojih algoritmov, niti ne morejo posodabljati obstoječih.

2.3 Dobre prakse in problemi obstoječih platform

Iz prejšnjih dveh primerov lahko vidimo, da imajo določene metode vizualizacije in deljenja medicinskih algoritmov slabosti, ki pa bi jih lahko odpravili z vzpostavitvijo lastne platforme. Platforme omogočajo dostop do vsebin z ali brez postopka registracije. V kolikor je algoritem po vsebini dovolj preprost ali celo namenjen uporabi za širšo javnost, je dostop preko obveznega postopka registracije negativna lastnost. Zagotovo pa je potrebno strokovno zahtevnejše ali morda celo avtorske vsebine omejiti na uporabnike, ki izvedejo postopek registracije.

Vizualno je algoritem najbolje predstaviti s pomočjo diagrama poteka, saj so tako vsi koraki algoritma jasno in nedvoumno prikazani. Pri takšnih vizualizacijah je potrebno omejiti prikazano količino besedila, saj lahko zaradi visoke gostote in pomanjkanja prostora algoritem hitro postane nejasen.

Dinamične vizualizacije algoritmov so bolj pregledne in informativne ter nudijo veliko več funkcionalnosti kot statične vizualizacije. Med drugim moramo biti pozorni na to, da algoritme oblikujemo in vizualiziramo v enotnem formatu, saj s tem pripomoremo k bolj informativnim in razumljivim odločitvam, ki jih sklepamo na podlagi vizualiziranih algoritmov. Dinamične in po formatu poenotene vizualizacije je tudi precej lažje dodajati, spreminjati in hraniti znotraj istega vira. Dodajanje algoritmov pri obstoječih platformah za navadne uporabnike ni možno. Algoritme lahko urejajo le odgovorni uredniki. Zaželeno je, da bi lahko algoritme dodajali vsi uporabniki. Preden bi taki algoritmi bili vidni splošni javnosti pa bi morali iti skozi postopek potrjevanja s strani pooblaščenega strokovnjaka.

Poglavje 3

Opis rešitve

Predlagana platforma naj bi omogočala preprost način za izgradnjo vizualizacij medicinskih algoritmov. Te vizualizacije naj bi bile v določenem obsegu prosto dostopne širši javnosti, določen del pa bi bil namenjen le registriranim uporabnikom. Platforma naj bi posledično zajemala tudi registracijski postopek za uporabnika. Vizualizacije naj bi bile hranjene v poenotenem formatu ter tako tudi predstavljene. Vizualizacije bi morale biti dinamične, da bi nad njimi lahko izvajali različne funkcionalnosti. Platforma naj bi o vizualizacijah hranila tudi določene podatke, kot so na primer ime, opis in namen algoritma. Omogočen naj bi bil sistem potrjevanja oblikovanih algoritmov s strani pooblaščenega uporabnika.

Pred začetkom implementacije nove platforme je bilo potrebno skrbno preučiti hibe ter izluščiti pozitivne lastnosti obstoječih platform. Platforma mora biti razdeljena na dva segmenta. Prvi segment je dostopen javnosti, torej vsakemu neregistriranemu uporabniku. Nuditi mora vpogled v javno dostopne vizualizacije ter enostaven pregled teh vizualizacij. Omogočati mora tudi postopek registracije na uporabnikovo željo. Drugi segment platforme je namenjen registriranim uporabnikom. Omogočati jim mora preprost način za dodajanje vizualizacij medicinskih algoritmov, urejanje algoritmov, pregled obstoječih algoritmov, administrativnim uporabnikom pa mora omogočati tudi urejanje uporabniških pravic ostalih uporabnikov. Registrirani uporab-

niki morajo imeti določene uporabniške pravice, ki jim omogočajo upravljanje z različnimi deli platforme.

Vizualizacije algoritmov morajo biti izvedene s pomočjo diagramov poteka. Uporabnik mora imeti na voljo preprosto orodje, ki omogoča gradnjo takšnih diagramov. Končna vizualizacija mora biti dinamična, kar pomeni, da lahko nad njo vršimo razne funkcionalnosti, ki pripomorejo k boljšemu prenosu informacij.

3.1 Opis uporabljenih tehnologij

Danes poznamo že vrsto različnih načinov oblikovanja, izgradnje ter vzdrževanja spletnih platform. Vsak dan lahko opazimo novosti na področju spletnih tehnologij, ki jih moramo pozorno spremljati, saj nam precej olajšajo delo in nam omogočajo dodajanje novih funkcionalnosti, hkrati pa uporabniku nudijo boljšo uporabniško izkušnjo. Za izdelavo predlagane platforme je bilo potrebno uporabiti veliko različnih spletnih tehnologij, knjižnic in orodij. Vse uporabljene komponente izdelane platforme so odprtokodne in prosto dostopne na spletu.

3.1.1 Spletni strežnik Apache

Apache je brezplačen in odprtokoden spletni strežnik za deljenje spletnih strani [7]. Že od leta 1996 predstavlja najbolj popularen strežnik HTTP. Njegov razvoj podpira odprta skupnost razvijalcev Apache Software Foundation, ki stremi k temu, da strežnik neprestano posodablja in mu dodaja novejšim tehnologijam primerne funkcionalnosti. Prav tako skrbijo, da je strežnik skladen z novimi različicami protokola HTTP.

3.1.2 PHP

Odprtokodni programski jezik PHP (angl. *Hypertext Preprocessor*) se uporablja za strežniško obdelovanje oziroma razvoj dinamičnih spletnih vsebin [8].

Končni uporabniki spletnih strani jezika PHP ne zaznajo, saj se v celoti izvaja na strežniku, vidni pa so rezultati njegovega delovanja. S pomočjo programskega jezika PHP lahko vzpostavimo povezavo s podatkovno bazo.

3.1.3 Podatkovna baza MySQL

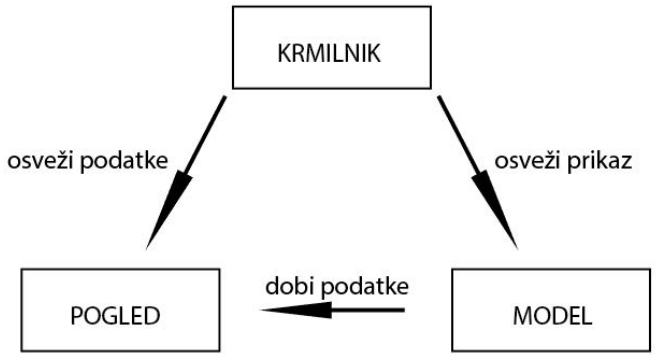
Odprtokodna implementacija relacijske podatkovne baze MySQL predstavlja sistem za upravljanje s podatkovnimi bazami [9]. Za delo s podatki uporablja jezik SQL. Je prenosljiva med operacijskimi sistemi in zaradi večopravilnosti bistvenih programskih niti mreže tudi zelo učinkovita. Kot je za relacijske podatkovne baze značilno, se podatki shranjujejo v tabele, kjer vrstice predstavljajo podatkovne vnose, stolpci pa predstavljajo atribute vnosov.

3.1.4 MVC

Prezentacijski razvijalski vzorec Model-Pogled-Krmilnik (angl. *Model-View-Controller*) je eden izmed pogosteje uporabljenih razvijalskih vzorcev spletnih aplikacij. Vzpostavljen je bil v 70. letih prejšnjega stoletja. Predstavlja osnovo za vse ostale razvijalske vzorce, njegov namen pa je jasno ločiti zadolžitve posameznega sloja glede na to, kaj kateri sloj počne. Model oskrbuje pogled s podatki, lahko pa vsebuje tudi podatkovno logiko. Pogled je predloga formata HTML, ki pridobi podatke od modela. Pogled je omejen na prikazovanje in zajemanje podatkov. Krmilnik koordinira model s pogledom, zadolžen je za kontroliranje toka dogodkov in orkestracijo danega procesa za dano akcijo - glej sliko 3.1 [10].

3.1.5 HTML 5

Označevalni jezik HTML (angl. *Hyper Text Markup Language*) je namenjen izdelavi spletnih strani v obliki, ki se prikaže končnemu uporabniku. Strukturno je jezik sestavljen iz značkovnih parov, ki določajo elemente. Vsaka značka ima tako začetno in končno značko, vmes pa lahko vpisujemo poljubno besedilo ali pa gnezdimo druge značke. Trenutna standardizirana različica, ki



Slika 3.1: Delovanje prezentacijskega razvijalskega vzorca MVC.

omogoča preprostejšo prikazovanje novejših multimedijskih vsebin ter ostalih naprednih funkcionalnosti, je HTML 5.2. HTML 5.2 podpira tudi moderne API-je kot na primer geo lokacijo, zgodovino brskanja in nalaganje ter branje uporabnikovih lokalnih datotek [11].

3.1.6 CSS

Kaskadne stilske predloge CSS (angl. *Cascading Style Sheets*) so predloge, ki preko preprostega slogovnega jezika skrbijo za prezentacijo spletnih strani. Uporabljamo jih za definicijo stila dokumentov HTML v obliki pravil, ki definirajo prikaz elementov HTML na spletni strani. Preko predlog CSS lahko elementom spreminjamo barvo, obliko, velikost, odmike, postavitev na strani in mnogo drugih atributov. Predloge CSS so bile razvite z namenom, da se struktura strani, ki jo podaja jezik HTML, loči od njene predstavitve, za katero skrbijo predloge CSS [12].

Bootstrap

Brezplačna in odprtokodna knjižnica Bootstrap temelji na kaskadnih stilskih predlogah CSS in označevalnemu jeziku HTML [13]. Namenjena je obliko-

vanju čelnega dela spletne aplikacije s pomočjo predpripravljenih predlog, ki urejajo izgled raznih elementov ter njihove postavitve. Med pomembnejše lastnosti knjižnice Bootstrap sodi odziven spletni dizajn, ki omogoča, da se elementi HTML prilagajajo velikosti zaslona uporabnikove naprave. S tem omogočimo boljšo uporabniško izkušnjo na različnih napravah.

3.1.7 JavaScript

Skriptni jezik JS (angl. *JavaScript*) je odprtokodni, visokonivojski, interpretiran in šibko tipiziran jezik za ustvarjanje dinamičnih in interaktivnih spletnih strani. Razvit je neodvisno od programskega jezika Java, vendar si z njim deli številne lastnosti in strukture. JavaScript lahko sodeluje s kodo HTML in s tem dinamično ureja njene elemente, lahko pa spreminja tudi atribute predlog CSS. Preko posebne značke lahko jezik JavaScript vgnezdimo tudi v dokument HTML. Jezik JavaScript se večinoma uporablja oziroma izvaja v spletnih brskalnikih na strani odjemalca, najdemo pa ga tudi na strežniški strani v spletnih strežnikih in podatkovnih bazah.

jQuery

Knjižnica jQuery je brezplačna in odprtokodna knjižnica skriptnega jezika JavaScript [14]. Namenjena je poenostavitvi uporabe skriptnega jezika JavaScript na strani odjemalca. Vsebuje vnaprej napisane funkcije Javascript, kar pomeni, da nam uporaba te knjižnice omogoča doseči boljše rezultate z manj programske kode. S pomočjo knjižnice jQuery lahko lažje navigiramo po spletnih straneh, izbiramo elemente spletne strani z uporabo objekta DOM (angl. *Document object model*), ustvarjamo animacije, urejamo odzive na dogodke ter ustvarjamo aplikacije AJAX.

Raphaël.js

Knjižnica Raphaël je majhna razširitev skriptnega jezika JavaScript [15]. Namenjena je delu z vektorskimi spletnimi grafikami. Preko preprostega nabora

ukazov omogoča dodajanje in spreminjanje vektorskih gradnikov, ki so zapisani v formatu SVG ali formatu VML. Vsebovalnik vektorskih gradnikov je element knjižnice Raphaël.js imenovan platno (angl. *canvas*). Gradnikom lahko dodajamo odzivnike na dogodke ter s tem ustvarimo interaktivne vektorske grafike. S pomočjo knjižnice Raphaël lahko ustvarjamo tudi naprednejše animacije.

Notify.js

Vtičnik Notify.js je razširitev knjižnice jQuery [16]. Omogoča preprosta in prilagodljiva obvestila, ki se sprožajo ob različnih dogodkih ali pa so vezana na elemente HTML.

3.1.8 AJAX

Set spletnih razvijalskih tehnik AJAX (angl. *Asynchronous JavaScript And XML*) izrablja funkcionalnosti mnogih spletnih tehnologij za kreiranje asinhronih spletnih aplikacij. Izvaja se na strani odjemalca. Tehnologija AJAX nam omogoča pridobivanje podatkov s strežnika ter njihovo prikazovanje na spletni strani, ne da bi morali to spletno stran ponovno naložiti. Tehnologija AJAX podpira tudi asinhrono pošiljanje podatkov na strežnik. Asinhronost doseže preko posebnih zahtevkov XMLHttpRequest, ki jih s pomočjo jezika JavaScript v ozadju pošilja na strežnik. Zahtevki XMLHttpRequest so lahko zapisani v formatu XML ali formatu JSON, kar je bolj pogosto. Tehnologija AJAX izboljša uporabniško izkušnjo, saj za prikaz novih podatkov strani ni potrebno ponovno nalagati.

3.1.9 Format JSON

Format datotek JSON (angl. *JavaScriptObjectNotation*) uporablja zapis v naravnem jeziku oblike ključ in vrednost za prenos podatkovnih objektov. Največkrat se uporablja kot format v zahtevkih XMLHttpRequest tehnologije AJAX in kot format za shranjevanje objektov.

3.1.10 Format SVG

Format SVG (angl. *Scalable Vector Graphics*) je slikovni format, ki temelji na jeziku XML. Uporablja se za prikazovanje dvodimenzionalih vektorskih slik. Ker temelji na jeziku XML, lahko posamezne elemente slike SVG naslavljamo, iščemo in stiskamo. Slike SVG je možno urejati s pomočjo urejevalnika besedil ali s pomočjo specializirane programske opreme za risanje.

3.1.11 Format PNG

Slikovni format PNG (angl. *Portable Network Graphics*) je relativno nov in nepatentiran format rastrskih slik. Razvit je bil z namenom pošiljanja slik preko spleta, zato podpira samo barvne prostore RGB. Podpira tudi brezizgubno kompresijo podatkov.

3.1.12 Git

Tehnologija Git je sistem za nadzor in upravljanje z izvorno kodo. Preko beleženja oddanih različic omogoča povrnitev trenutnega stanja na starejšo različico. Podpira reševanje konfliktov, ki nastanejo ob sokratnem spreminjanju datotek. S pomočjo tehnologije Git lahko hitreje razvijamo potek dela, ohranjamo integriteto podatkov ter nudimo distribuirano podporo nelinearnim potekom dela.

3.2 Opis strukture platforme

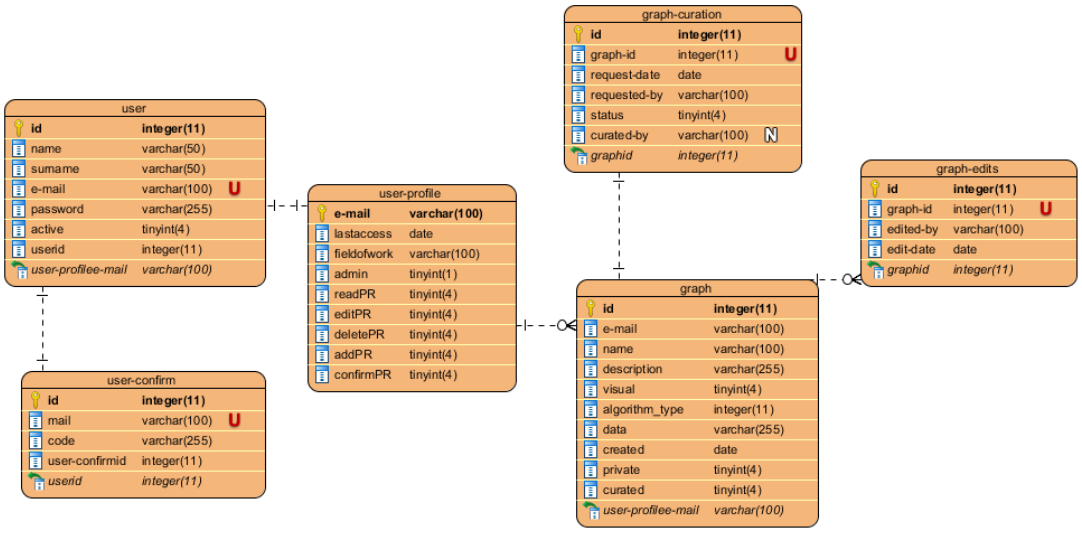
3.2.1 Model platforme

Platforma je zgrajena po modelu odjemalec-strežnik. Na strani odjemalca je platforma dostopna preko spletnega brskalnika, kar omogoča neposredno možnost uporabe aplikacije brez potrebe po dodatnih predhodnih namestitvah programske ali strojne opreme. Uporabnik s pomočjo spletnega brskalnika in vmesnih protokolov, predvsem zahtevka GET in POST protokola

HTTP, preko spleta operira s podatki, ki so na strežniku Apache. Strežniški del platforme je zgrajen na podlagi paradigme MVC, ki omogoča jasno in učinkovito strukturo ter preprost dostop do podatkov in strežniške funkcionalnosti platforme. Strežniški del ima preko razreda PDO (angl. *PHP Data Object*) dostop do podatkovne baze MySQL, ki se nahaja na istem gostiteljskem sistemu kot strežnik.

3.2.2 Podatkovna baza

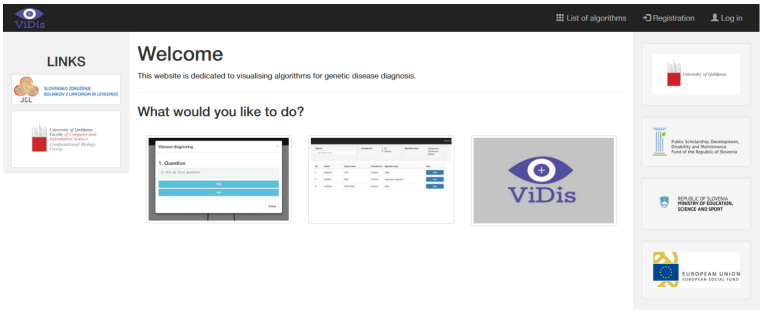
Podatkovna baza je zgrajena iz šestih tabel. Tabela 'user' hrani osebne podatke o uporabniku, tabela 'user-profile' hrani podatke o uporabniškem profilu in uporabniških pravicah, taleba 'user-confirm' pa hrani potrditveni ključ za zaključek postopka registracije uporabnika. Tabela 'graph' hrani vizualizacije algoritmov ter njihove metapodatke, tabela 'graph-edits' hrani evidenco sprememb algoritmov, tabela 'graph-curation' pa hrani zahtevke za kuracijo algoritmov. Relacijske povezave med tabelami so prikazane na sliki 3.2.



Slika 3.2: Podatkovna baza platforme.

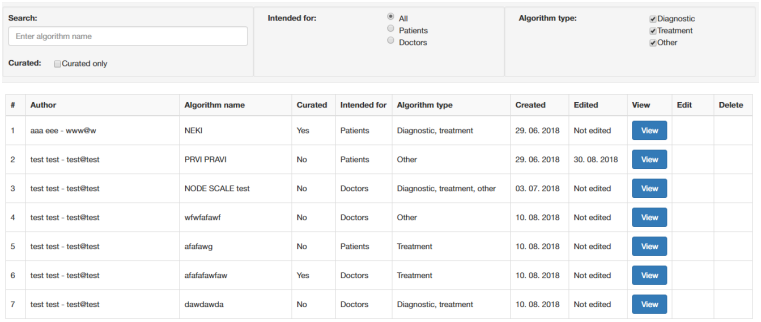
3.2.3 Neregistriran uporabnik

Na strani odjemalca je platforma razdeljena na dva segmenta. Prvi segment je namenjen neregistriranemu uporabniku. Na domači strani se na vrhu strani nahaja navigacijska vrstica s povezavami do strani za izpis seznama medicinskih algoritmov, strani za registracijo in strani za prijavo v platformo. Na skrajni levi strani navigacijske vrstice je logotip platforme, ki je hkrati tudi povezava na domačo stran. Na sredini telesa strani so povezave do strani za prikaz podrobnega predstavitvenega opisa platforme, seznama medicinskih algoritmov ter povezava do strani za diagnosticiranje bolnikov, ki pa ni del te diplomske naloge. Domača stran je prikazana na sliki 3.3.



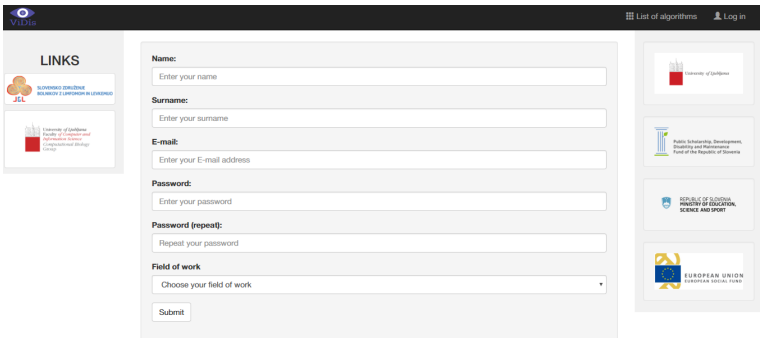
Slika 3.3: Domača stran platforme.

Na strani za prikaz opisa platforme je predstavljen namen razvoja platforme, zastavljeni cilji ter doseženi rezultati. Na strani za izpis seznama medicinskih algoritmov prikazanega na sliki 3.4 se neregistriranemu uporabniku izpišejo vsi algoritmi, ki so namenjeni javnemu dostopu. Uporabnik ima možnost filtriranja algoritmov preko večnivojskega filtra za iskanje po tabeli. Filter ni izključujoč, kar pomeni da lahko algoritme iščemo na podlagi različnih parametrov.



Slika 3.4: Izpis seznama medicinskih algoritmov.

Na strani za registracijo lahko neregistrirani uporabnik izpolni prikazan obrazec. Obrazec vsebuje vnosna polja za ime, priimek, e-poštni naslov ter geslo in ponovitev gesla. Po končanem izpolnjevanju obrazca, pošlje platforma uporabniku potrditveno e-pošto ter preusmeri uporabnika na stran za ponovni vnos e-poštnega naslova. Stran za registracijo je prikazana na sliki 3.5.



Slika 3.5: Stran za registracijo.

3.2.4 Registriran uporabnik

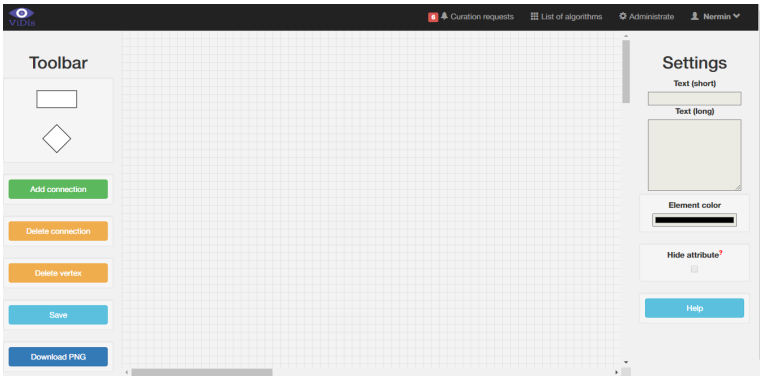
Registriran uporabnik se lahko na strani za prijavo v sistem prijavi z e-poštnim naslovom in geslom. Drugi segment platforme predstavlja funkcionalnosti, ki so na voljo registriranemu in prijavljenemu uporabniku.

Uporabniške pravice

Vsak registriran uporabnik ima določene uporabniške pravice. Te pravice lahko spreminja le administrator platforme, ki ima v ta namen prav tako določeno uporabniško pravico. Uporabniških pravic je šest. Hranijo se v podatkovni bazi, podrobneje v tabeli 'user-profile'. Vsaka omogoča izvajanje določenih funkcionalnosti znotraj platforme. Pravica 'read' omogoča pregledovanje objavljenih algoritmov. Pravica 'add' omogoča gradnjo in dodajanje novih vizualizacij algoritmov. Pravica 'edit' omogoča spreminjanje obstoječih algoritmov. Pravica 'delete' omogoča brisanje obstoječih algoritmov. Pravica 'curate' omogoča potrditev ali zavrnitev prošnje za kuracijo algoritma. Pravica 'admin' omogoča urejanje uporabniških pravic ostalih uporabnikov, ne omogoča pa dostopa do ostalih funkcionalnosti.

Orodje za gradnjo vizualizacij

Prijavljenemu uporabniku, ki ima uporabniško pravico 'add', se v navigacijski vrstici izpiše povezava do strani za gradnjo vizualizacij 'create algorithm'. Takšna povezava je vsebovana tudi v telesu domače strani. Stran za gradnjo vizualizacij je sestavljena iz treh komponent (glej sliko 3.6). Na levi strani se nahaja orodna vrstica (angl. *toolbar*), na sredini je glavno platno ter na desni strani okno z nastavitvami (angl. *settings*).



Slika 3.6: Stran za gradnjo vizualizacij.

Orodna vrstica vsebuje osnovne elemente za gradnjo vizualizacij. Strukturna elementa sta osnovno vozlišče, ki je predstavljeno s pravokotnikom, in odločitveno vozlišče, ki je predstavljeno z rombom. V orodni vrstici se nahaja še gumb za povezovanje elementov, gumb za brisanje povezav, gumb za brisanje elementov ter gumb za shranjevanje vizualizacije, ki odpre pogovorno okno za vnos metapodatkov algoritma. Centralno platno je polje, kjer se gradijo algoritmi s pomočjo elementov iz orodne vrstice. Velikost platna je 3000 $\times$ 3000 slikovnih točk. Za ozadje platna je nastavljena mreža, ki omogoča lažjo poravnavo elementov na platnu. Okno z nastavitvami vsebuje atribute elementov, ki se nahajajo na platnu. Ti atributi so kratek opis, dolg opis, barva ter skrivanje barve. V nastavitvenem oknu se nahajata še gumb za pomoč in gumb za shranjevanje vsebine platna v slikovnem formatu PNG.

Filter za iskanje algoritmov in pregledovalnik vizualizacij

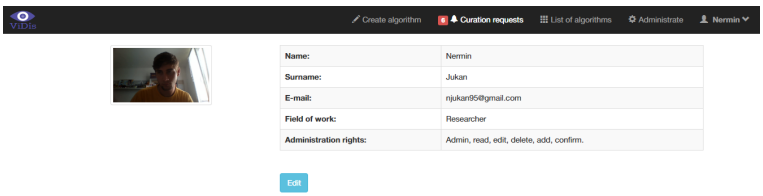
Prijavljen uporabnik, ki ima nastavljeno uporabniško pravico 'view', lahko preko povezave 'list of algorithms' dostopa do seznama, kjer se mu v tabeli izpišejo vsi vnešeni algoritmi. Prikazani podatki o algoritmih so identifikacijska številka algoritma, avtor algoritma, stanje kuracije, namen algoritma, tip algoritma, datum izgradnje algoritma ter gumbi za pregled, urejanje ali brisanje algoritma, ki so seveda prikazani le, če ima uporabnik nastavljene ustrezne uporabniške pravice. Nad tabelo se nahaja polje z iskalnimi filtri. Prvi del filtra je namenjen besedilnemu iskanju imen algoritmov, vsebuje pa tudi potrditveno polje za prikazovanje algoritmov, ki so bili preko postopka kuracije potrjeni. Drugi del filtra določa prikazovanje algoritmov glede na njihov namen. Tretji del filtra prikaže algoritme, ki so določenega tipa. Filter ni izključujoč, kar pomeni, da se končni rezultat prikaže na podlagi vseh treh delov filtra. Filter si lahko predstavljamo kot cevovod, ki ima tri dele. Vsak del izloči določene algoritme in ostale pošlje do naslednjega dela.

Uporabnik, ki ima nastavljeno uporabniško pravico 'view', lahko v tabeli algoritmov izbere pogled določenega algoritma. Izvede se preusmeritev na pregledovalnik algoritmov, kjer se algoritem naloži na osnovno platno.

Uporabnik tukaj algoritmov ne more urejati, lahko pa uporablja določene interaktivne funkcionalnosti algoritmov, ki jih nudi platforma. Na desni strani prikazovalnika ima uporabnik prikazane še vse ostale algoritme, tako da lahko menja prikazan algoritem, ne da bi pred tem moral zapustiti prikazovalnik.

Uporabniški profil

Na desni strani navigacijske vrstice se nahaja ime prijavljenega uporabnika ter spustni seznam, ki vsebuje sliko uporabniškega profila, ime in priimek uporabnika, uporabnikov e-poštni naslov, povezavo do uporabnikovega uporabniškega profila in gumb za odjavo iz sistema. Uporabniški profil je stran, kjer se uporabniku izpišejo njegovi podatki, torej ime in priimek, e-poštni naslov s katerim je registriran, področje delovanja ter uporabniške pravice. Na levi strani je prikazana profilna slika uporabnika. Uporabnik lahko s klikom na gumb 'edit' uredi svoje podatke ali pa naloži novo uporabniško sliko. Uporabniški profil je prikazan na sliki 3.7.



Slika 3.7: Stran uporabniškega profila.

Urejanje uporabniških pravic

Uporabnik z uporabniško pravico 'admin' ima dostop do nadzorne plošče, kjer lahko ureja uporabniške pravice ostalih uporabnikov. V tabeli ima prikazane vse uporabnike urejene po padajočem abecednem vrstnem redu. Ob kliku na gumb 'edit' lahko za posameznega uporabnika spremeni uporabniške pravice, ki jih nato shrani ob kliku na gumb 'save' v vrstici tega uporabnika. Prijavljeni administrativni uporabnik ne more spreminjati svojih uporabniških

pravic. S tem zagotovimo, da ima sistem vsaj enega uporabnika z uporabniško pravico 'admin'.

Potrjevanje algoritmov

Določeni uporabniki imajo nastavljeno uporabniško pravico 'curate', ki jim omogoča kuriranje algoritmov. V navigacijski vrstici imajo povezavo do strani za urejanje kuracij, poleg povezave pa se dinamično izpisuje še število odprtih zahtevkov za kuracijo. Na strani za kuriranje se uporabniku v tabeli izpišejo posamezni zahtevki. Vsak zahtevek vsebuje identifikacijsko številko algoritma, ime algoritma, ime avtorja zahtevka, datum nastanka zahtevka, status zahtevka, odgovorni urednik zahtevka ter gumba za ogled algoritma in odziva na kuracijski zahtevek. Vsak kurator se lahko odzove na vse kuracijske zahtevke. Zahtevek lahko odobri ali zavrne, odločitev pa pojasni v odzivu na zahtevek. Po zaključenem urejanju zahtevka je avtor zahtevka preko e-pošte obveščen o spremenjenem statusu zahtevka.

3.3 Implementacija platforme

3.3.1 Postavitev razvojnega okolja

Prva naloga implementacije platforme je bila priprava razvojnega okolja. Na operacijski sistem Windows 10 smo namestili brezplačni, odprtokodni paket XAMPP, ki vsebuje skupek odprtokodne programske opreme. Paket XAMPP sestavlja spletni strežnik Apache, podatkovna baza MySQL ter programski jezik PHP. Namestili smo tudi integrirano razvojno okolje JetBrains PhpStorm 2018, ki nudi podporo programskim jezikom PHP, HTML, JavaScript ter CSS. Omogoča tudi sinhronizacijo s podatkovno bazo. Na spletnem poratalu GitHub smo inicializirali nov repozitorij za nalaganje in spremljanje sprememb programske kode.

3.3.2 Implementacija modela MVC

Za osnovni model MVC nismo izbrali nobenega obstoječega ogrodja, zato smo model MVC ustvarili sami. Najprej smo izdelali hierarhično strukturo direktorijev. Dodali smo direktorije za jedro aplikacije, krmilnike, poglede, modele, podatkovno bazo ter ostale vire. Nato smo napisali jedro platforme, ki omogoča navigacijo po drevesni strukturi na podlagi prejetega naslova URL, ter osnovni razred „Kontroler“. S pomočjo datoteke „.htaccess“ smo omejili dostop do jedra aplikacije preko brskalnika. Izdelali smo razširitev razreda „Kontroler“ za dostop do domače strani ter osnovni pogled domače strani. Osnovno postavitev domače strani smo oblikovali z jezikoma HTML in CSS ter knjižnico Bootstrap. Ustvarili smo navigacijsko vrstico, levi in desni stranski razdelek ter osrednje telo strani. Vse elemente smo napolnili z začasno vsebino, ki smo jo kasneje nadomestili z ustreznimi slikami ter povezavami za navigiranje po platformi.

3.3.3 Postavitev podatkovne baze

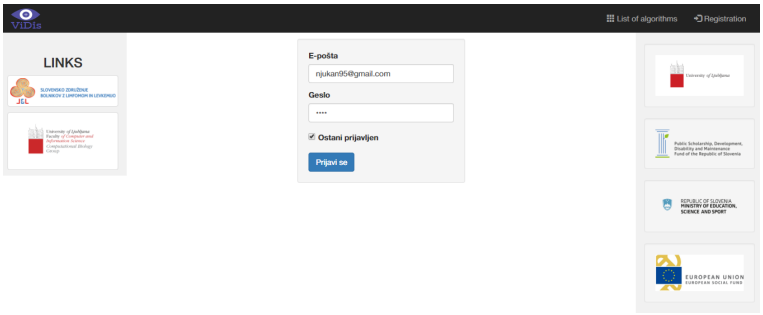
S pomočjo brezplačne spletne programske opreme phpMyAdmin smo ustvarili novo podatkovno zbirko. Določili smo pravice za dostop do podatkovne zbirke ter ustvarili prazno tabelo za testiranje dostopa. Za povezovanje s podatkovno bazo smo uporabili razred PDO (angl. *PHP Data Object)*, ki predstavlja konsistenčen vmesnik za dostop do podatkovne baze. Razredu PDO smo nastavili parametre za avtentikacijo ter ustrezen gonilnik za podatkovno bazo MySQL. V vsakem trenutku lahko obstaja samo eno instanca razreda PDO. S tem smo omogočili le eno odprto povezavo do podatkovne baze in boljši izkoristek virov na strežniku. Nastavili smo persistenčne povezave do podatkovne baze, kar prav tako izboljša delovanje platforme.

3.3.4 Izdelava sistema za registracijo in prijavo

Na strežniku smo izdelali nov krmilnik za usmerjanje na strani, povezane s postopkom registracije. Dodali smo osnovni pogled za registracijo, ki vsebuje

obrazec za vnos uporabniških podatkov. Obrazec sestavljajo polja za vnos imena, priimka, e-poštnega naslova, gesla, ponovitve gesla ter spustni seznam za izbiro področja delovanja. Platforma preko zahtevka POST vnešene podatke dostavi do krmilnika za registracijo. Na strežniku smo uredili validacijo uporabniških vnosov ter dodali ustrezne odgovore na nepravilno izpolnjen obrazec. Strežnik v primeru napak vnosa uporabnika preusmeri nazaj na osnovno stran za registracijo ter mu izpiše obvestilo. Krmilniku za registracijo smo dodali funkcionalnost pošiljanja e-pošte na uporabnikov e-poštni naslov. V podatkovni bazi smo ustvarili tabelo „user“, ki hrani uporabnikove podatke. Ustvarili smo še tabelo „user-profile“, ki hrani podatke o uporabniškem profilu ter tabelo „user-confirm“, ki hrani ključ za aktivacijo uporabniškega računa. Dodali smo pogled za potrditev aktivacije uporabniškega računa ter vnosno polje za ponoven vnos uporabnikovega e-poštnega naslova. V krmilniku smo dodali funkcijo za pošiljanje potrditvene e-pošte, ki vsebuje unikaten ključ za aktivacijo računa. Dodali smo še poizvedbo za posodobitev atributa *active* v tabeli *user*.

Za prijavo v sistem smo dodali nov krmilnik ter pogled, ki vsebuje obrazec za prijavo. Obrazec sestavljajo vnosna polja za uporabnikov e-poštni naslov, geslo ter izbirno polje za ohranitev uporabnikovih podatkov za prijavo. Stran za prijavo vidimo na sliki 3.8. Ohranjanje podatkov za prijavo smo omogočili z uporabo piškotkov. Nastavili smo validacijo uporabnikovih vnosov z uporabo funkcije *htmlspecialchars* ter odgovore na neuspešne prijave v sistem. V navigacijsko vrstico smo dodali povezavi na stran za registracijo in stran za prijavo. Ob uspešni prijavi v sistem smo uredili vzpostavitev seje, ki hrani podatke o trenutno prijavljenem uporabniku ter njegove uporabniške pravice. Prijavljenemu uporabniku smo iz navigacijske vrstice skrili povezavi do strani za registracijo in prijavo ter s pomočjo seje onemogočili dostop do teh dveh strani. Nastavili smo tudi preusmeritev na domačo stran. V navigacijsko vrstico prijavljenega uporabnika smo dodali spustni seznam, ki prikazuje podatke o prijavljenem uporabniku ter gumb za odjavo iz platforme.



Slika 3.8: Stran za prijavo v platformo.

3.3.5 Izdelava uporabniškega profila

Dodali smo krmilnik za navigiranje po straneh uporabniškega profila ter dva pogleda uporabniškega profila. Prvi pogled prikazuje podatke o prijavljenem uporabniku. Podatke smo prikazali v obliki tabele. Poleg tabele smo prikazali uporabnikovo profilno sliko. Nastavili smo privzeto profilno sliko, ki jo lahko uporabnik nadomesti s svojo sliko. Na dno strani smo postavili gumb s povezavo na pogled za urejanje profila. Drugi pogled omogoča spreminjanje uporabniških podatkov. Vnosna polja so vmeščena v tabelo, s tem smo ohranili konsistenčnost izgleda strani. Omogočili smo spreminjanje imena, priimka ter področja delovanja, ne pa tudi e-poštnega naslova, ker je uporabljen kot unikatni identifikator uporabnika. Pod sliko smo postavili gumb za nalaganje nove profilne slike. Gumb je preko unikatnega identifikatorja in knjižnice jQuery vezan na dogodek, ki v ločeni datoteki JavaScript sproži izbiro in nalaganje nove slike. V spustni seznam navigacijske vrstice smo dodali uporabnikovo profilno sliko ter gumb s povezavo do uporabniškega profila.

3.3.6 Izdelava orodja za gradnjo vizualizacij

Ustvarili smo krmilnik ter pogled orodja za gradnjo vizualizacij. V navigacijsko vrstico smo dodali povezavo na osnovni pogled. Stran smo razdelili na

tri segmente. Na levo smo postavili vsebovalnik za „orodno vrstico“ (angl. *toolbar*), na sredini smo inicializirali vsebovalnik za glavno platno (angl. *canvas*) in na desno smo postavili vsebovalnik za „nastavitveno vrstico“ (angl. *settings*).

V orodno vrstico smo dodali značko „<a>“. Značka „<a>“ omogoča poteg vsebine značke na platno. V značko „<a>“ smo nato vgnezdili značko „<svg>“, v značko „<svg>“ pa smo dodali osnovni gradnik vizualizacij „proces“, ki je ponazorjen s pravokotnikom. Identičen postopek smo ponovili za gradnik „odločitveno vozlišče“, ki ga ponazarja romb. Obema elementoma smo nastavili tudi identifikator ter omogočili poteg z atributom *draggable*. Elementoma smo pripojili tudi rokovalnik dogodkov za dogodek *ondragstart*, ki ob pričetku potega elementa v atribut *dataTransfer* doda unikatni identifikator elementa. Orodni vrstici smo dodali še pet namenskih gumbov. Prvi gumb ustvari povezavo med dvema elementoma na platnu. Drugi gumb zbriše povezavo med dvema elementoma na platnu. Tretji gumb zbriše element iz platna. Četrti gumb prikaže pogovorno okno za shranjevanje algoritma. Peti gumb sproži prenos vsebine platna v slikovnem formatu PNG. Za pretvorbo platna v slikovni format PNG smo v orodno vrstico dodali skrito platno in vsebovalnik za sliko formata PNG. Dodali smo tudi skripto Raphaël.Export [17], ki skrbi za pretvorbo iz formata Raphaël.paper v format PNG.

Nastavitvena vrstica je namenjena spreminjanju atributov posameznih elementov na platnu. Vsak element na platnu ima definirane lastne vrednosti, ki se prikažejo v vnosnih oziroma izbirnih poljih nastavitvene vrstice. V nastavitveno vrstico smo dodali dve vnosni polji za besedilo. V prvem polju smo število besedilnih znakov omejili na 100. V drugo polje smo dodali značko „<pre>“, ki ohranja format besedila. Drugemu vnosnemu polju nismo omejili števila znakov. Pod vnosnimi polji za besedilo smo dodali pogovorno okno za izbiro barve, potrditveno polje za dodajanje atributa *hide* in gumb, ki prikaže pogovorno okno z vsebino za pomoč uporabniku. Vsa vnosna polja smo nastavili tako, da so ob prihodu na stran onemogočena.

Po postavitvi orodne in nastavitvene vrstice smo na spletu poiskali knjižnico JavaScript, ki omogoča interaktivno gradnjo grafik SVG. Našli smo knjižnico Raphaël.js, ki je dobro dokumentirana in vsebuje velik nabor funkcionalnosti. Knjižnico smo prenesli v direktorij za vire ter uvozili v pogled za gradnjo vizualizacij. V vsebovalniku glavnega platna smo ustvarili instanco Raphaëlovega platna. Dimenzije smo določili na 3000 × 3000 slikovnih točk. Ker je platno segalo izven okvirjev vsebovalnika, smo vsebovalnik prilagodili tako, da smo mu v datoteki *main.css* dodali dva drsnika. En drsnik premika pogled na platnu v smeri osi X, drugi pa v smeri osi Y. V predlogi CSS smo platnu za ozadje nastavili mrežo. Na platno smo s pomočjo skripte Raphaël Pan-Zoom Plugin [18] dodali funkcionalnost približevanja in oddaljevanja.

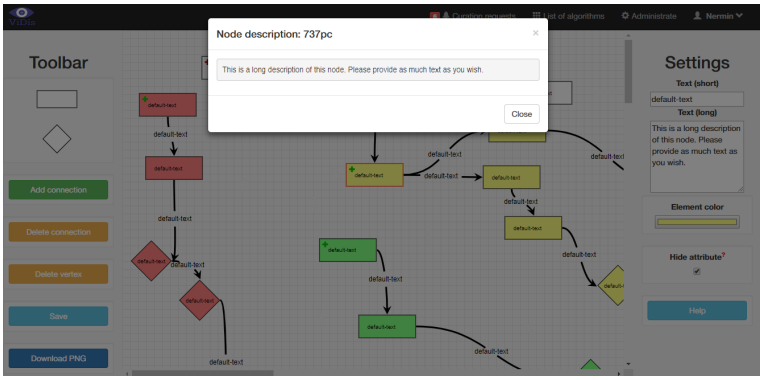
Na vsebovalnik platna smo pripojili rokovalnik dogodkov za dogodek *ondrop*. Rokovalnik zazna dogodek, ko čez platno potegnemo element iz orodne vrstice in ga spustimo na platno. Rokovalnik zabeleži točne koordinate, kjer smo element spustili ter zazna identifikator elementa, ki smo ga spustili na platno. Preko zabeleženih koordinat in identifikatorja elementa lahko s knjižnico Raphaël.js izrišemo element na platno.

Knjižnica Raphaël.js omogoča združevanje elementov v množice (angl. *sets*). Ob izrisu smo tako osnovnemu elementu dodali še več drugih elementov, ki tvorijo množico. Raphaëlove množice so v jeziku JavaScript predstavljene kot tabele, zato je pomembno na katero mesto v tabeli vstavimo kateri element. Vsak inicializiran Raphaëlov element ima dodeljen unikatni identifikator. S pomočjo funkcije *canvas.getById (id)* se lahko sklicujemo na elemente katerih identifikatorje poznamo. Množice nimajo identifikatorja, zato smo ustvarili navidezni identifikator oziroma števec ter ga s pomočjo Raphaëlove funkcije *element.data (key, value)* zapisali v prvi element množice, torej ali v pravokotnik ali v romb. Vse množice shranjujemo v tabelo. Nahajajo se na indeksu, ki je bil enak vrednosti našega števca ob inicializaciji množice. Na podlagi identifikatorja elementa za izris smo določili kateri dodatni elementi se naj dodajo v množico. Tako elementu „proces“ kot elementu „odločitveno vozlišče“ dodamo Raphaëlov besedilni

element, ki hrani kratek opis vsebine elementa, elementu „proces“ pa smo dodali še dodaten podatkovni zapis *element.data ("description", "default")*, ki hrani podroben opis vsebine elementa.

Knjižnica Raphaël.js podpira rokovalnike za različne dogodke. Elementom in množicam smo tako dodali rokovalnike, ki se odzovejo na določene dogodke. Osnovna gradnika „proces“ in „odločitveno vozlišče“ imata vezana rokovalnika *mouseup* in *click*. Tip dogodka shranimo v spremenljivko *dragging-set*, ki definira kateri rokovalnik naj se odzove. Oba rokovalnika najprej sprožita funkcijo *setActive(this)*, ki označi element kot trenutno aktiven. Funkcija elementu spremeni barvo obrobe na rdečo ter napolni nastavitvena polja s podatki, ki jih hrani element. Rokovalnik *mouseup* se odziva na dogodke potega elementa. Ob potegu elementa se sproži funkcionalnost premika celotne množice elementa. Klik na element „proces“ sproži prikaz pogovornega okna, kjer se izpiše elementov podatkovni zapis *"description"*, primer zapisa lahko vidimo na sliki 3.9. Besedilni elementi imajo vezane tri rokovalnike. Prvi je rokovalnik za *mouseover* dogodek, ki poveča velikost fonta. Drugi je rokovalnik za *mouseout* dogodek, ki velikost fonta zmanjša. Tretji rokovalnik je namenjen *click* dogodku. Rokovalnik nastavi kurzor na konec besedila v prvo vnosno polje za besedilo v nastavitveni vrstici. Tudi na platno smo vezali rokovalnik namenjen *click* dogodku. Ob kliku na prazen del platna, torej del, kjer ni nobenega elementa, se *active* element ponastavi na vrednost null, kar elementu spremeni obrobo nazaj na črno. Ponastavijo se tudi vnosna polja v nastavitveni vrstici. Rokovalnike dogodkov smo dodali tudi vnosnim poljem nastavitvene vrstice. Ob spreminjanju besedila vnosnega polja se dinamično spreminja tudi besedilo v izbranem elementu, za kar je odgovoren rokovalnik za *onkeyup* dogodek. Elementom gradnika „proces“ se širina dinamično prilagaja glede na širino vsebovanega besedila. Pogovorno okno za izbiro barve spremeni barvo polnila aktivnemu elementu. Potrditveno polje za dodajanje atributa *hide* doda simbol „+“ elementu gradnika „proces“. Na „+“ simbol smo vezali rokovalnik za *click* dogodek. Rokovalnik sproži rekurzivno skrivanje elementov, ki so enake barve kot element,

ki je sprožil dogodek. Rokovalnik skrije tudi vhodne in izhodne povezave skritih elementov. Funkcionalnost dodajanja povezav smo ustvarili tako, da smo po kliku na gumb *Add connection* v posebno spremenljivko zabeležili, da gre za dogodek dodajanja povezave. To spremenljivko smo nato dodali v telo rokovalnika za *click* dogodek. Rokovalnik počaka, da uporabnik izbere dva različna elementa na platnu ter nato med njima ustvari povezavo. Ob dodajanju povezav smo onemogočili odpiranje pogovornega okna za podroben opis elementa. Brisanje povezave ali elementa deluje na enak način kot dodajanje povezave, najprej je potrebno klikniti na ustrezen gumb ter nato izbrati element na platnu.



Slika 3.9: Pogovorno okno dolgega opisa elementa.

Na gumb *Help* smo vezali odpiranje pogovornega okna za prikaz pomoči. V pogovorno okno smo vnesli kratke opise elementov ter funkcionalnosti orodja za gradnjo vizualizacij. Gumb *Save* odpre pogovorno okno za vnos metapodatkov algoritma ter shranjevanje algoritma. V oknu se nahajajo vnosna polja za ime ter opis algoritma, izbirni gumbi za določitev dostopnosti in namena algoritma, izbirno polje za določitev vsebine algoritma ter potrditveno polje za pošiljanje potrditvenega zahtevka. Shranjevanje algoritmov smo implementirali s pomočjo skripte Raphaël.JSON [19], ki objekte na Raphaëlovem platnu pretvori v format JSON. Elementom smo morali pripeti dodatne atribute, ki omogočajo uspešno povratno pretvorbo iz formata

JSON. Algoritme in njihove metapodatke shranimo v podatkovno bazo v tabelo *graph*. S knjižnico Notify.js smo na strani omogočili prikaz informativnih obvestil.

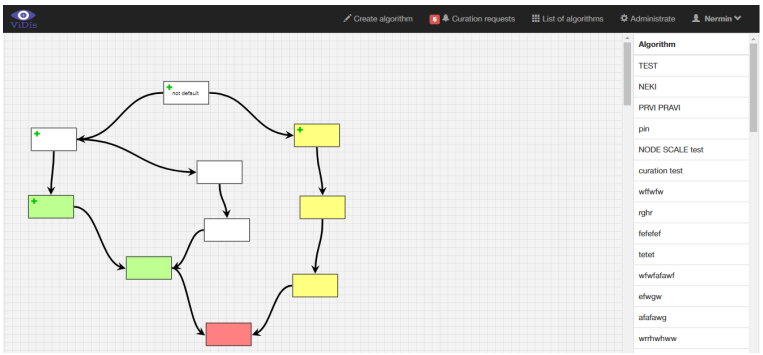
3.3.7 Prikaz, filtriranje in urejanje algoritmov

Za prikaz vseh algoritmov, ki so shranjeni v podatkovni bazi in so javno dostopni, smo implementirali nov pogled in uporabili tabelo s stilskimi dodatki knjižnice Bootstrap. V celice smo nastavili zapis podatkov o avtorju algoritma, imenu algoritma, statusu potrditve, namenu ter tipu algoritma, datumu nastanka in datumu zadnjih popravkov. Dodali smo še gumbe za pogled, urejanje in brisanje algoritma. Za pridobitev vseh podatkov smo napisali zapleteno poizvedbo SQL, ki združi podatke iz štirih različnih tabel podatkovne baze. Algoritmom, ki so zasebni in katerih lastnik je trenutno prijavljen uporabnik, smo vrstice obarvali z rumeno barvo.

Za lažje iskanje algoritmov smo izdelali večnivojski filter, ki si ga lahko predstavljamo kot cevovod. Filter je napisan v jeziku JavaScript in dinamično prikazuje končne rezultate, brez da bi pošiljal poizvedbe na strežnik. Filter v zanki preveri vsako vrstico tabele, če izpolnjuje iskane kriterije. Prvi nivo filtra predstavlja iskanje po imenu algoritma ter prikaz tistih algoritmov, katerih status potrditve je odobren. Besedilni iskalnik preveri imena algoritmov v celicah in prikaže vrstico, če celica vsebuje iskan podniz. Naslednji nivo filtrira algoritme po namenu. Dodali smo tri možne izbire, prva izbira prikaže vse algoritme, druga in tretja pa prikažeta algoritme, ki ustrezajo nastavljenim atributom. Tretji nivo filtrira algoritme po njihovem tipu. Na tem nivoju je algoritem prikazan, če tip algoritma vsebuje vsaj eno od izbranih možnosti. Po prehodu čez vse tri nivoje filtra se nam prikaže končna tabela z ustreznimi algoritmi.

Za vizualni pregled algoritmov smo ustvarili nov pogled imenovan *view.php*. Pogled je sestavljen iz Raphaëlovega platna ter stranske tabele z imeni ostalih algoritmov, vidimo ga lahko na sliki 3.10. Ob preusmeritvi na ta pogled pošljemo na strežnik zahtevek GET ter identifikacijsko številko algoritma,

ki ga zahtevamo. S skriptami JavaScript obdelamo odgovor in pretvorimo algoritem iz formata JSON v Raphaëlove elemente. V glavni skripti smo za ta pogled nastavili spremenljivko *viewOnly* na resnično. S tem smo zagotovili omejeno nalaganje funkcionalnosti algoritma. Elementom dodamo le določene rokovalnike dogodkov. Elementom smo onemogočili premikanje ali spreminjanje besedila. Onemogočeno je tudi brisanje ali dodajanje elementov. Omogočili smo, da se ob kliku na gradnik „proces“ odpre pogovorno okno, ki vsebuje podroben opis elementa. Prav tako smo omogočili funkcionalnost skrivanja vozlišč glede na njihove barve in povezave.

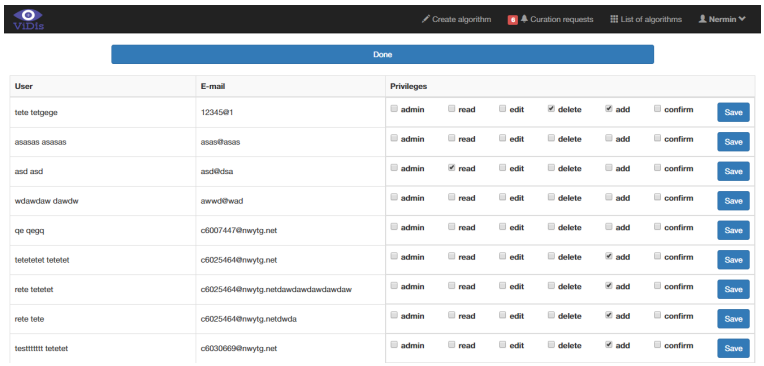


Slika 3.10: Stran za vizualni pregled algoritma.

Urejanje algoritmov smo omogočili preko orodja za gradnjo algoritmov. Na naslov za gradnjo algoritmov pošljemo zahtevek GET ter identifikacijsko številko algoritma, ki ga želimo urediti. Skripta JavaScript poskrbi za pretvorbo algoritma in formata JSON v Raphaëlove elemente. Na elemente dodamo vse rokovalnike dogodkov. Uredili smo tudi, da se pogovorno okno za shranjevanje algoritmov napolni s prejetimi podatki algoritma. Spremembe algoritmov se zabeležijo v podatkovno bazo v tabelo *graph-edits*.

3.3.8 Uporabniške pravice in administracija uporabnikov

Uporabniških pravic je šest: administrator, dodajanje, pogled, urejanje, brisanje, potrjevanje. Pravice ob prijavi uporabnika v sistem shranimo v sejo. Vsakemu novemu uporabniku smo dodelili pravico dodajanja algoritmov. To pomeni, da lahko uporablja orodje za gradnjo algoritmov in algoritme shranjuje. Prav tako smo omogočili pregled, urejanje in brisanje lastnih algoritmov. Brez pravice za pregled uporabnik ne more odpirati tujih algoritmov. Brez pravice za urejanje uporabnik ne more urejati tujih algorimov. Brez pravice za brisanje uporabnik ne more brisati tujih algoritmov. Zahtevane pravice smo definirali pred nalaganjem pogleda. V kolikor uporabnik nima ustreznih pravic, se na strežniku zgodi preusmeritev na drugo stran. Zaradi preprečevanja nepotrebnih preusmeritev smo v tabeli algoritmov skrili gumbe za izvedbo določenih funkcionalnosti. Prav tako smo v navigacijski vrstici skrili povezave na strani, do katerih uporabnik nima dovoljenja za dostop.



Slika 3.11: Stran za administracijo uporabnikov.

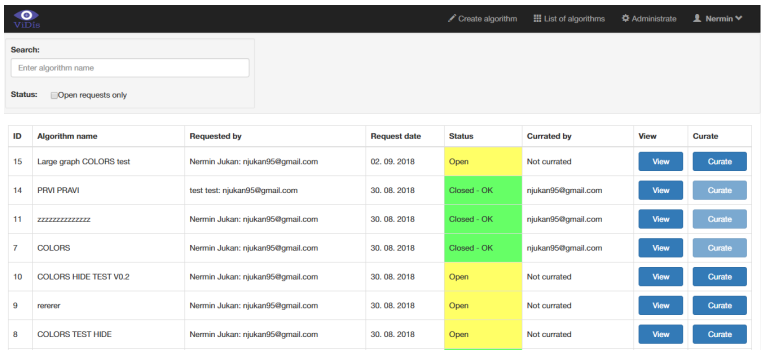
Uporabniku z uporabniško pravico admin smo omogočili administracijo drugih uporabnikov na posebni strani. Najprej smo izdelali nov krmilnik ter dodali dva nova pogleda. Oba pogleda v tabeli prikazujeta seznam uporabnikov in njihove uporabniške pravice prikazane kot potrditvena polja. Na prvem pogledu smo polja onemogočili, nad tabelo pa smo dodali gumb za pre-

usmeritev na drugi pogled, kjer lahko administrator uporabnikom spreminja pravice. Pred preusmeritvijo na drugo stran smo dodali še pogovorno okno, ki opozori uporabnika na morebitne nevarnosti spreminjanja uporabniških pravic. Na strani, kjer lahko uporabniške pravice spreminjamo, smo v zadnjo celico vsake vrstice dodali gumb *Save*. Na tej strani smo omogočili spreminjanje potrditvenih polj, zato lahko uporabnik spreminja uporabniške pravice ostalih uporabnikov. Na vsak gumb *Save* smo vezali funkcijo, ki preko zahtevka POST tehnologije AJAX zabeleži spremembe v podatkovno bazo. Onemogočili smo spreminjanje lastnih uporabniških pravic. S tem smo zagotovili, da bo vedno obstajal vsaj en administrativni uporabniški račun. Zmanjšali smo tudi verjetnost zlorab administrativnega dostopa. Stran za spreminjanje uporabniških pravic je vidna na sliki 3.11.

3.3.9 Potrjevanje algoritmov

Platformi smo dodali sistem za potrjevanje algoritmov. Najprej smo v podatkovni bazi ustvarili tabelo *graph-curations*. Nato smo na strani orodja za gradnjo algoritmov v pogovorno okno za shranjevanje dodali potrditveno polje za ustvarjanje zahtevka za potrditev algoritma. Ustvarili smo nov pogled, kjer se v tabeli prikazujejo zahtevki za potrditev algoritmov. Dostop do te strani smo omogočili le uporabnikom, ki imajo uporabnišo pravico za potrjevanje. Stran je vidna na sliki 3.12. V navigacijsko vrstico smo dodali povezavo do te strani. Poleg povezave smo v navigacijsko vrstico dodali še polje s številko. Ustvarili smo novo skripto JavaScript ter s pomočjo knjižnice jQuery in tehnologije AJAX napisali funkcijo, ki na določen časovni interval pošilja na strežnik zahtevke za pridobitev števila odprtih zahtevkov za potrditev algoritmov. Na strani za prikaz zahtevkov smo dodali še enostaven filter za iskanje po imenu algoritma ter prikaz samo tistih zahtevkov, katerih status je „odprt“. V tabelo smo dodali še gumb za odpiranje pregleda algoritma ter gumb *Curate* za obdelavo zahtevka, ki odpre pogovorno okno in ga napolni s podatki zahtevka. V pogovorno okno smo dodali izbirno polje za določitev novega statusa zahtevka in vnosno polje za besedilo,

kamor mora urednik zahtevka zapisati kratko obrazložitev. Gumb *Confirm* sproži pošiljanje e-pošte avtorju zahtevka. E-pošta vsebuje naslov algoritma na katerega se je zahtevek nanašal, odločitev urednika ali zahtevek potrdi ali zavrne ter urednikovo obrazložitev in kontakne podatke. V kolikor je bil zahtevek odobren s strani urednika, se v podatkovni bazi posodobi atribut algoritma *curated* v tabeli *graph*.



Slika 3.12: Stran za pregled potrditvenih zahtevkov algoritmov.

Poglavje 4

Primer uporabe

V tem poglavju opišemo primer uporabe aplikacije pri izdelavi novih in prikazovanju obstoječih algoritmov. Uporabo aplikacije predstavimo iz vidika končnega uporabnika.

4.1 Izdelava novega algoritma

Za ustvarjanje novega algoritma mora imeti uporabnik v platformi aktiviran uporabniški račun ter nastavljene ustrezne uporabniške pravice.

4.1.1 Registracija in prijava

Uporabnik mora najprej izvesti dvodelni postopek registracije. Prvi del postopka zahteva, da uporabnik izpolni vsa vnosna polja na strani za registracijo. Vnesti mora ime, priimek, e-poštni naslov, geslo, ponovitev gesla ter področje delovanja. Stran za registracijo je prikazana na sliki 3.5. Po oddaji obrazca aplikacija preveri, če so bila vsa polja ustrezno izpolnjena. V kolikor je prišlo do kakšne napake, aplikacija preusmeri uporabnika nazaj na stran za registracijo in izpiše katera polja more uporabnik ponovno izpolniti. Če aplikacija ne zazna nobene napake, pošlje uporabniku na e-poštni naslov potrditveno povezavo, ki vsebuje unikaten ključ, in preusmeri uporabnika na stran za aktivacijo uporabniškega računa. Sledi drugi del postopka registra-

cije, kjer mora uporabnik v svojem e-poštnem nabiralniku odpreti povezavo, ki jo je prejel od platforme. V kolikor uporabnik ni od platforme prejel nobene e-pošte, lahko s ponovnim vnosom e-poštnega naslova sproži ponovno pošiljanje nove aktivacijske povezave. Ko uporabnik odpre povezavo, platforma aktivira uporabniški račun in uporabnik se lahko na strani za prijavo prijavi v platformo. Stran za prijavo vidimo na sliki 3.8.

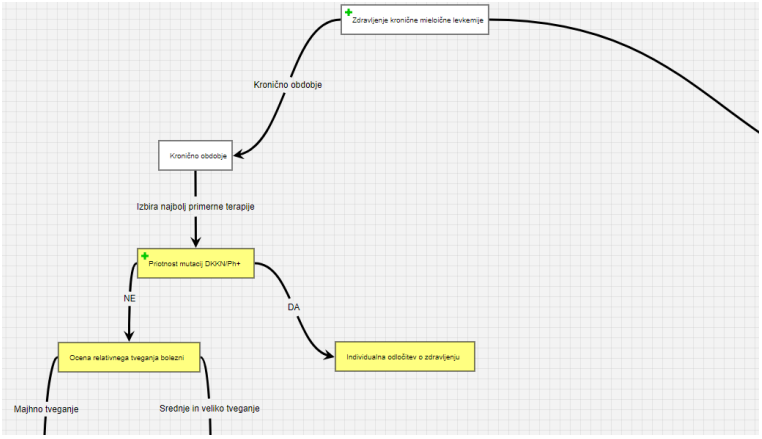
4.1.2 Gradnja algoritma

Vsak uporabnik ima ob aktivaciji računa nastavljeno uporabniško pravico 'add', ki mu omogoča gradnjo novih algoritmov. Uporabnik lahko dostopa do orodja za gradnjo vizualizacij preko povezave *Create algorithm* v navigacijski vrstici ali v telesu domače strani. Stran za gradnjo vizualizacij vidimo na sliki 3.6. Orodje ima tri komponente. Na levi strani lahko uporabnik iz orodne vrstice na platno povleče gradnike algoritmov. Na desni strani se izpisujejo vrednosti atributov izbranega elementa. Izbrani element je na platnu predstavljen z rdečo obrobo. Na sredini se nahaja centralno platno, kamor uporabnik iz orodne vrstice povleče gradnike. Obstajata dve vrsti gradnikov. Prvi je „proces“, ki ponazarja proces, stanje ali neko določeno vsebino v delu algoritma. Drugi gradnik je „odločitveno vozlišče“, ki predstavlja odsek v algoritmu, kjer se potek algoritma razdeli na več možnih delov, ki so odvisni od določenih vstopnih parametrov. Uporabnik lahko na platno doda poljubno število gradnikov. S klikom na gumb *Add connection* lahko poveže dva elementa, tako da po kliku na gumb izbere dva elementa na platnu. Vrstni red izbire elementov je pomemben, saj so povezave usmerjene. Uporabnik lahko izbriše povezave s klikom na gumb *Delete connection* ter klikom na povezavo. Prav tako lahko izbriše elemente s klikom na gumb *Delete vertex* in klikom na element, ki ga želi izbrisati. Uporabnik lahko izbranemu elementu spremeni barvo, kratek opis, dolg opis ali pa mu, če vrsta gradnika to dovoljuje, doda element za skrivanje vozlišč. S klikom na gumb *Save* uporabnik odpre pogovorno okno za shranjevanje algoritma. Vnesti mora ime algoritma, določiti dostopnost algoritma, dodati opis in namen algoritma ter določiti vsebino

algoritma. Uporabnik lahko z označitvijo potrditvenega polja zahteva tudi potrditev algoritma s strani pooblaščenega urednika.

4.2 Prikaz algoritma

Vsi shranjeni algoritmi, ki niso zasebni, so prikazani v tabeli na strani za prikaz algoritmov, do katere lahko uporabnik dostopa preko povezave *List of algorithms*. Stran za izpis seznama medicinskih algoritmov vidimo na sliki 3.4. Tukaj lahko s pomočjo večnivojskega filtra išče algoritme. Algoritme lahko sortiramo po statusu potrditve, imenu, namenu ter vsebini algoritma. Filtri niso izključujoči, zato vsi vplivajo na končen prikaz v tabeli. V kolikor uporabnik nima pravic za pregled, urejanje ali brisanje drugih algoritmov, lahko te operacije izvaja samo nad svojimi algoritmi. Ob kliku na gumb *View* določenega algoritma se uporabniku odpre novo okno, kjer se mu na platnu izriše algoritem. Stran za vizualni pregled algoritma vidimo na sliki 3.10. Tukaj uporabnik algoritma ne more spreminjati, lahko pa skriva posamezne segmente algoritma ali pa odpira pogovorna okna s podrobnimi opisi elementov. Na desni strani pogleda lahko uporabnik v tabeli izbere drug algoritem, ki se nato naloži na platno. Ob kliku na gumb *Edit* določenega algoritma je uporabnik preusmerjen na stran za gradnjo algoritmov, kjer se mu izriše izbrani algoritem. Tukaj lahko algoritem poljubno spreminja ter ponovno shrani, spremeni pa lahko tudi metapodatke algoritma. Primer izseka algoritma za zdravljenje kronične mieloične levkemije prikazuje slika 4.1.



Slika 4.1: Izsek algoritma za zdravljenje kronične mieloične levkemije.

Poglavje 5

Zaključek

Izdelana platforma predstavlja prvo spletno aplikacijo za vizualno gradnjo interaktivnih algoritmov ter njihovo deljenje tako znotraj znanstvenoraziskovalne skupnosti kakor tudi s širšo javnostjo. Dostopna je preko spletnega brskalnika, kar omogoča uporabnikom preprost dostop do vsebine platforme. Platforma podpira enotne in interaktivne vizualizacije algoritmov. Medtem ko druge spletne platforme niso namenjene pogostemu spreminjanju ali dodajanju vsebine, razvita platforma omogoča enostavno posodabljanje ter dodajanje algoritmov. Algoritme prikazuje na dinamičen način, kar pomeni da ima lahko uporabnik prikazan celoten algoritem ali pa določene segmente algoritma skrije. Preko pogovornega dialoga lahko uporabnik pridobi podrobnejši opis posameznih delov algoritma, kar ohrani vizualizacijo pregledno, poleg tega pa ne omejuje količine podatkov shranjene v posameznem vozlišču algoritma. S pomočjo sistema za potrjevanje algoritmov lahko uporabnik hitro loči med vsebinsko verificiranimi in neverificiranimi algoritmi. Sistem za potrjevanje algoritmov posredno tudi izboljšuje njihovo kakovost.

Platforma ima velik potencial in veliko možnosti za nadaljni razvoj. V orodju za gradnjo vizualizacij bi lahko spremenili izris povezav, saj trenuten izris ni optimalen. Predlagana rešitev bi vključevala deljene povezave, ki bi jih uporabnik lahko na platnu razporejal sam. Besedilnim poljem bi se lahko dodala funkcionalnost dodajanja nove vrstice oziroma ročnega nastavljanja

velikosti elementov. S tem bi uporabnik lažje razporedil elemente ter oblikoval algoritme. V urejevalnik algoritmov bi lahko dodali zgodovino urejanja. S tem bi uporabnik lahko razveljavil določene spremembe. Poleg dodajanja novih funkcionalnosti in nastavitev v orodje za gradnjo algoritmov, se lahko dopolni še funkcionalnost obveščanja uporabnikov. Uporabniški profili bi lahko služili kot nekakšne vizitke, ki bi omogočale povezovanje uporabnikov in deljenje idej oziroma predlogov. Osnovni model MVC bi se dalo izboljšati in ga narediti bolj odpornega na napake oziroma napade. Platformo je vsekakor potrebno spremljati ter upoštevati uporabniške izkušnje in predloge. Dobro bi bilo zagotoviti podporo uporabnikom ter nadgrajevati platformo z uporabo prihajajočih, novejših tehnologij.

Literatura

[1] Wolfgang Aigner, Katharina Kaiser, and Silvia Mikscg. Visualization methods to support guideline-based care management. *Studies in Health Technology and Informatics*, 139:140–159, 2008.

[2] Georgy Kopanitsa, Claudia Hildebrand, Jürgen Stausberg, and Karl-Hans Englmeier. Visualization of medical data based on EHR standards. 52:43–50, 12 2012.

[3] American Academy of Family Physicians - AFP Algorithms. Dosegljivo: `https://www.aafp.org/afp/algorithms/viewAll.html`. [Dostopano: 20. 8. 2018].

[4] Spencer A Morris, H Floyd Hatcher, and Deepa K Reddy. Digoxin therapy for heart failure: An update. *American family physician*, 74:613–618, 2006.

[5] The Medical Algorithm Project. Dosegljivo: `https://www.medicalalgorithms.com`. [Dostopano: 20. 8. 2018].

[6] Revised trauma score. Dosegljivo: `https://www.medicalalgorithms.com/revised-trauma-score`. [Dostopano: 20. 8. 2018].

[7] The apache http server project. Dosegljivo: `https://httpd.apache.org/`. [Dostopano: 23. 8. 2018].

[8] PHP. Dosegljivo: `http://php.net/`. [Dostopano: 23. 8. 2018].

[9] MySQL. Dosegljivo: https://www.mysql.com/. [Dostopano: 23. 8. 2018].

[10] Peter Stegnar. Razvoj spletnih aplikacij z vzorci MVC na strežniku in odjemalcu. Diplomska naloga, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2013.

[11] Steve Faulkner, Arron Eicholz, Travis Leithead, Alex Danilo, and Sangwhan Moon. HTML 5.2. Dosegljivo: https://www.w3.org/TR/html52/, 2017. [Dostopano: 27. 8. 2018].

[12] CSS. Dosegljivo: https://www.w3.org/standards/webdesign/htmlcss.html, 2018. [Dostopano: 27. 8. 2018].

[13] Mark Otto and Jacob Thornton. Bootstrap. Dosegljivo: https://getbootstrap.com/, 2011. [Dostopano: 25. 8. 2018].

[14] John Resig. jQuery. Dosegljivo: https://jquery.com/, 2006. [Dostopano: 25. 8. 2018].

[15] Dmitry Baranovskiy. Raphaël.js. Dosegljivo: http://dmitrybaranovskiy.github.io/raphael/, 2008. [Dostopano: 25. 8. 2018].

[16] Jaime Pillora. Notify.js. Dosegljivo: https://notifyjs.com/, 2013. [Dostopano: 25. 8. 2018].

[17] Elbert Alias. Raphaël.export. Dosegljivo: https://github.com/AliasIO/Raphael.Export, 2011. [Dostopano: 1. 9. 2018].

[18] Juan S. Escobar. Raphaël pan-zoom plugin. Dosegljivo: https://github.com/escobar5/raphael-pan-zoom, 2011. [Dostopano: 1. 9. 2018].

[19] Elbert Alias. Raphaël.json. Dosegljivo: https://github.com/AliasIO/Raphael.JSON, 2011. [Dostopano: 1. 9. 2018].