Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matevž Peterec

Razvoj spletne aplikacije e-Asistent z uporabo spletnega ogrodja Yii

diplomsko delo
na univerzitetnem študiju računalništva in informatike

Mentor: izr. prof. dr. Viljan Mahnič
Somentor: izr. prof. dr. Samo Ribarič

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01888/2013
Datum: 07.01.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATEVŽ PETEREC**

Naslov: **RAZVOJ SPLETNE APLIKACIJE E-ASISTENT S POMOČJO SPLETNEGA OGRODJA YII**

DEVELOPMENT OF THE WEB APPLICATION E-ASISTENT USING THE WEB FRAMEWORK YII

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Proučite ogrodje za razvoj spletnih aplikacij Yii in ga uporabite za izdelavo aplikacije "e-Asistent" za pomoč pri pripravi in izvedbi izpitov na Medicinski fakulteti. Aplikacija naj omogoča vzdrževanje izpitnih vprašanj, prijavo študentov na izpit, kreiranje in izpis izpitnih pol, avtomatsko ocenjevanje in statistično obdelavo rezultatov. V nalogi predstavite postopek razvoja in zasnovo sistema v skladu s konceptom MVC (Model-View-Controller). Podrobno opišite primere uporabe, strukturo podatkovne baze in realizacijo poslovne logike.

Mentor:

izr. prof. dr. Viljan Mahnič

Dekan:

prof. dr. Nikolaj Zimic

Somentor:

izr. prof. dr. Samo Ribarič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matevž Peterec, vpisna številka **63050077**, sem avtor diplomskega dela z naslovom:

Razvoj spletne aplikacije e-Asistent z uporabo spletnega ogrodja Yii

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Viljana Mahniča in somentorstvom izr. prof. dr. Sama Ribariča,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, 15. aprila 2013 Podpis avtorja:

Zahvalujem se mentorju izr. prof. dr. Viljanu Mahniču za pomoč in nasvete pri izdelavi diplomskega dela, za izkazano pripravljenost in potrpežljivost pri pregledovanju mojega dela.

Zahvaljujem se tudi somentorju izr. prof. dr. Samu Ribariču za pomoč pri izdelavi praktičnega dela diplome in za čas, ki ga je namenil za strokovne posvete o delovanju sistema.

Posebna zahvala gre tudi moji družini, ki me je med študijem moralno in finančno podpirala. Skupaj z mojo partnerico Majo so mi nudili vso potrebno podporo in verjeli, da bom uspešno dokončal študij.

Svojim najbližjim.

Kazalo

Povzetek

V diplomskem delu je predstavljen razvoj spletne aplikacije e-Asistent, ki nudi nadzorovan proces priprave in poprave izpita. Aplikacija omogoča sestavljanje izpitnih vprašanj, prijavljanje kandidatov na izpit in tiskanje izpitov. Izpitna vprašanja se rešuje z označevanjem pravilnih odgovorov. Z optičnim branjem se rešene izpitne pole v obliki slik vnese v spletno aplikacijo, kjer se jih avtomatsko popravi in oceni. Omogočeno je tudi beleženje rezultatov in izvajanje statističnih izračunov. Aplikacija, ki je bila razvita v sodelovanju z Medicinsko fakulteto v Ljubljani, časovno in organizacijsko optimizira proces priprave in poprave izpitov ter uspešno nadomešča starejši sistem. Potek razvoja dela je opisan po metodologiji slapovnega modela razvoja programske opreme. V delu so predstavljena tudi orodja, ki so pripomogla k hitrejši in učinkovitejši realizaciji aplikacije; poudarek je predvsem na odprtokodnem spletnem ogrodju Yii.

Ključne besede:

Yii, spletno ogrodje, slapovni model, življenjski model razvoja informacijskih sistemov, primeri uporabe, PHP, aplet Java

Abstract

This thesis presents the development of web application e-Asistent, which provides a controlled process of preparation and correction of an exam. It allows you to prepare exam questions, to register candidates for the exam and to print exams. The questions of the exam are completed by marking the right answer. Completed examination papers are scanned as images, which are imported to the web application, where they are marked and rated. It also records the results and performs statistical calculations. The web application, that was developed in cooperation with the Faculty of Medicine in Ljubljana, optimizes time and organization of the examination process and also successfully replaces their old system. The development of the application is shown as a waterfall software development process. The thesis also presents tools, which have contributed to more efficient and faster implementation of the application, with an emphasis primarily on a web framework called Yii.

Key words:

Yii, web framework, waterfall process, software development process, use cases, PHP, Java applet

Seznam uporabljenih kratic

- AJAX – Asynchronous JavaScript and XML. Predstavlja skupino medsebojno povezanih spletnih razvojnih tehnik, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij.
- API – Application programming interface. Programski vmesnik.
- AR – Active Record. Razred, napisan v jeziku PHP, ki predstavlja preslikavo tabele podatkovne baze. Njeni atributi so predstavljeni kot atributi tega razreda. Instanca AR predstavlja vrstico tabele v podatkovni bazi.
- AWT – Abstract Window Toolkit. Knjižnica za gradnjo grafičnih vmesnikov v Javi.
- BMP – Bitmap Image File. Format slike v rastrski grafiki.
- CMS – Content Management System. Sistem za upravljanje s spletnimi vsebinami.
- CRUD – Create, Read, Update, Delete. Najosnovnejše akcije na podatkih. Ustvari, beri, uredi in izbriši.
- CSV – Comma-separated values. Format tekstovne datoteke, pri katerem so vrednosti ločene z vejico.
- EER – Enhanced entity-relationship. Razširjeni konceptualni model, uporaben za oblikovanje podatkovne baze. Predstavljen je z entitetami in relacijami med njimi.

- GET – tip zahteve protokola HTTP. Podatki so prenašajo prek URL.
- HTML, HTML5 – HyperText Markup Language. Jezik za označevanje in oblikovanje dokumentov.
- HTTP – HyperText Transfer Protocol. Protokol za prenašanje hiperteksta.
- JAR – Java Archive. Arhiv, ki predstavlja javansko knjižnico ali aplikacijo.
- JDBC – Java Database Connectivity. Javanski gonilnik za povezavo na podatkovno bazo.
- JSON – JavaScript Object Notation. Preprost format za izmenjavo podatkov. Temelji na skriptnem jeziku JavaScript.
- MVC – Model-View-Controller. Arhitektura "model, pogled in kontroler". Pogosto uporabljena v spletnih ogrodjih.
- ORM – Object-relational mapping. Programska tehnika za preslikavo strukture podatkovne baze v objektno usmerjeni način.
- PDF – Portable Document Format. Odprt standard za izmenjavo elektronskih dokumentov, namenjen predvsem od platforme neodvisnemu prikazovanju dokumentov.
- PNG – Portable Network Graphics. Format slik v rastrski grafiki.
- POST – tip zahteve protokola HTTP. Podatki se prenašajo v telesu zahteve.
- RUP – Rational Unified Process. Metodologija za razvoj programske opreme
- TXT – Text File. Tekstovna datoteka.
- URL – Uniform Resource Locator. Enolični naslov spletnega vira.
- XML – Extensible Markup Language. Razširljivi označevalni jezik za opis strukturiranih podatkov.

Poglavje 1

Uvod

V današnjem času postajajo spletne aplikacije z vidika uporabnikov svetovnega spleta vedno bolj pomembne. Opazimo lahko, da počasi izpodrivajo namizne aplikacije, kar je po svoje razumljivo: uporabnikom spletne aplikacije se ni treba ukvarjati z namestitvijo programa na računalnik, spletna aplikacija je neodvisna od platforme, podatki so shranjeni na strežniku, če uporabljamo spletno aplikacijo v brskalnikovem peskovniku, je odjemalčev računalnik varnejši pred virusi, poleg tega za uporabo spletne aplikacije potrebujemo le dostop do interneta. Slabosti spletnih programov pa so slabša interakcija med uporabnikom in aplikacijo, možnost uhajanja pomembnih podatkov s centralnega strežnika, nujen je internetni dostop itd. Za razvoj spletnih aplikacij imamo na voljo veliko orodij, ki nam delo olajšajo in pohitrijo. V zadnjem času se uporabljajo predvsem aplikacije, znane pod kratico CMS (*Content Management System*) in spletna ogrodja (angl. *Web Framework*). Za razvoj preprostejših spletnih aplikacij, ki vsebujejo večino standardnih funkcij (kot sta branje in zapisovanje v podatkovno bazo), so aplikacije CMS zadostne. Za razvoj bogatih aplikacij pa je primernejše spletno ogrodje. Na tržišču je veliko spletnih ogrodij, ki se razlikujejo tako v programskih jezikih kot tudi v sami arhitekturi.

Uporaba namiznih in spletnih aplikacij se širi tudi v vzgojno-izobraževalnih organizacijah. Ena izmed teh je tudi Medicinska fakulteta v Ljubljani, ki upo-

rablja sklop namiznih aplikacij za ustvarjanje in hranjenje izpitnih vprašanj, pripravo in popravo izpitov, razpored sedežnega reda, razporede ustnih izpitov itd. Programi sklopa e-Asistent so razbremenili delo asistentov in profesorjev, predvsem v času izpitnega obdobja. Ni si težko predstavljati, koliko dela je potrebnega ob prijavi dvesto študentov na posamezni izpitni rok in popravi dvesto izpitov "na roke". Spremenjeni način reševanja izpitov z označevanjem pravilnih odgovorov je veliko doprinesel, med drugim tudi možnost elektronskega popravljanja izpitnih pol. Zaradi programov e-Asistenta so izpiti popravljeni veliko hitreje (študenti izvejo rezultate že v eni uri po pisanju izpita). Žal pa ima ta sistem tudi pomankljivosti, predvsem v hranjenju podatkov (samostojna datoteka s specifičnim načinom kodiranja), ne nudi centraliziranega hranjenja podatkov (vsak uporabnik ima podatke na svojem računalniku), zato morajo uporabniki sami skrbeti za njihovo organizacijo. Posledično se statistika reševanja vprašanj in izpitov na podlagi neurejenih podatkov izračuna napačno. Težave je sistemu povzročala tudi sama struktura Medicinske fakultete. Fakulteto sestavljajo različni inštituti in kadri. Pri pripravi izpitov oz. izpitnih vprašanj se morajo te enote uskladiti, kajti pri nekem študijskem predmetu lahko sodeluje več enot hkrati. Posledično je prišlo do velike zmede pri izmenjavanju podatkov za sestavo skupnega izpita. Pojavila se je potreba po skupnem, centraliziranem informacijskem sistemu, ki uporabnikom omogoča boljšo organiziranost, transparetno uporabo podatkov in verodostojnejše statistične izračune. Tako sem se lotil razvoja nove spletne aplikacije e-Asistent, ki simulira delovanje starega sistema in odpravi njegove pomanjkljivosti. Hkrati razširi organizacijske funkcionalnosti z vpeljavo uporabniških vlog in njihovih pravic ter nadzorovanega dostopa do sistema, izboljša pa tudi nadzor nad procesom priprave in poprave izpita.

Informacijski sistem sem poskušal razvijati po metodologiji slapovnega življenjskega modela, vendar se je v praksi to izkazalo za težje od pričakovanega.

V diplomskem delu bom predstavil razvoj spletne aplikacije e-Asistent (ki bi v prihodnosti lahko nadomestila stari sistem) po slapovnem modelu.

Delo je sestavljeno iz treh delov in zaključka. V Poglavju 2 bom predstavil problem starega sistema, možne rešitve in splošno funkcionalnost novega sistema. V Poglavju 3 bom opisal orodja, ki sem jih uporabil za realizacijo novega sistema. Opisal bom spletno ogrodje, ki mi je omogočilo hitrejšo in enostavnejšo implementacijo. V Poglavju 4 bom opisal življenjski model razvoja informacijskega sistema in predstavil faze razvoja spletne aplikacije e-Asistent po metodologiji slapovnega modela. V zaključku bom predstavil rezultat razvoja po izbranem modelu, nadaljnji razvoj sistema, možne izboljšave in izkušnje, ki sem jih med delom pridobil.

Poglavje 2

Predstavitev problema

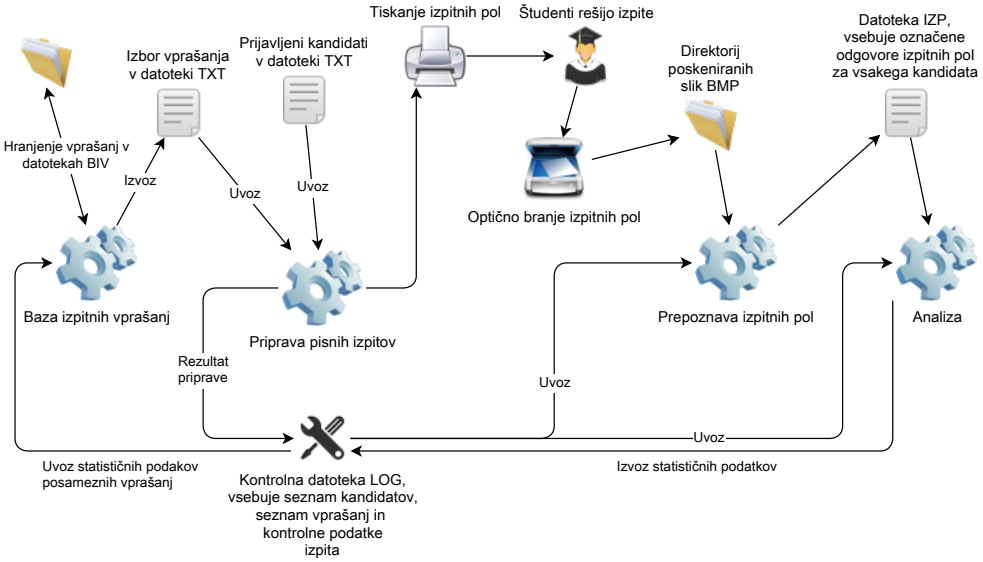
2.1 Opis problema, s katerim se je soočila Medicinska fakulteta

Vsako leto se veliko študentov vpiše na Medicinsko fakulteto v Ljubljani, zato se fakulteta v izpitnem obdobju sooča z velikim številom študentov, ki hkrati opravljajo pisne izpite in kolokvije. Obseg časa in dela, potrebnega za popravo vseh izpitov, je tako postal prevelik. K reševanju tega problema lahko pripomoreta reševanje izpitov z označevanjem pravilnih odgovorov in možnost elektronskega popravljanja izpitnih pol. Zato je nastal sistem e-Asistent, tj. sklop namiznih aplikacij oz. programov, ki je olajšal delo priprave in poprave izpitov, vendar še vedno ni predstavljal optimalne rešitve.

2.2 Stari sistem e-Asistent in njegove pomanjkljivosti

Medicinska fakulteta že nekaj let uporablja sklop programov e-Asistent. Vsak program predstavlja določeno funkcionalnost, kot so:

- priprava izpitov in izpitnih pol,



Slika 2.1: Prikaz procesa priprave in poprave izpitov v starem sistemu

- poprava izpitnih pol,
- urejanje izpitnih vprašanj in njihovo hranjenje v lastnem datotečnem formatu,
- analiza rešenih izpitnih pol.

Na sliki 2.1 je predstavljen proces priprave in poprave izpita z uporabo programov. Opazimo, da programi podatkovno niso neposredno povezani. Problem je, da morajo uporabniki skrbeti za organizacijo datotek vprašanj, datotek prijavljenih kandidatov in celo kontrolnih datotek izpitov. Podatki niso centralizirani, kar pomeni, da z neveščo uporabo teh programov in njihovih datotek lahko hitro pride do velike zmede. Za verodostojne statistične izračune uspešnosti reševanja izpitov, vprašanj in posameznih kandidatov je potrebno zgodovino nekje hraniti, kar pa stari sistem ne omogoča. Program Analiza pokaže le statistično obdelavo podatkov na trenutno popravljenem izpitu, ne pa tudi statistične obdelave skozi zgodovino rešenih izpitov. Statistika se beleži le za posamezna vprašanja, ko preko tekstovne datoteke “log” vnesemo statistiko vprašanj v program Baza izpitnih vprašanj.

Brez uporabe centralne baze je tudi težje izvedljivo sestavljanje izpitnih vprašanj za določene izpite, ki zahteva sodelovanje skupine ljudi iz različnih inštitutov. Sistem ne nudi nadzora dostopa do funkcionalnosti glede na akademske vloge (nima implementiranega kontrolnega sistema upoštevanja vlog in pripadajočih pravic) in ne nadzira poteka izpitnega procesa (vstavi vprašanja v izpit, prijavi kandidate, natisni izpite, popravi izpite), tako da uporabnik ne ve, v katerem stanju procesa se je ustavil.

2.3 Sistem Moodle kot možna rešitev

Sistem Moodle [3] je spletna aplikacija, ki nudi skoraj vse pedagoške dejavnosti, ki se pojavljajo v vzgojno-izobraževalnih ustanovah. Predstavlja spletno učilnico, do katere dostopajo tako učenci kot učitelji in kamor učitelji naložijo različne učne vire, naloge, kvize itd. Učenci imajo do teh virov nadzorovan dostop. Sistem Moodle bi lahko nadomestil sklop programov e-Asistent, če bi opustili fizično reševanje izpitov. Vsak študent bi preprosto dostopal do določenega predmeta v spletni učilnici, kjer bi se izvajalo reševanje izpitov v obliki kvizov (6. poglavje v [3]). Vendar zaradi nekaterih pomanjkljivosti ta rešitev ni primerna. Reševanje izpitov je namreč na Medicinski fakulteti strogo nadzorovano, saj gre za študente, ki bodo postali bodoči zdravniki, kar zahteva veliko znanja na pamet, odločanja v trenutku itd. Bilo bi torej neprimerno, da bi študentje medicine izpite reševali doma. Glede na to, da se na izpit lahko prijavi več kot dvesto študentov, je težko izvesti elektronsko reševanje izpitov po računalniških učilnicah, saj pride do problema uskladitve časa izpitov za posamezne skupine, zaseda se velik del prostora, posledično je potrebnih več računalnikov pa tudi implementacija zaščite pred dostopom do prepovedane literature na spletu.

2.4 Nova spletna aplikacija e-Asistent in njena funkcionalnost

Druga možna rešitev je razvoj spletne aplikacije, ki združi vse funkcionalnosti sklopa programov e-Asistent. Aplikacija bi bila nameščena na enem od strežnikov Medicinske fakultete, do katerega bi imeli dostop vsi profesorji in asistenti na fakulteti. Uporabljala bi se centralna baza, v katero bi se shranjevali vsi rezultati izpitov, procesi priprave in poprave izpitov, vsa izpitna vprašanja, kandidati in uporabniki sistema.

Sistem bi nudil nadzorovan vstop v sistem prek uporabniškega imena in gesla. Upošteval bi različne vloge uporabnikov, kot so profesor, asistent in administrator; vsaka vloga bi imela svoje pravice. Po navadi profesor pooblasti svojega pomočnika ali asistenta za pripravo ali popravo izpitov, tako da bi bilo treba omogočiti tudi izmenjavo dovoljenj med uporabniki. Novi sistem bi moral vsebovati funkcionalnosti vseh štirih programov starega sistema. Prvi vmesnik bi nudil urejanje vprašanj, preko katerega bi uporabniki ustvarjali, urejali in brisali vprašanja iz baze. Drugi vmesnik bi nudil urejanje izpitov. Uporabnik bi lahko ustvarjal, urejal in brisal izpite. Vsebovati bi moral tudi potek celotnega procesa priprave in poprave izpita. Poprava izpita bi morala omogočati vnos slik izpitnih pol v sistem, zaznavo in popravo pol. Tretji vmesnik bi predstavljal prikaz statističnih podatkov tako za uporabnika, vprašanje, izpit kot tudi za posameznega študenta. Vnos kandidatov v sistem bi potekal preko datotek ali ročnega vpisovanja v sistem preko vmesnika. Administrator bi moral skrbeti za urejanje uporabnikov v sistemu in za določene šifrante, kot so študijski programi, letniki in smeri študija ter možne teme vprašanj.

Poglavje 3

Orodja, uporabljena za razvoj spletnega sistema e-Asistent

3.1 Spletna ogrodja

Živimo v času, ko si spletnega sveta ne moremo predstavljati brez spletnih aplikacij. Želja po hitrejši in enostavnejši gradnji še vedno raste in z njo tudi orodja, ki nam poskušajo olajšati realizacijo še tako zapletenih in obsežnih aplikacij. Eden od sklopov teh orodji se imenuje spletno ogrodje (angl. *Web Framework*). Spletno ogrodje je skupek programske kode, organizirane v urejeno strukturo, namenjene razvijanju spletnih aplikacij. Lahko si ga predstavljamo kot že napol napisan program, ki ga je treba le še dokončati. Vendar ta prednost terja tudi svoj davek. Prvi del programa je naredil nekdo drug, zato nimamo nobenega nadzora nad tem, kako in ali je varno narejen, ali bo ustrezal naši gradnji aplikacije itd.

Nekatera spletna ogrodja imajo zelo specifično strukturo, omejitve in potrebe ter zahtevajo veliko konfiguracij, še preden sploh začnemo resno razvijati našo aplikacijo. To zahteva daljše učenje pravil okolja, v katerem delamo, vendar pozneje omogoča hitrejše razvijanje aplikacij. Na drugi strani pa so ogrodja, ki se želijo dolgotrajnemu učenju izogniti. So preprostejša, prilagodljivejša, razvijalcem pa je dovoljeno posegati v njihova jedra. Taka ogrodja

so bolj podobna knjižnicam, ki vsebujejo določen nabor funkcij, s katerimi uokvirijo naš potek dela.

Uporaba spletnih ogrodij ni vedno dobrodošla. V naslednjih poglavjih predstavljam prednosti in slabosti njihove uporabe. Podrobnosti o spletnih ogrodjih, slabostih in prednostih njihove uporabe si lahko pogledate v [6].

3.1.1 Prednosti

Kdaj je dobro uporabiti spletno ogrodje.

- Če razvijamo spletno aplikacijo z dinamično vsebino, kot so e-trgovine, socialna omrežja, center novic, e-knjižnice itd.
- Predvsem za razvoj aplikacij, ki so lahko na začetku uporabe majhne, sčasoma pa postanejo svetovno uporabljane in pri tem ne zahtevajo večjih sprememb v kodi.
- Za izdelavo aplikacij, ki vsebujejo module in komponente, koristne za razvoje drugih aplikacij.
- Za projekte, ki morajo biti dokončani v določenem roku. Kjer imamo osebje, ki si stalno izmenjuje delo, in stranke, ki morajo biti obveščene o delu na projektu.

Uporaba spletnega ogrodja je smiselna predvsem takrat, ko razvijamo aplikacijo, ki se mora povezovati na podatkovno bazo, izmenjavati podatke in veliko komunicirati z uporabnikom (uporabniki lahko dodajajo svoje vsebine in jih spreminjajo, brišejo). Te funkcionalnosti predstavljajo standardno prakso v spletnih ogrodjih.

3.1.2 Slabosti

Kdaj ni dobro uporabiti spletnega ogrodja oz. je njegova uporaba stvar osebne odločitve.

- Za spletne strani, kjer uporabniku ni potrebno spreminjati vsebine.

- Za majhne projekte z manjšo podatkovno bazo z omejeno povezavo.
- Za obsežne aplikacije, pri katerih spletno ogrodje ni več kos njihovemu obsegu. Kjer je potrebno imeti optimalen nadzor nad vsem, kar se dogaja.
- Za izredne projekte, ki se razvijajo v povsem novo smer in ne upoštevajo standardnih funkcionalnosti.
- V ekipi, ki se novim pravilom in okolju noče ali zaradi njihove težke razumljivosti ne more podrediti.

3.2 Spletno ogrodje Yii

Eno izmed spletnih ogrodij, ki mi je olajšalo realizacijo spletne aplikacije, je spletno ogrodje Yii. Kratica Yii pomeni "*Yes It Is!*", kar predstavlja odgovor na vprašanja, kot so "*Ali je dovolj hiter?*", "*Ali je profesionalen?*", "*Ali je varen?*"itd. Ogrodje je ustvaril Čjang Šue (orig. *Qiang Xue*). Izdelovati ga je začel v začetku leta 2008 [8]. Čjang je bil vrsto let razvijalec in vzdrževalec spletnega ogrodja PRADO, pri čemer si je pridobil veliko izkušenj z delovanjem spletnih ogrodij. Kritike uporabnikov, da je ogrodje PRADO preveč kompleksno, so ga spodbudile k izdelavi lastnega spletnega ogrodja, ki bi bilo predvsem enostavnejše, hitrejše in prilagodljivejše. V decembru 2008 je bilo ogrodje uradno objavljeno in je poželo veliko pohval zaradi svoje hitrosti glede na ostala spletna ogrodja PHP. Ogrodje je primerno predvsem za večje aplikacije, saj je njegova struktura enostavna in ga lahko hitro prilagodimo svojim potrebam. Glavne prednosti in vzrok za popularnost so:

- enostavnost,
- učinkovitost,
- razširljivost.

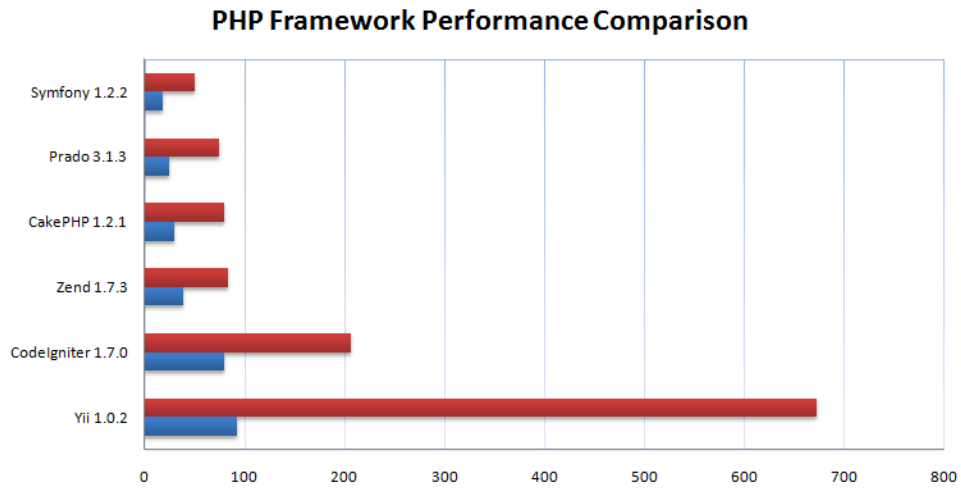
Enostavnost se odraža predvsem v tem, da je za razvijalca dovolj le poznavanje jezika PHP in objektno usmerjenega načina programiranja v njem. Za delovanje aplikacije Yii potrebuje le strežnik, ki podpira jezik PHP različice 5.1.0 ali novejše. Podpira filozofijo "*convention over configuration*", kot je omenjeno v [8]. To pomeni, da bomo v primeru upoštevanja pravil napisali manj kode in porabili manj časa za razvoj aplikacije.

Učinkovitost se kaže v možnosti hitrega razvoja aplikacij, če se le držimo konvencij, saj se le malo kode ponavlja. Yii je oblikovan tako, da se aplikacije razvijajo po principu DRY (*Don't repeat yourself*). Vse aplikacije posledično uporabljajo strukturo MVC (*Model-View-Controller*). Modeli, pogledi in kontrolerji so tako ustrezno ločeni, kar pripomore k manjšemu obsegu kode, poleg dobre organizacije pa ponuja tudi hitro izvajanje.

Vsak sklop kode ogrodja je mogoče razširiti ali ga oblikovati po lastni želji. Vnašanje različnih zunanjih knjižnic v spletno ogrodje in njihova uporaba sta preprosta. Več o ogrodju Yii je opisano v [8].

3.2.1 PHP

Spletno ogrodje Yii je napisano v programskem jeziku PHP, široko uporabljanem skriptnem jeziku, ki je namenjen posebej za razvoj spleta. Zlahka se ga vključi tudi v HTML. Koda je po večini napisana na objektno usmerjeni način, saj tako lažje predstavi razširljivost in modularnost ogrodja. Za razvoj aplikacije v ogrodju je posledično potrebno poznati programski jezik PHP in objektno usmerjeni način programiranja. Struktura MVC, ki jo Yii zahteva, razvijalca prisili, da sledi določenim konvencijam. Kontrolerji morajo biti deklarirani kot podrazredi, ki so izpeljani iz razreda *Controller*, prav tako modeli, ki morajo biti izpeljani iz *CActiveRecord* ali *CActiveForm* (starš je *CFormModel*). Omenjeni starševski razredi pa so izpeljani iz osnovnih razredov *CController* in *CModel*. Pogledi so prav tako datoteke PHP, ki večinoma vsebujejo kodo HTML. Za dinamične sklope se uporabljajo funkcije PHP, ki izpišejo vsebino, poslano s strani kontrolerja.



	Yii 1.0.2	CodeIgniter 1.7.0	Zend 1.7.3	CakePHP 1.2.1	Prado 3.1.3	Symfony 1.2.2
RPS with APC	673	206	83	79	75	50
RPS without APC	93	79	38	29	24	18

Slika 3.1: Spletno ogrodje Yii v primerjavi z drugimi popularnimi spletnimi ogrodji. Na osi x so vrednosti RPS (*Request Per Second*), ki označujejo število zahtev na sekundo, ki jih ogrodje lahko izvede (več je boljše). Izvedena je primerjava vrednosti brez in z uporabo APC (*Alternative PHP Cache*). APC predstavlja način optimizacije kode PHP z uporabo predpomnenja. Na osi y so predstavljena različna spletna ogrodja. Podrobnejši opis testiranja je dostopen na naslovu *http://www.yiiframework.com/performance/*.

3.2.2 Struktura MVC

Yii spada med spletna ogrodja, ki upoštevajo strukturo MVC (Model-View-Controller). Za vsako kodo modela, pogleda in kontrolerja je določena datotečna struktura. Aplikacije z arhitekturo MVC tipično obdelajo spletno zahtevo na sledeč način (povzeto po [8]):

- brskalnik pošlje zahtevo strežniku, ki gosti aplikacijo MVC,
- odzove se kontroler, ki zahtevo obdela,
- kontroler interaktira z modelom,
- kontroler pokliče pogled,
- pogled predstavi podatke (v kodi HTML) in jih vrne brskalniku za prikaz.

3.2.3 Model

Model je odgovoren za vzdrževanje stanj na ravni podatkov. Predstavljen je z objektom razreda *CModel* oz. z objektom otroka tega razreda. Tipično vsebuje atribute, ki so poimenovani z oznakami (oznake služijo predvsem pri prikazu atributov v pogledih). Validacija je določena s pravili za vsak atribut. Atribut predstavlja polje tabele podatkovne baze ali pa polje nekega vnosnega polja formularja. Obstajata dve vrsti modelov: prvi izhaja iz razreda *CFormModel* in drugi iz razreda *CActiveRecord*. Prvi je namenjen shranjevanju v pomnilnik prek izpolnjenega formularja, drugi pa hranjenju podatkov v podatkovni bazi.

3.2.4 Pogled

Pogled je odgovoren za predstavitev podatkov, ki jih dobimo posredno, preko modelov, uporabnikom. Predstavljen je kot skripta PHP, sestavljena večinoma iz kode HTML in stavkov PHP za sklope dinamičnih vsebin (zanke in pogojni

stavki). Yii ponuja tudi tako imenovane pomočnike HTML (angl. *HTML helper*). To so razredi z določenimi funkcijami, ki olajšajo delo tako, da vračajo izpise v kodi HTML (predvsem elemente formularjev) .

3.2.5 Kontroler

Kontroler sprejema spletne zahteve in jih ustrezno obdela. Sprejema uporabniške akcije, interaktira z ustreznimi modeli in skrbi, da pogled prikaže ustrezno vsebino. Njegove funkcije se imenujejo akcije, ker jih večinoma sprožijo uporabniki preko elementov vmesnika. Funkcije v kontrolerju se morajo nujno začeti z besedo *action*, potem pa sledi ime akcije (identiteta akcije). Način proženja njegovih akcij je odvisen od strukture URL, ki je poslana aplikaciji na strežniku.

```
http://hostname/index.php?r=ControllerID/ActionID
```

Če uporabljamo preslikovalnik naslovov URL, ki ga nastavi Yii, potem so tudi naslovi veliko bolj prijazni uporabnikovim očem (opomba: če akcija ni podana, se privzeto kliče akcijo *index* določenega kontrolerja).

```
http://hostname/ControllerID/ActionID
```

Če uporabnik kliče akcijo z določenim parametrom. Parametri se pošiljajo preko metode GET.

```
http://hostname/ControllerID/ActionID/ParemeterID
```

Primer naslova URL, ko se kliče kontroler *Vprašanja*, akcijo *Uredi* s parametrom 2. Izvrši se urejanje vprašanja z identiteto 2 v moji aplikaciji e-Asistent.

```
http://e-Asistent-web/vprašanja/uredi/2
```

3.2.6 ORM in AR

V spletnih aplikacijah je uporaba relacijskih baz zelo popularna. S pomočjo ORM (*Object-Relational mapping*) je relacijska baza predstavljena kot sklop

```
/*primer vstavljanja vrstic v podatkovno bazo*/
$student=new Student;
$student->firstname="Janez";
$student->lastname="Novak";
$student->...
$student->save();

/*primer branja iz podatkovne baze*/
$student=Student::model()->findByPk(2); //dobimo studenta s id=2
$studenti=Student::model()->find("firstname=:firstname",
                                 array(':firstname'=>"Janez"));
```

Slika 3.2: Primer uporabe AR. Kreiramo novo instanco modela *Student*. Ob izvršitvi funkcije *save* se shrani nova vrstica v tabelo *Student* v podatkovni bazi. Pri branju podatkovne baze uporabljamo funkcije *find*.

objektov. Yii vsebuje abstraktni nivo ORM nad podatkovno bazo v obliki uporabe AR (*Active Record*). Active Record predstavlja abstraktni dostop do podatkovne baze na objektno usmerjeni način. Njegove instance oz. objekti AR so vrstice tabele podatkovne baze. Stolpci vrstice so pa atributi objekta AR. Bolj podroben opis AR je v [8], na sliki 3.2 pa je primer uporabe.

3.2.7 Struktura map in pomembnih datotek projekta

V spletnem ogrodju Yii se z zagonom skripte *yiic.php* vzpostavi drevo map in datotek, ki predstavlja projekt. Skripta je vsebovana v spletnem ogrodju Yii, ki je navadno nameščeno na lokaciji, ki ni dostopna spletu. Koda projekta in jedro spletnega ogrodja sta torej fizično ločena. Vse pomembne datoteke so v mapi *protected*, ki je tudi ustrezno zaščitena (dovoljeno je samo branje). V njej imamo ločene mape, kot so *Controllers*, *Models* in *Views*, saj ogrodje podpira strukturo MVC. Poleg tega vsebuje še mapo *extensions* (v kateri imamo dodatne zunanje knjižnice), *components* (vsebuje pomembne gradnike spletne aplikacije) in *data* (vsebuje sheme podatkovne baze). V mapi *config* je zelo pomembna konfiguracijska datoteka *main.php*, v kateri se nastavi,

kateri sistem za upravljanje podatkovne baze se uporablja, splošni naslov spletne aplikacije, uporaba preslikovalnika URL naslovov, katere komponente se uporabljajo itd. Zagon spletne aplikacije se izvrši s skripto *index.php*. Skripta uvozi jedro spletnega ogrodja in izvrši se dejanski zagon aplikacije ob klicu ukaza *Yii::createWebApplication($config)->run()*. Na sliki 3.3 je prikazano drevo map in datotek projekta Yii.

3.2.8 Vmesnik Gii

Od različice 1.1.2 dalje Yii ponuja svoj implementirani generator kode, imenovan Gii. Vmesnik omogoča enostavno in hitro generiranje kode za kontrolerje, modele, formularje, akcije CRUD in module. Tako se razvijalci lažje držijo vnaprej določenih pravil in konvencij pri sestavljanju ustreznih komponent aplikacije znotraj spletnega ogrodja. Slika 3.4 prikazuje uporabo vmesnika Gii.

3.3 MySQL

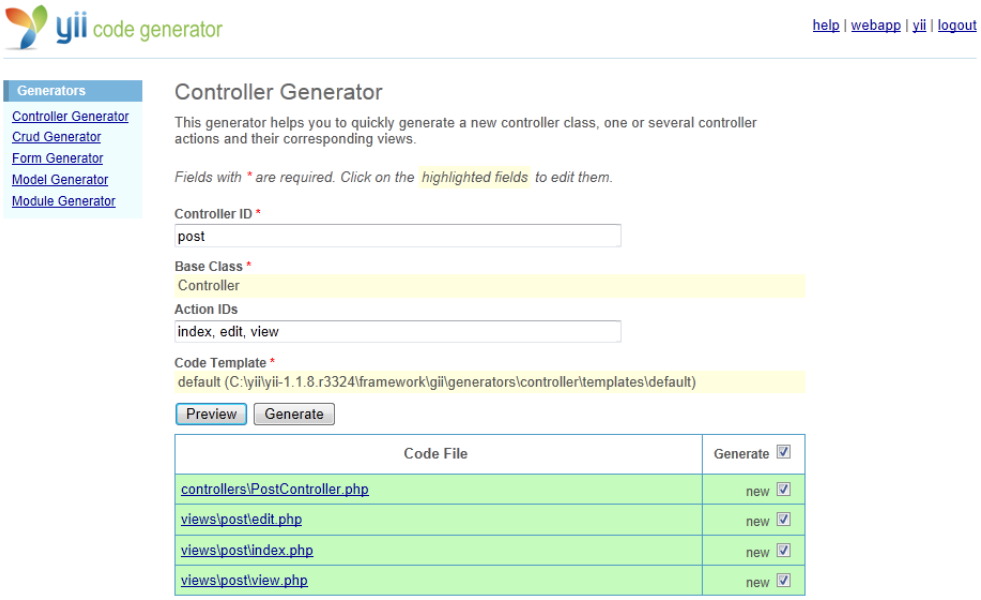
MySQL je odprtokodni sistem za upravljanje relacijske podatkovne baze, ki uporablja jezik SQL. Zaradi svoje popularnosti je na voljo veliko brezplačnih programskih vmesnikov za dostop do baze. Ne manjka niti orodij za načrtovanje in ravnanje s to podatkovno bazo. Za načrtovanje sem uporabljal MySQL Workbench, ki ponuja načrtovanje in oblikovanje baze prek sestavljanja diagramov EER, administracijo serverja, uporabnikov itd. Orodje phpMyAdmin sem uporabljal za administracijo kreirane baze in njenih tabel pa tudi za samo testiranje.

3.4 Aplet Java

Za kreiranje vmesnika, ki hkrati obdela poskenirane slike izpitnih pol, zazna označbe in posledično oceni izpite, sem uporabil programski jezik Java. Aplet je "javanski programček", ki po navadi teče znotraj drugega programa,

```
index.php                Web application entry script file
index-test.php           entry script file for the functional tests
assets/                  containing published resource files
css/                     containing CSS files
images/                  containing image files
themes/                  containing application themes
protected/               containing protected application files
   yiic                  yiic command line script for Unix/Linux
   yiic.bat              yiic command line script for Windows
   yiic.php              yiic command line PHP script
   commands/             containing customized 'yiic' commands
      shell/             containing customized 'yiic shell' commands
   components/           containing reusable user components
      Controller.php     the base class for all controller classes
      UserIdentity.php   the 'UserIdentity' class used for authentication
   config/               containing configuration files
      console.php        the console application configuration
      main.php           the Web application configuration
      test.php           the configuration for the functional tests
   controllers/          containing controller class files
      SiteController.php the default controller class
   data/                 containing the sample database
      schema.mysql.sql   the DB schema for the sample MySQL database
      schema.sqlite.sql  the DB schema for the sample SQLite database
      testdrive.db       the sample SQLite database file
   extensions/           containing third-party extensions
   messages/             containing translated messages
   models/               containing model class files
      LoginForm.php      the form model for 'login' action
      ContactForm.php    the form model for 'contact' action
   runtime/              containing temporarily generated files
   tests/                containing test scripts
   views/                containing controller view and layout files
      layouts/           containing layout view files
         main.php        the base layout shared by all pages
         column1.php     the layout for pages using a single column
         column2.php     the layout for pages using two columns
      site/              containing view files for the 'site' controller
         pages/          containing "static" pages
            about.php    the view for the "about" page
         contact.php     the view for 'contact' action
         error.php       the view for 'error' action (displaying external errors)
         index.php       the view for 'index' action
         login.php       the view for 'login' action
```

Slika 3.3: Drevo map in datotek projekta Yii



Slika 3.4: Vmesnik za generiranje kode Gii. Primer generiranja kode za kontroler *PostController.php*.

največkrat znotraj javansko osveščenih brskalnikov (uporabnik mora imeti v brskalniku nameščen vtičnik za Javo). Vmesnik uporablja knjižnice Swing za kreiranje grafičnih elementov vmesnika in AWT za izrisovanje slik in upravljanje z grafiko. Več o načinu delovanja omenjenih knjižnic si lahko ogledate v [4, 7]. Aplet je zapakiran v izvršno datoteko formata JAR in integriran v spletno aplikacijo.

Poglavje 4

Potek razvoja spletne aplikacije e-Asistent-web

Življenjski cikel razvoja programske opreme (angl. *Software Development Life Cycle Model*), kot že samo ime pove, predstavlja razvoj sistema, razdeljen na več faz. Modeli življenjskih ciklov razvoja odgovorjajo na vprašanje, kako si faze sledijo in kaj predstavljajo. Življenjski cikel razvoja je po navadi razdeljen v naslednje faze:

- analiza,
- načrtovanje,
- implementacija,
- testiranje in uvedba.

Obstajajo različni modeli življenjskih ciklov razvoja glede na način uporabe omenjenih faz. Imamo slapovne, iterativne, prototipne, spiralne in inkrementalne modele razvoja. Podrobnejši opis različnih modelov si lahko ogledate tudi v [1, 9].

4.1 Razvoj aplikacije v okviru tipov modelov

Za razvoj spletne aplikacije sem v osnovi uporabljal slapovni model razvoja, vendar je to v praksi težko izpeljati. Za uspešno izvedbo projekta sem moral uporabiti kombinacijo več modelov. Predvsem sem se približal iterativnem načinu razvoja – ko sem se vračal po fazah, da sem sistemu dodal še katero funkcionalnost. Prototipnemu načinu razvoja pa sem se približal, ko sem moral izboljšati vmesnik za zaznavo izpitnih pol, pri katerem je program "Poprava izpitnih pol" starega sistema služil kot prototip.

4.1.1 Slapovni model

Pri slapovnem modelu razvoja projekt razbijemo na omenjene faze razvoja, ki jim sledimo po vrsti. Na splošno ta metodologija razvoja ne dopušča vračanja po fazah. Na naslednjo fazo lahko preskočimo le takrat, ko je prejšnja končana (zato se imenuje tudi zaporedni življenjski model). Vendar je v praksi to skoraj nemogoče izpeljati, saj projekta nikoli ne vidimo v celoti končanega, še preden se ga lotimo razvijati.

4.1.2 Iterativni model

Slapovni in iterativni model procesa velikokrat razumemo narobe. Za veliko projektov je rečeno, da se razvijajo po iterativnem stilu, vendar se dejansko razvijajo po slapovnem. Glavno razliko med modeloma predstavlja način razbijanja procesa na različne faze. Pri iterativnem modelu razvoja razbijemo projekt na podmnožice funkcionalnosti sistema oz. na iteracije. Prva iteracija določa nek sklop funkcionalnosti sistema, ki ga razvijemo po slapovnem stilu razvoja. Po končani iteraciji je sistem že delno narejen in nadaljujemo z naslednjo iteracijo, kjer se način razvoja ponovi. Iteriramo do zadovoljive končne celote sistema. Razlika je podrobneje opisana v [5].

4.1.3 Prototipni model

Prototipni razvoj modela predstavlja različico iterativnega stila razvoja, le da se prototipi tu razvijajo postopno (iterativno), dokler ne dosežemo zadovoljive kakovosti sistema.

4.2 Razvoj aplikacije v fazi analize

4.2.1 Zajem zahtev

V fazi analize smo z osebjem Medicinske fakultete definirali, kaj bi novi sistem delal in kako bi deloval. Samo razumevanje zahtev ni bilo težavno: zajel naj bi vse zahteve, ki jih je izpolnjeval že stari sistem e-Asistent. Zahteve so naslednje:

- možnost urejanja izpitnih vprašanj (ustvari, uredi, izbriši in beri vprašanje),
- možnost urejanja izpitov (ustvari, uredi, izbriši in beri izpit),
- možnost priprave izpita (vnesi izpitna vprašanja, prijavi in odjavi kandidate na izpit),
- možnost tiskanja izpitov (izpiti in pripadajoče izpitne pole),
- možnost elektronske poprave in ocenjevanja izpitov.

Med pogovorom pa so se pojavile še dodatne zahteve za izboljšanje sistema. Te so:

- prijava v sistem,
- upoštevanje različnih uporabniških vlog in njihovih pravic pri izvajanju funkcij sistema,
- urejanje uporabnikov sistema (ustvari, uredi, izbriši uporabnika),
- možnost urejanja kandidatov oz. študentov (ustvari, uredi, izbriši in beri kandidata),

- možnost urejanje določenih pravic s strani administratorja in profesorja,
- možnost nadzora nad procesom priprave in poprave izpita.

4.2.2 Model primerov uporabe kot rezultat analize

Za lažje razumevanje delovanja sistema sem uporabil model primerov uporabe. Diagram primerov uporabe na sliki 4.1 prikazuje akterje, ki sodelujejo s primeri uporabe. Podrobnejši opis primerov uporabe sem napisal v formatu RUP, tako kot je opisan v [2].

1. **Primer uporabe**

   1.1. Ime primera uporabe: **Dodeljevanje in urejanje pravic med uporabniki**.

      1.1.1. Kratek opis: Primer uporabe administratorju omogoča, da dodeli določene uporabniške pravice drugemu uporabniku ali jih ureja. Gre za pravice urejanja uporabniških izpitnih vprašanj, uporabniških izpitov ter uporabniških izpitnih procesov priprave in poprave.

      1.1.2. Akter: Administrator.

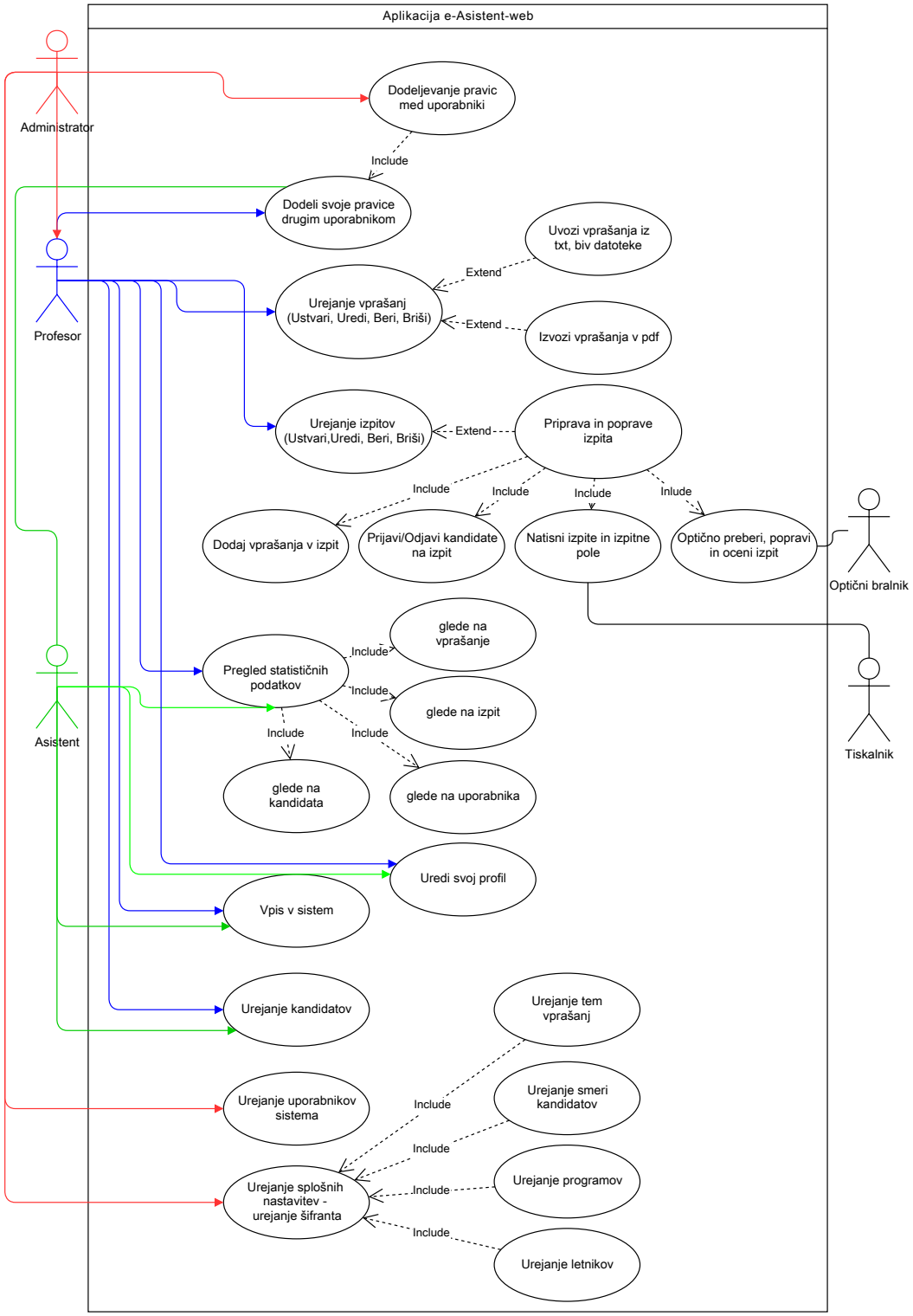
   1.2. Tok dogodkov.

      1.2.1. Administrator lahko izbere možnost *Dodeli pravice drugim uporabnikom*, lahko pa ureja že dodeljene pravice.

      1.2.2. Običajni tok.

         1.2.2.1. Odpre se formular, preko katerega določimo novo dovoljenje.

         1.2.2.2. Izbere uporabnika, ki dovoljuje oparacije s svojimi predmeti (na voljo so le Profesorji, Administrator lahko v imenu drugega profesorja nekaj dovoli).



Slika 4.1: Diagram modela primerov uporabe spletne aplikacije e-Asistent

1.2.2.3. Izbere uporabnika, ki mu bo dovoljeno izvajati akcije na predmetih.

1.2.2.4. Izbere, na kaj se nanaša dovoljenje. To so lahko vprašanja, izpiti in proces priprave in poprave izpita.

1.2.2.5. Označi eno ali več akcij, kot so beri, uredi, izbriši in ustvari. To so akcije, ki bodo dovoljene.

1.2.2.6. Potrdi formular in dovoljenje se shrani v sistem.

1.2.3. Alternativni tok.

1.2.3.1. V tabeli prikazanih dovoljenj izbere določeno dovoljenje za urejanje.

1.2.3.1.1. Odpre se formular, enak kot 1.2.2.1., sledimo dogodkom po 1.2.2.1.

1.2.3.2. V tabeli prikazanih dovoljenj določeno dovoljenje izbrišemo.

1.2.3.2.1. Pojavi se potrditveno okno.

1.2.3.2.2. Uporabnik potrdi.

1.2.3.2.3. Sistem izbriše dovoljenje.

1.2.4. Predpogoj: Administrator mora biti prijavljen v sistem.

2. **Primer uporabe**

2.1. Ime primera uporabe: **Urejanje in dodeljevanje svojih pravic drugim uporabnikom**.

2.1.1. Kratek opis: Primer uporabe omogoča, da Profesor dodeli drugemu Profesorju ali Asistentu pravice nad svojimi izpiti, vprašanji, procesi. Lahko tudi ureja in briše svoja obstoječa dovoljenja.

2.1.2. Akter: Profesor.

2.2. Tok dogodkov.

2.2.1. Profesor izbere možnost *Dodeli svoje pravice drugim uporabnikom*, lahko pa tudi ureja in briše svoja obstoječa dovoljenja.

2.2.2. Običajni tok.

2.2.2.1. Odpre se formular, prek katerega določi novo dovoljenje.

2.2.2.2. Izbere uporabnika, ki mu bo dovolil akcije na svojih predmetih.

2.2.2.3. Izbere, na kaj se nanaša dovoljenje, to so lahko vprašanja, izpiti ali procesi priprave in poprave izpita.

2.2.2.4. Označi eno ali več akcij, kot so beri, uredi, izbriši in ustvari. Te akcije dovoli izvajati na svojih predmetih.

2.2.2.5. Potrdi formular in sistem shrani njegovo dovoljenje.

2.2.3. Alternativni tok.

2.2.3.1. V tabeli prikazanih dovoljenj izbere dovoljenje, ki ga želi urediti.

2.2.3.1.1. Odpre se formular, enak kot v 2.2.2.1., sledimo dogodkom po 2.2.2.1.

2.2.3.2. V tabeli izbere dovoljenje za brisanje.

2.2.3.2.1. Pojavi se potrditveno okno.

2.2.3.2.2. Uporabnik potrdi.

2.2.3.2.3. Sistem izbriše dovoljenje.

2.2.4. Predpogoj: Profesor mora biti prijavljen v sistem.

3. **Primer uporabe**

3.1. Ime primera uporabe: **Urejanje vprašanj**.

3.1.1. Kratek opis: Primer uporabe omogoča uporabniku dodajanje, urejanje in brisanje izpitnih vprašanj.

3.1.2. Akter: Administrator, Profesor, Asistent (če mu je dovoljeno s strani Administratorja ali Profesorja).

3.2. Tok dogodkov.

3.2.1. Uporabnik izbere možnost *Dodaj novo vprašanje*, lahko pa tudi ureja in briše vprašanja. Ima tudi možnost uvoza vprašanj prek tekstovnih datoteke in izvoza vprašanj v datoteko PDF.

3.2.2. Običajni tok.

3.2.2.1. Uporabnik izbere *Dodaj novo vprašanje* in odpre se mu formular.

3.2.2.2. V vnosna polja *Vsebina* vnese vsebino vprašanja in pripadajoče možne odgovore, odgovore lahko dodaja in briše.

3.2.2.3. Določene odgovore lahko označi kot pravilne.

3.2.2.4. V vnosna polja zapiše zahtevnost in pomembnost vprašanja, ki določata težo vprašanja.

3.2.2.5. Označi, ali je vprašanje namenjeno za kolokvij ali izpit.

3.2.2.6. V vnosno polje *Komentar* zapiše svoj komentar.

3.2.2.7. Na seznamu možnih smeri označi smeri, ki jim to vprašanje pripada.

3.2.2.8. S seznama možnih tem izbere temo vprašanja.

3.2.2.9. Potrdi formular in vprašanje se vnese v sistem z omenjenimi lastnostmi.

3.2.3. Alternativni tok.

3.2.3.1. Urejanje vprašanja.

3.2.3.1.1. Uporabnik izbere vprašanje v tabeli, ki ga želi urejati.

3.2.3.1.2. Uporabnik sledi dogodkom od 3.2.2.1.

3.2.3.2. Brisanje vprašanja.

3.2.3.2.1. Uporabnik izbere vprašanje v tabeli, ki ga želi izbrisati.

3.2.3.2.2. Pojavi se potrditveno okno.

3.2.3.2.3. Uporabnik ga potrdi.

3.2.3.2.4. Sistem izbriše vprašanje.

3.2.3.3. Brisanje več označenih vprašanj.

3.2.3.3.1. Uporabnik označi več vprašanj v tabeli in pritisne na gumb *Zbriši*.

3.2.3.3.2. Pojavi se potrditveno okno.

3.2.3.3.3. Uporabnik ga potrdi.

3.2.3.3.4. Sistem izbriše označena vprašanja.

3.2.3.4. Uvoz vprašanj.

3.2.3.4.1. Uporabnik izbere možnost *Uvozi vprašanja*.

3.2.3.4.2. Uporabnik obišče primer uporabe **Uvozi vprašanja iz txt, biv datoteke**.

3.2.4. Predpogoj: Uporabnik mora biti vpisan v sistem. Če ima uporabnik vlogo Asistenta, mora imeti dovoljenje primarnega akterja (glej primer uporabe **Dodeljevanje in urejanje svojih pravic**).

4. **Primer uporabe**

4.1. Ime primera uporabe: **Uvozi vprašanja iz txt, biv datoteke**.

4.1.1. Kratek opis: Primer uporabe uporabniku omogoča uvažanje vprašanj prek tekstovnih datotek.

4.1.2. Akter: Administrator, Profesor, Asistent (če mu je dovoljeno s strani Administratorja ali Profesorja).

4.2. Tok dogodkov.

4.2.1. Uporabnik izbere možnost *Uvozi vprašanja*.

4.2.2. Običajni tok.

4.2.2.1. Odpre se formular.

4.2.2.2. V formularju ima možnost izbrati datoteko za uvoz.

4.2.2.3. Potrdi formular in sistem uvozi vprašanja v bazo.

4.2.2.4. Sistem uporabniku izpiše vprašanja, uspešno uvožena v sistem.

4.2.3. Alternativni tok.

4.2.3.1. Napačen format datoteke.

4.2.3.1.1. Odpre se formular.

4.2.3.1.2. Uporabnik izbere napačen format datoteke prek formularja.

4.2.3.1.3. Potrdi formular in sistem izpiše obvestilo o napačnem formatu datoteke.

4.2.3.2. Napaka pri uvoženem vprašanju.

4.2.3.2.1. Odpre se formular.

4.2.3.2.2. Uporabnik izbere tekstovno datoteko pravilnega formata.

4.2.3.2.3. Potrdi formular in sistem javi napako pri uvoženih vprašanjih, ki niso pravilno zapisana.

4.2.4. Predpogoj: Uporabnik mora biti vpisan v sistem. Če ima uporabnik vlogo Asistenta, mora imeti dovoljenje primarnega akterja (glej primer uporabe **Dodeljevanje in urejanje svojih pravic**).

5. **Primer uporabe**

5.1. Ime primera uporabe: **Urejanje izpitov**.

5.1.1. Kratek opis: Primer uporabe omogoča uporabniku ustvarjanje, urejanje in brisanje izpitov.

5.1.2. Akter: Administrator, Profesor, Asistent (če mu je dovoljeno s strani Administratorja ali Profesorja).

5.2. Tok dogodkov.

5.2.1. Uporabnik izbere možnost *Ustvari nov izpit*, lahko pa tudi ureja ali izbriše izbrani izpit.

5.2.2. Običajni tok.

5.2.2.1. Uporabnik izbere možnost *Ustvari nov izpit* in odpre se formular.

5.2.2.2. V vnosno polje vpiše naslov izpita*.

5.2.2.3. Izbere datum izpita*.

5.2.2.4. V vnosno polje vpiše prostor izpita.

5.2.2.5. Določi mejo za pozitivno oceno*.

5.2.2.6. Izbere način točkovanja izpita (vse ali 0, delež pravilnih odgovorov, pravilni +, nepravilni –)*.

5.2.2.7. Potrdi formular in sistem shrani nov izpit, prikaže se tabela obstoječih izpitov.

5.2.3. Alternativni tok.

5.2.3.1. Uredi izpit.

5.2.3.1.1. Uporabnik izbere določen izpit v tabeli in odpre se mu formular kot v 5.2.2.1.

5.2.3.1.2. Sledi dogodkom od 5.2.2.1.

5.2.3.2. Izbriši izpit.

5.2.3.2.1. Uporabnik iz tabele izbere izpit, ki ga želi izbrisati.

5.2.3.2.2. Pojavi se potrditveno okno.

5.2.3.2.3. Uporabnik ga potrdi in sistem izbriše izpit.

5.2.3.3. Podrobnosti izpita.

5.2.3.3.1. Uporabnik iz tabele izbere izpit, ki ga želi pregledati.

5.2.3.3.2. Sistem na zaslonu prikaže vse lastnosti izpita. Prikazane so tudi prijave na izpit in vprašanja, vnešena v izpit, če so že nastavljena v primeru uporabe **Priprava in poprava izpita**.

5.2.3.4. Napaka pri dodajanju in urejanju izpita.

5.2.3.4.1. Uporabnik izbere *Ustvari nov izpit* ali izbere iz tabele izpit za urejanje.

5.2.3.4.2. Prikaže se formular.

5.2.3.4.3. Uporabnik tokrat ne izpolni vseh vnosnih polj, ki so označena z *, in potrdi formular.

5.2.3.4.4. Sistem opozori uporabnika, da je pozabil izpolniti določeno polje, in formular ostane odprt.

5.2.4. Predpogoj: Uporabnik mora biti vpisan v sistem. Če ima uporabnik vlogo Asistenta, mora imeti dovoljenje primarnega

akterja (glej primer uporabe **Dodeljevanje in urejanje svojih pravic**).

6. **Primer uporabe**

6.1. Ime primera uporabe: **Priprava in poprava izpita**.

6.1.1. Kratek opis: Primer uporabe omogoča, da uporabnik pripravi izpit. Izpitu doda ustrezna izpitna vprašanja, prijavi kandidate, natisne izpitne pole in izpite za vsakega kandidata. Pri popravi uporabnik optično prebere rešene izpitne pole in jih v obliki slik vnese v sistem, ki jih popravi in oceni.

6.1.2. Akter: Administrator, Profesor, Asistent (če mu je dovoljeno s strani Administratorja ali Profesorja).

6.2. Tok dogodkov.

6.2.1. Uporabnik izbere možnost *Zaženi proces priprave in poprave izpita* v tabeli že obstoječih izpitov.

6.2.2. Običajni tok.

6.2.2.1. Uporabnik sledi dogodkom v primeru uporabe **Dodaj vprašanja v izpit**.

6.2.2.2. Nato sledi dogodkom v primeru uporabe **Prijavi/Odjavi kandidate kandidate na izpit**.

6.2.2.3. Nato sledi dogodkom v primeru uporabe **Natisni izpite in izpitne pole**.

6.2.2.4. Nato sledi dogodkom v primeru uporabe **Skeniraj, popravi in oceni izpit**.

6.2.3. Predpogoj.

6.2.3.1. Uporabnik mora biti vpisan v sistem.

6.2.3.2. Če ima uporabnik vlogo Asistenta, mora imeti dovoljenje primarnega akterja (glej primer uporabe **Dodeljevanje in urejanje svojih pravic**).

6.2.3.3. Izpit, ki ga pripravimo in popravimo, mora biti že ustvarjen.

7. **Primer uporabe**

7.1. Ime primera uporabe: **Dodaj vprašanja v izpit**.

7.1.1. Kratek opis: Primer uporabe omogoča uporabniku dodajanje vprašanj v izpit.

7.1.2. Akter: Administrator, Profesor, Asistent (če mu je dovoljeno s strani Administratorja ali Profesorja).

7.2. Tok dogodkov.

7.2.1. Uporabnik izbere možnosti *Uredi vprašanja* pod določenim izpitom v tabeli izpitov.

7.2.2. Običajni tok.

7.2.2.1. Pojavita se dve tabeli: ena predstavlja izpit z vnešenimi vprašanji, druga prikaže vprašanja.

7.2.2.2. Uporabnik izbere vprašanje iz tabele vprašanj.

7.2.2.3. Uporabnik doda vprašanje v izpit.

7.2.2.4. Sistem shrani vprašanje v izpit in doda vprašanje v tabelo vprašanj izpita.

7.2.2.5. Uporabnik ponavlja dogodek 7.2.2.2., dokler niso vsa želena vprašanja v izpitu.

7.2.3. Alternativni tok.

7.2.3.1. Dodaj več vprašanj hkrati v izpit.

7.2.3.1.1. Uporabnik označi ustrezna vprašanja v tabeli vprašanj.

7.2.3.1.2. Pritisne gumb *Dodaj označena vprašanja v izpit.*

7.2.3.1.3. Pojavi se potrditveno okno.

7.2.3.1.4. Uporabnik ga potrdi.

7.2.3.1.5. Sistem shrani vprašanja v izpit in doda vprašanja v tabelo vprašanj izpita.

7.2.3.2. Odstrani vprašanje iz izpita.

7.2.3.2.1. Uporabnik izbere vprašanje v tabeli vprašanj izpita.

7.2.3.2.2. Sistem odstrani vprašanje iz izpita.

7.2.3.3. Odstrani več vprašanj hkrati iz izpita.

7.2.3.3.1. Uporabnik označi ustrezna vprašanja izpita v tabeli.

7.2.3.3.2. Pritisne gumb *Odstrani označena vprašanja iz izpita.*

7.2.3.3.3. Pojavi se potrditveno okno.

7.2.3.3.4. Uporabnik ga potrdi.

7.2.3.3.5. Sistem izbriše vprašanja iz izpita.

7.2.4. Predpogoj.

7.2.4.1. Uporabnik mora biti vpisan v sistem.

7.2.4.2. Če ima uporabnik vlogo Asistenta, mora imeti dovoljenje primarnega akterja (glej primer uporabe **Dodeljevanje in urejanje svojih pravic**).

7.2.4.3. Izpit mora obstajati.

8. **Primer uporabe**

8.1. Ime primera uporabe: **Prijavi/Odjavi kandidate na izpit**.

8.1.1. Kratek opis: Primer uporabe omogoča uporabniku prijavljanje in odjavljanje kandidatov od izpita.

8.1.2. Akter: Administrator, Profesor, Asistent (če mu je dovoljeno s strani Administratorja ali Profesorja).

8.2. Tok dogodkov.

8.2.1. Uporabnik izbere možnost *Uredi prijave* pod določenim izpitom v tabeli izpitov.

8.2.2. Običajni tok.

8.2.2.1. Pojavita se dva seznama, na prvem seznamu so na izpit neprijavljeni kandidati, na drugem pa prijavljeni.

8.2.2.2. Uporabnik v prvi tabeli (vsi neprijavljeni kandidati) izbere kandidate, ki jih želi prijaviti na izpit.

8.2.2.3. Uporabnik klikne gumb *Prijavi*.

8.2.2.4. Sistem ustvari prijave na ta izpit, v drugi tabeli se pojavijo prijavljeni kandidati.

8.2.2.5. Uporabnik ponavlja dogodek 8.2.2.2., dokler niso vsi ustrezni kandidati prijavljeni.

8.2.3. Alternativni tok.

8.2.3.1. Odjavi kandidate z izpita.

8.2.3.1.1. Uporabnik označi ustrezne kandidate v tabeli prijavljenih kandidatov.

8.2.3.1.2. Pritisne gumb *Odjavi*.

8.2.3.1.3. Sistem izbriše izbrane prijave in kandidati se preselijo iz druge v prvo tabelo.

8.2.3.2. Opozorilo o izpitu brez izpitnih vprašanj.

8.2.3.2.1. Uporabnik v primeru uporabe **Dodaj vprašanja v izpit** ne doda nobenih vprašanj v izpit.

8.2.3.2.2. Sistem opozori uporabnika, da izpit ne vsebuje izpitnih vprašanj, in se ne sooči s primerom uporabe **Prijavi/Odjavi kandidate**.

8.2.4. Predpogoj.

8.2.4.1. Uporabnik mora biti vpisan v sistem.

8.2.4.2. Če ima uporabnik vlogo Asistenta, mora imeti dovoljenje primarnega akterja (glej primer uporabe **Dodeljevanje in urejanje svojih pravic**).

8.2.4.3. Izpit mora obstajati.

8.2.4.4. Primer uporabe **Dodaj vprašanja v izpit** mora biti končan, da se začne ta primer uporabe.

9. **Primer uporabe**

9.1. Ime primera uporabe: **Natisni izpite in izpitne pole**.

9.1.1. Kratek opis: Primer uporabe uporabniku omogoča natis izpitnih pol in izpitov za vsakega prijavljenega kandidata.

9.1.2. Akter: Primarni: Administrator, Profesor, Asistent (če mu je dovoljeno s strani Administratorja ali Profesorja). Sekundarni: Tiskalnik.

9.2. Tok dogodkov.

9.2.1. Uporabnik izbere možnost *Tiskaj* pod določenim izpitom v tabeli izpitov.

9.2.2. Običajni tok.

9.2.2.1. Prikaže se formular.

9.2.2.2. Uporabnik v vnosno polje napiše navodila, ki se bodo prikazala na vsaki izpitni poli (navodila za reševanje izpita).

9.2.2.3. Uporabnik izbere možnost mešanja vprašanj med kandidati pri samem tiskanju (vsak kandidat dobi svoj vrstni red vprašanj/vsi imajo isti vrstni red vprašanj).

9.2.2.4. Uporabnik izbere način tiskanja, enostransko ali obojestransko.

9.2.2.5. Uporabnik potrdi formular.

9.2.2.6. Sistem zgenerira dokument PDF, ki predstavlja vse izpite in izpitne pole za vsakega kandidata posebej.

9.2.2.7. Uporabnik prevzame dokument PDF in ga natisne.

9.2.3. Alternativni tok.

9.2.3.1. Opozorilo neprevzetega dokumenta PDF.

9.2.3.1.1. Uporabnik izbere možnost *Popravi* pod določenim izpitom v tabeli izpitov, ne da bi natisnil izpite in izpitne pole.

9.2.3.1.2. Sistem ga opozori, da je treba prevzeti dokument PDF in ga natisniti.

9.2.3.2. Opozorilo neprijavljenih kandidatov.

9.2.3.2.1. Uporabnik ne prijavi nobenega kandidata v primeru uporabe **Prijavi/Odjavi kandidate na izpit**.

9.2.3.2.2. Sistem opozori uporabnika, da izpit nima nobenih prijav, in se ne sooči s primerom uporabe **Natisni izpite in izpitne pole**.

9.2.4. Predpogoj.

9.2.4.1. Uporabnik mora biti vpisan v sistem.

9.2.4.2. Če ima uporabnik vlogo Asistenta, mora imeti dovoljenje primarnega akterja (glej primer uporabe **Dodeljevanje in urejanje svojih pravic**).

9.2.4.3. Izpit mora obstajati.

9.2.4.4. Primer uporabe **Prijavi/Odjavi kandidate na izpit** mora biti končan, da se začne ta primer uporabe.

10. **Primer uporabe**

10.1. Ime primera uporabe: **Optično preberi, popravi in oceni izpit**.

10.1.1. Kratek opis: Primer uporabe omogoča uporabniku, da optično prebrane izpitne pole vnese v sistem, kjer se jih elektronsko zazna in popravi. Na podlagi poprave se ocenijo izpiti kandidatov.

10.1.2. Akter: Primarni: Administrator, Profesor, Asistent (če mu je dovoljeno s strani Administratorja ali Profesorja). Sekundarni: Skener.

10.2. Tok dogodkov.

10.2.1. Uporabnik izbere možnost *Popravi* pod določenim izpitom v tabeli izpitov.

10.2.2. Običajni tok.

10.2.2.1. Uporabniku se prikaže vmesnik.

10.2.2.2. Uporabnik preko vmesnika naloži slike izpitnih pol.

10.2.2.3. Sistem slike sprocesira in uporabniku sporoči, da je zaznal vse označene odgovore kandidatov.

10.2.2.4. Uporabnik pritisne na gumb *Shrani popravljene izpitne pole*.

10.2.2.5. Sistem oceni izpitne pole za vsakega kandidata in shrani rezultate v sistem.

10.2.3. Alternativni tok.

10.2.3.1. Napačna zaznava izpitne pole.

10.2.3.1.1. Vmesnik prikaže zaznane slike izpitnih pol.

10.2.3.1.2. Opozori uporabnika na zaznave izpitnih pol z napako.

10.2.3.1.3. Uporabnik izbere izpitno polo z napačno zaznavo.

10.2.3.1.4. Vmesnik prikaže izpitno polo in napačno zaznane označbe.

10.2.3.1.5. Uporabnik prek vmesnika popravi napake v zaznavi.

10.2.3.1.6. Uporabnik sledi dogodkom od 10.2.2.4.

10.2.3.2. Napaka pri ocenjevanju izpitnih pol.

10.2.3.2.1. Uporabnik pritisne na gumb *Shrani popravljene izpitne pole*.

10.2.3.2.2. Sistem ga opozori, pri katerih izpitnih polah je prišlo do napake pri shranjevanju v bazo.

10.2.4. Predpogoj.

10.2.4.1. Uporabnik mora biti vpisan v sistem.

10.2.4.2. Če ima uporabnik vlogo Asistenta, mora imeti dovoljenje s strani Administratorja ali Profesorja (glej primer uporabe **Dodeljevanje in urejanje svojih pravic**).

10.2.4.3. Izpit mora obstajati.

10.2.4.4. Primer uporabe **Natisni izpite in izpitne pole** mora biti končan, da se ta primer uporabe začne.

10.2.4.5. Izpitne pole morajo kandidati fizično rešiti in oddati pristojnim.

10.2.4.6. Izpitne pole je treba optično prebrati in shraniti slike.

11. **Primer uporabe**

11.1. Ime primera uporabe: **Pregled statističnih podatkov**.

11.1.1. Kratek opis: Primer uporabe uporabniku omogoča vpogled v statistične podatke uspešnosti reševanja posameznega vprašanja ali izpita. Pogleda si lahko tudi uspešnost uporabnika pri njegovih izpitih in uspešnost posameznih kandidatov.

11.1.2. Akter: Administrator, Profesor, Asistent.

11.2. Tok dogodkov.

11.2.1. Uporabnik izbere želeno statistiko uporabnika, posameznih vprašanj, kandidatov ali izpitov.

11.2.2. Običajni tok.

11.2.2.1. Uporabnik lahko vstopi v primer uporabe **Statistični podatki glede na uporabnika**, **Statistični podatki glede na vprašanje**, **Statistični podatki glede na izpit** ali **Statistični podatki glede na kandidata**.

11.2.3. Predpogoj.

11.2.3.1. Uporabnik mora biti vpisan v sistem.

12. **Primer uporabe**

12.1. Ime primera uporabe: **Statistični podatki glede na vprašanje**.

12.1.1. Kratek opis: Primer uporabe omogoča uporabniku vpogled v statistične podatke uspešnosti reševanja posameznega vprašanja.

12.1.2. Akter: Administrator, Profesor, Asistent.

12.2. Tok dogodkov.

12.2.1. Uporabnik izbere možnost *Statistika vprašanja*.

12.2.2. Običajni tok.

12.2.2.1. Sistem izpiše tabelo uporabniških in dovoljenih vprašanj.

12.2.2.2. Uporabnik izbere *Statistika vprašanja* pod določenim vprašanjem.

12.2.2.3. Uporabniku se prikažejo grafi s statistiko uspešnosti reševanja vprašanja.

12.2.3. Alternativni tok.

12.2.3.1. Ni dovolj podatkov za prikaz statistike.

12.2.3.1.1. Uporabnik izbere možnost *Statistika vprašanja* pod določenim vprašanjem, ki še ni nastopalo v izpitu.

12.2.3.1.2. Uporabniku se prikažejo prazni grafi, izpiše se opozorilo, da ni dovolj podatkov za izris grafov.

12.2.4. Predpogoj.

12.2.4.1. Uporabnik mora biti vpisan v sistem.

13. **Primer uporabe**

13.1. Ime primera uporabe: **Statistični podatki glede na izpit**.

13.1.1. Kratek opis: Primer uporabe omogoča uporabniku vpogled v statistične podatke uspešnosti reševanja posameznega izpita.

13.1.2. Akter: Administrator, Profesor, Asistent.

13.2. Tok dogodkov.

13.2.1. Uporabnik izbere možnost *Statistika izpita*.

13.2.2. Običajni tok.

13.2.2.1. Sistem izpiše uporabniške in dovoljene izpite, ki so že popravljeni.

13.2.2.2. Uporabnik izbere možnost *Statistika izpita* pod določenim izpitom.

13.2.2.3. Uporabniku se prikažejo grafi s statistiko uspešnosti reševanja vprašanja.

13.2.3. Predpogoj.

13.2.3.1. Uporabnik mora biti vpisan v sistem.

14. Primer uporabe

14.1. Ime primera uporabe: **Statistični podatki glede na kandidata**.

14.1.1. Kratek opis: Primer uporabe omogoča uporabniku vpogled v statistične podatke o uspešnosti reševanja posameznega kandidata.

14.1.2. Akter: Administrator, Profesor, Asistent.

14.2. Tok dogodkov.

14.2.1. Uporabnik izbere možnost *Statistika kandidata*.

14.2.2. Običajni tok.

14.2.2.1. Sistem izpiše tabelo kandidatov.

14.2.2.2. Uporabnik izbere možnost *Statistični podatki kandidata* pod določenim kandidatom.

14.2.2.3. Uporabniku se prikažejo grafi s statistiko kandidatove uspešnosti reševanja skozi čas izpitov.

14.2.3. Alternativni tok.

14.2.3.1. Ni dovolj podatkov za prikaz statistike.

14.2.3.1.1. Uporabnik izbere možnost *Statistični podatki kandidata* pod določenim kandidatom, ki še ni opravljal izpita.

14.2.3.1.2. Uporabniku se prikažejo prazni grafi, izpiše se opozorilo, da ni dovolj podatkov za izris grafov.

14.2.4. Predpogoj.

14.2.4.1. Uporabnik mora biti vpisan v sistem.

15. **Primer uporabe**

15.1. Ime primera uporabe: **Statistični podatki glede na uporabnika**.

15.1.1. Kratek opis: Primer uporabe uporabniku omogoča vpogled v njegove statistične podatke o uspešnosti reševanja posameznega kandidata.

15.1.2. Akter: Profesor.

15.2. Tok dogodkov.

15.2.1. Uporabnik izbere možnost *Statistika uporabnika*.

15.2.2. Običajni tok.

15.2.2.1. Uporabniku se prikaže graf s statistiko uspešnosti reševanja kandidatov skozi čas njegovih izpitov.

15.2.3. Alternativni tok.

15.2.3.1. Ni dovolj podatkov za prikaz statistike.

15.2.3.1.1. Uporabniku se prikaže prazen graf, izpiše se opozorilo, da ni dovolj podatkov za izris grafa (ko profesor še ni izpeljal nobenega izpita v svojem imenu).

15.2.4. Predpogoj.

15.2.4.1. Uporabnik mora biti vpisan v sistem.

16. **Primer uporabe**

16.1. Ime primera uporabe: **Urejanje kandidatov**.

16.1.1. Kratek opis: Primer uporabe omogoča uporabniku dodajanje, urejanje in brisanje kandidatov.

16.1.2. Akter: Administrator, Profesor, Asistent.

16.2. Tok dogodkov.

16.2.1. Uporabnik izbere možnost *Dodaj kandidata* ali pa uredi ali izbriše kandidata.

16.2.2. Običajni tok.

16.2.2.1. Odpre se formular.

16.2.2.2. Uporabnik vpiše vpisno številko kandidata v vnosno polje*.

16.2.2.3. Uporabnik vpiše ime in priimek kandidata v vnosno polje*.

16.2.2.4. Uporabnik vpiše elektronski naslov kandidata v vnosno polje.

16.2.2.5. Uporabnik izbere študijski program, ki mu kandidata pripada*.

16.2.2.6. Uporabnik izbere študijsko smer kandidata*.

16.2.2.7. Uporabnik izbere letnik, ki ga kandidat obiskuje*.

16.2.2.8. Uporabnik potrdi formular.

16.2.2.9. Sistem shrani novega kandidata in prikaže tabelo s kandidati.

16.2.3. Alternativni tok.

16.2.3.1. Uredi kandidata.

16.2.3.1.1. Uporabnik izbere možnost *Uredi kandidata* pod določenim kandidatom v tabeli.

16.2.3.1.2. Sledi dogodkom od 16.2.2.1. in sledi dogodkom.

16.2.3.1.3. Uporabnik potrdi formular.

16.2.3.1.4. Sistem posodobi kandidata in prikaže tabelo s kandidati.

16.2.3.2. Izbriši kandidata.

16.2.3.2.1. Uporabnik izbere možnost *Izbriši kandidata* pod določenim kandidatom v tabeli.

16.2.3.2.2. Pojavi se potrditveno okno.

16.2.3.2.3. Uporabnik okno potrdi in sistem izbriše kandidata.

16.2.3.3. Napaka pri dodajanju in urejanju kandidata.

16.2.3.3.1. Uporabnik izbere urejanje kandidata v tabeli ali možnost *Dodaj kandidata*.

16.2.3.3.2. Pojavi se formular kot v 16.2.2.1. in sledi dogodkom.

16.2.3.3.3. Uporabnik ne izpolni vseh vnosnih polj, ki so označena z *, in potrdi formular.

16.2.3.3.4. Sistem opozori uporabnika, da je pozabil izpolniti določeno polje, in formular ostane odprt.

16.2.3.4. Napaka vpisne številke pri dodajanju in urejanju kandidata.

16.2.3.4.1. Uporabnik izbere urejanje kandidata v tabeli ali *Dodaj kandidata*.

16.2.3.4.2. Pojavi se formular kot v 16.2.2.1. in sledi dogodkom.

16.2.3.4.3. Uporabnik ne izpolni vnosnega polja *Vpisna številka* s številkami ali pa ga izpolni z manj kot osmimi številkami in potrdi formular.

16.2.3.4.4. Sistem opozori uporabnika, da je pozabil izpolniti določeno polje s številkami, in formular ostane odprt.

16.2.4. Predpogoj: Uporabnik mora biti vpisan v sistem.

17. **Primer uporabe**

17.1. Ime primera uporabe: **Urejanje uporabnikov sistema**.

17.1.1. Kratek opis: Primer uporabe omogoča administratorju dodajanje, urejanje in brisanje uporabnikov sistema.

17.1.2. Akter: Administrator.

17.2. Tok dogodkov.

17.2.1. Administrator izbere možnost *Dodaj uporabnika* in uredi ali izbriše uporabnika iz tabele obstoječih uporabnikov.

17.2.2. Običajni tok.

17.2.2.1. Odpre se mu formular.

17.2.2.2. Vnese uporabniško ime uporabnika v vnosno polje*.

17.2.2.3. Vnese začasno geslo uporabnika v vnosno polje*.

17.2.2.4. Vnese ime in priimek uporabnika v vnosno polje*.

17.2.2.5. Vnese elektronski naslov uporabnika v vnosno polje*.

17.2.2.6. Vnese naslov uporabnika v vnosno polje*.

17.2.2.7. Določi vlogo uporabnika (Administrator, Profesor ali Asistent)*.

17.2.2.8. Potrdi formular.

17.2.2.9. Sistem shrani novega uporabnika in prikaže tabelo z uporabniki.

17.2.3. Alternativni tok.

17.2.3.1. Uredi uporabnika.

17.2.3.1.1. Administrator izbere možnost *Uredi uporabnika* pod določenim uporabnikom v tabeli.

17.2.3.1.2. Odpre se formular in sledi dogodkom od 17.2.2.1.

17.2.3.1.3. Sistem posodobi uporabnika in prikaže tabelo z uporabniki.

17.2.3.2. Izbriši uporabnika.

17.2.3.2.1. Administrator izbere možnost *Izbriši uporabnika* pod določenim uporabnikom v tabeli.

17.2.3.2.2. Pojavi se potrditveno okno.

17.2.3.2.3. Administrator ga potrdi in sistem izbriše uporabnika.

17.2.3.3. Napaka pri dodajanju in urejanju uporabnika.

17.2.3.3.1. Administrator izbere možnost *Uredi uporabnika* pod določenim uporabnikom v tabeli ali možnost *Dodaj uporabnika*.

17.2.3.3.2. Pojavi se formular kot v 17.2.2.1. in sledi dogodkom.

17.2.3.3.3. Administrator ne izpolni vseh vnosnih polj, ki so označena z *, in potrdi formular.

17.2.3.3.4. Sistem opozori Administratorja, da je pozabil izpolniti določeno polje, in formular ostane odprt.

17.2.4. Predpogoj: Administrator mora biti vpisan v sistem.

18. **Primer uporabe**

18.1. Ime primera uporabe: **Urejanje splošnih nastavitev – urejanje šifranta**.

18.1.1. Kratek opis: Primer uporabe Administratorju omogoča dodajanje, urejanje in brisanje tem, študijskih smeri, študijskih programov in letnikov.

18.1.2. Akter: Administrator.

18.2. Tok dogodkov.

18.2.1. Administrator izbere možnost *Dodaj temo/smer/program/letnik* ali pa v tabelah tem, smeri, letnikov in programov izbere *Uredi temo/smer/program/letnik* in *Izbriši temo/smer/program/letnik*.

18.2.2. Običajni tok.

18.2.2.1. Administrator izbere *Dodaj temo/smer/program/letnik*.

18.2.2.2. Pojavi se formular z vnosnim poljem.

18.2.2.3. Vpiše ime teme/smeri/programa/letnika v vnosno polje.

18.2.2.4. Potrdi formular.

18.2.2.5. Sistem shrani novo temo/smer/program/letnik in prikaže tabele tem, smeri, programov, letnikov.

18.2.3. Alternativni tok.

18.2.3.1. Uredi temo/smer/program/letnik.

18.2.3.1.1. Administrator izbere *Uredi temo/smer/program/letnik* v tabeli in odpre se formular kot v 18.2.2.1.

18.2.3.1.2. Administrator uredi ime v vnosnem polju in potrdi formular.

18.2.3.1.3. Sistem posodobi temo/smer/program/letnik in prikaže tabele tem, smeri, programov, letnikov.

18.2.3.2. Izbriši temo/smer/program/letnik.

18.2.3.2.1. Administrator izbere *Izbriši temo/smer/program/letnik* v tabeli.

18.2.3.2.2. Pojavi se potrditveno okno.

18.2.3.2.3. Administrator ga potrdi in sistem izbriše temo/smer/program/letnik.

18.2.3.3. Napaka pri dodajanju in urejanju uporabnika.

18.2.3.3.1. Administrator izbere *Uredi uporabnika* v tabeli ali *Dodaj uporabnika* in pojavi se formular kot v 18.2.2.1.

18.2.3.3.2. Administrator ne izpolni vnosnega polja za ime in potrdi formular.

18.2.3.3.3. Sistem opozori Administratorja, da je pozabil izpolniti vnosno polje *Ime*, in formular ostane odprt.

18.2.4. Predpogoj: Administrator mora biti vpisan v sistem.

19. **Primer uporabe**

19.1. Ime primera uporabe: **Vpis v sistem**.

19.1.1. Kratek opis: Primer uporabe uporabniku omogoča vpis v sistem.

19.1.2. Akter: Administrator, Profesor, Asistent.

19.2. Tok dogodkov.

19.2.1. Uporabnik vstopi na spletno stran sistema, kjer se pojavi vpisno okno.

19.2.2. Običajni tok.

19.2.2.1. Uporabnik v vnosno polje vnese uporabniško ime in geslo.

19.2.2.2. Pritisne na gumb *Vpiši se*.

19.2.2.3. Sistem preveri uporabniško ime in geslo ter sprejme uporabnika.

19.2.2.4. Uporabnik vstopi v primer uporabe **Urejanje vprašanj**.

19.2.3. Alternativni tok.

19.2.3.1. Napačno geslo ali napačno uporabniško ime.

19.2.3.1.1. Uporabnik vnese v vnosno polje uporabniško ime in geslo.

19.2.3.1.2. Pritisne na gumb *Vpiši se*.

19.2.3.1.3. Sistem ugotovi, da v bazi ni uporabnika s tem uporabniškim imenom oz. geslom.

19.2.3.1.4. Sistem opozori uporabnika, da je vpisal napačno uporabniško ime ali geslo.

19.2.3.1.5. Uporabniku se zopet pokaže vpisno okno.

19.2.3.2. Prvi vpis v sistem.

19.2.3.2.1. Uporabnik vnese v vnosno polje uporabniško ime in geslo.

19.2.3.2.2. Pritisne na gumb *Vpiši se*.

19.2.3.2.3. Sistem preveri uporabniško ime in geslo. Ugotovi, da se uporabnik prvič vpisuje v sistem.

19.2.3.2.4. Uporabnik vstopi v primer uporabe **Uredi svoj profil**.

20. **Primer uporabe**

20.1. Ime primera uporabe: **Uredi svoj profil**.

20.1.1. Kratek opis: Primer uporabe omogoča uporabniku, da si spremeni svoje podatke, uporabniško ime in geslo.

20.1.2. Akter: Administrator, Profesor, Asistent.

20.2. Tok dogodkov.

20.2.1. Uporabnik izbere možnost *Uredi svoj profil*.

20.2.2. Običajni tok.

20.2.2.1. Uporabniku se odpre formular.

20.2.2.2. Vnese uporabniško ime v vnosno polje*.

20.2.2.3. Vnese geslo v vnosno polje*.

20.2.2.4. Vnese svoj ime in priimek v vnosno polje*.

20.2.2.5. Vnese svoj elektronski naslov v vnosno polje*.

20.2.2.6. Vnese svoj naslov v vnosno polje*.

20.2.2.7. Potrdi formular.

20.2.2.8. Sistem posodobi uporabnika.

20.2.3. Alternativni tok.

20.2.3.1. Napaka pri urejanju profila.

20.2.3.1.1. Uporabniku se odpre formular z vnosnimi polji.

20.2.3.1.2. Uporabnik pusti nekatera vnosna polja, ki so označena z *, prazna in potrdi formular.

20.2.3.1.3. Sistem opozori uporabnika, da je pozabil izpolniti določeno polje, in formular ostane odprt.

20.2.4. Predpogoj: Uporabnik mora biti vpisan v sistem.

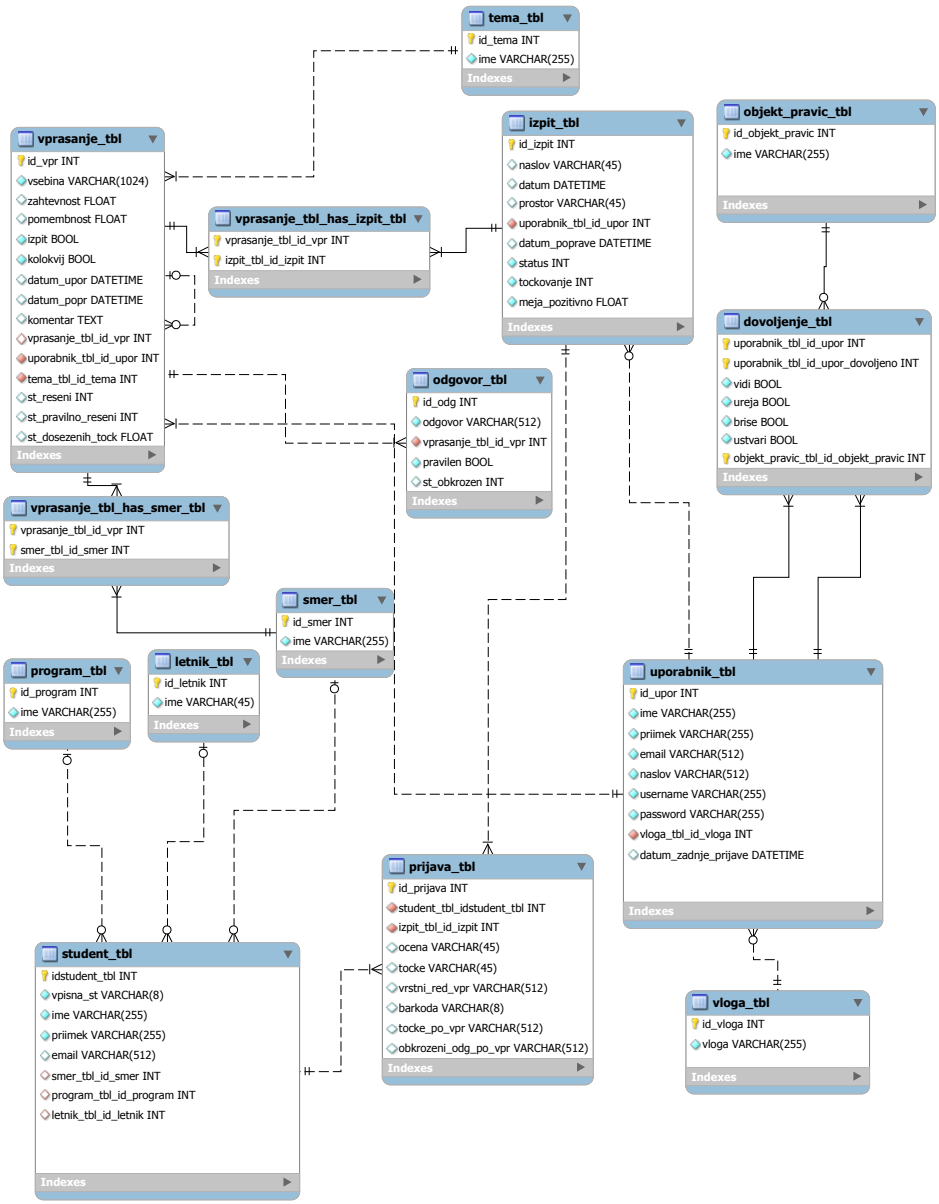
4.3 Razvoj aplikacije v fazi načrtovanja

V fazi načrtovanja sem se osredotočil predvsem na načrtovanje podatkovne baze. Načrtovanje programskih modulov sem usmeril v način strukture MVC, ki jo zahteva spletno ogrodje Yii.

4.3.1 Načrtovanje podatkovne baze

Za sestavo diagrama EER sem uporabil orodje MySQL Workbench, opisano v poglavju 3.3. Diagram, ki je predstavljen na sliki 4.2, se gradi kot konceptualni model, saj ga sestavljajo entitetni tipi in povezave med njimi. Prehod s konceptualnega modela na logični model je avtomatiziran že med samo gradnjo diagrama (primer: ko uporabimo povezavo Many-to-many, že avtomatično doda vmesno tabelo). Relacije med entitetnimi tipi so podrobneje prikazane na sliki 4.2. Opisi tipov:

- **Uporabnik:** Predstavlja uporabnika spletne aplikacije. Poleg običajnih lastnosti, kot so *ime*, *priimek*, *email* in *naslov*, vsebuje še *uporabniško*



Slika 4.2: Diagram EER podatkovne baze

ime in *geslo* za vstop v sistem. Poleg tega vsebuje tudi *datum zadnje prijave* in pa povezavo na entitetni tip **Vloga**.

- **Vprašanje:** Predstavlja vprašanje, ki ga bo mogoče vstaviti v izpit. Najbolj pomemben atribut je *vsebina*, ki določa vprašanje. Vsebuje atributa *zahtevnost* in *pomembnost*, ki določata težo vprašanju. *Datum uporabe* je datum, ko je bil izpit nazadnje uporabljen. *Datum poprave* je datum zadnjega urejanja vprašanja. Imamo še zastavici *kolokvij* in *izpit*, ki določata, ali je vprašanje namenjeno izpitu ali kolokviju. *Komentar* je komentar vprašanja. Zadnji trije atributi, *število rešenih*, *število pravilno rešenih* in *število doseženih točk*, so namenjeni statistični obdelavi vprašanja. Vprašanje je v razmerju s tipom **Odgovor**, saj lahko vprašanje vsebuje več odgovorov. Razmerje s tipom **Uporabnik** določa avtorja vprašanja. Vprašanje je tudi v razmerju s samim seboj, saj lahko izpitno vprašanje vsebuje tudi podvprašanje in druge možne odgovore. Identifikator vprašanja je *id_vpr*.

- **Odgovor:** Predstavlja možen odgovor izpitnega vprašanja, ki ga študentje lahko označijo. Odgovor vsebuje atribut *odgovor*, ki določa vsebino odgovora. Naslednji atribut, *pravilen*, določa zastavico, ali je odgovor pravilen ali ne. Atribut *st_obkrozen* je namenjen statistični obdelavi izpitnega vprašanja. Razmerje s tipom **Vprašanje** določa, da pripada natanko enemu vprašanju. Identifikator je *id_odg*.

- **Izpit:** Izpit vsebuje običajne atribute, kot so *naslov*, *prostor* in *datum*. Atribut *datum_poprave* je datum poprave izpita prek aplikacije. Atribut *status* določa stanje procesa pri pripravi in popravi izpita (stanja so: uredi vprašanja v izpitu, prijavi kandidate na izpit, natisni izpite in izpitne pole, popravi in oceni izpite). Atribut *meja_za_pozitivno* je decimalna številka, ki določa mejo za pozitivno oceno, *tockovanje* pa določa način točkovanja rešenih vprašanj izpita. Izpit je v razmerju s tipom **Uporabnik**, ki določa lastnika izpita. Identifikator je *id_izpit*.

- **Študent:** Študent, kot samo ime pove, predstavlja študente. Ima običajne atribute, kot so *ime*, *priimek*, *vpisna* (vpisna številka) in *email*. Razmerje s tipi **Program**, **Smer** in **Letnik** določa, v kateri študijski program, smer in letnik je študent vpisan. Identifikator je *idstudent_tbl*.

- **Prijava:** Prijava načeloma predstavlja razmerje med tipoma **Študent** in **Izpit**. Študent je lahko prijavljen na veliko izpitov, izpit pa lahko vsebuje veliko prijavljenih študentov. Poleg tega vsebuje še druge pomembne atribute. Atributa *ocena* in *tocke* prikazujeta rezultat študenta po popravi izpita. Atribut *vrstni_red_vpr* predstavlja vrstni red vprašanj, ki jih bo študent dobil na izpitu. To omogoča možnost mešanja vprašanj in odgovorov za vsakega kandidata posebej. Atribut *barkoda* je koda, prepoznana na izpitni poli, ki se mora ujemati z identifikatorjem te entitete. Atribut *tocke_po_vpr* je zapis, koliko točk je dosegel kandidat pri vsakem vprašanju, *obkrozeni_odg_po_vpr* pa zapis, katere odgovore je označil pri posameznem vprašanju. Identifikator je *id_prijava*.

- **Dovoljenje:** Predstavlja dovoljenje, ki ga nudi uporabnik drugemu uporabniku za svoja vprašanja, izpite ter procese priprave in poprave izpita (primer: Profesor A dovoli branje in urejanje svojih vprašanj Asistentu B, ne dovoli pa mu ustvarjanja in brisanja vprašanj). Sestavljajo ga atributi *vidi*, *ureja*, *briše* in *ustvari*, ki služijo kot zastavice pri določanju dovoljenja. Dovoljenje je v razmerju s tipom **Objekt pravic** in dvakrat s tipom **Uporabnik**, ki določa "KDO (uporabnik) KOMU (uporabnik) dovoli KAJ (objekt pravic)". Identifikatorji so *uporabnik_tbl_id_upor*, *uporabnik_tbl_id_upor_dovoljeno* in *objekt_pravic_tbl_id_objekt_pravic*.

- **Vloga:** Predstavlja vloge uporabnika, kot so Administrator, Profesor in Asistent, in je v razmerju s tipom **Uporabnik**. Pripada natanko enemu uporabniku. Atribut *vloga* je ime vloge. Identifikator je *id_vloga*.

- **Tema:** Vsako izpitno vprašanje mora pripadati neki temi oz. nekemu sklopu snovi. Tema ima atribut *ime*, kjer je zapisano ime teme. Iden-

tifikator je *id_tema*.

- **Program:** Vsak študent spada v določen študijski program, zato je v razmerju s tipom **Študent**. Program ima atribut *ime*, kjer je zapisano ime programa. Identifikator je *id_program*.

- **Letnik:** Vsak študent obiskuje določen letnik, zato je v razmerju s tipom **Študent**. Letnik ima atribut *ime*, kjer je zapisano ime letnika. Identifikator je *id_letnik*.

- **Smer:** Študent je v času študija izbral določeno smer. Smer je v razmerju s tipom **Študent**. Veliko študentov pripada določeni smeri. Smer je tudi v razmerju z tipom **Vprašanje**, saj so vprašanja lahko namenjena študentom določene smeri, tudi za več različnih smeri hkrati. Ima atribut *ime*, ki predstavlja ime smeri. Identifikator je *id_smeri*.

- **Objekt pravic:** Objekt pravic predstavlja, na kaj se nanaša dovoljenje, zato je v razmerju s tipom **Dovoljenje**. Pripada natanko enemu dovoljenju. Vsebuje atribut *ime*, ki predstavlja ime objekta, na katerega se nanaša dovoljenje (primer: vprašanja, izpit). Identifikator je *id_objekt_pravic*.

4.3.2 Načrtovanje programskih modulov

Načrtovanje programskih modulov skoraj ni bilo potrebno. Del načrtovanja je določen že vnaprej s strukturo MVC, ki jo zahteva spletno ogrodje Yii. Osredotočiti sem se moral predvsem na kontrolerje in na njihove pripadajoče funkcije. Model primerov uporabe (opisan v 4.2.2) je bilo potrebno preslikati v programske module, v tem primeru v ustrezne kontrolerje. Vsak primer uporabe oz. oblak primerov uporabe (ki so med seboj povezani) je predstavljal svoj kontroler z lastnimi funkcijami. Izjema je morda oblak primerov uporabe, ki predstavlja urejanje izpita in njegovo razširitev. Tukaj sem ločil dva kontrolerja: priprava

in poprava. Priprava skrbi za celoten proces priprave, od kreiranja izpita, vnašanja vprašanj v izpit do prijavljanja kandidatov na izpit in tiskanja izpita. Poprava skrbi za popravo in ocenitev izpita.

4.4 Razvoj aplikacije v fazi izvedbe

Za implementacijo spletne aplikacije e-Asistent sem uporabil spletno ogrodje Yii. Z ukazom yiic (prek skripte yiic.php) sem ustvaril projekt z organizirano strukturo direktorijev in datotek. Struktura mi je že na datotečnem nivoju ločila sklope kode, ki predstavljajo kontrolerje, modele in poglede. Za lažje ravnanje s kodo sem projekt uvozil v razvojno orodje NetBeans IDE. Projektu sem dodal povezave tudi na knjižnice spletnega ogrodja Yii, s čimer sem si poenostavil dostop do funkcij in pospešil izdelavo aplikacije.

Kontroler, kot pomemben del arhitekture MVC, predstavlja sklop funkcij oz. akcij. Funkcijo kontrolerja bom od tu naprej naslavljal kar z besedo akcija, saj se mi zdi primernejša. Ko uporabnik dostopi do določene podstrani spletne aplikacije preko naslova URL, pot določa klic ustreznega kontrolerja in ustrezne akcije (opisano v poglavju 3.2.5). To pomeni, da uporabnik izvrši akcijo na kontrolerju. Vsak kontroler uporabi ustrezne modele in prikaže ustrezne poglede v določeni akciji.

V fazi analiziranja (opisano v poglavju 4.2.2) sem opisal primere uporabe, ki vsebujejo določene funkcionalnosti. Te funkcionalnosti sem implementiral preko kontrolerjev in njihovih akcij. Aplikacijo sestavljajo naslednji kontrolerji:

- **Vpis**,
- **Vprašanja**,
- **Priprava**,
- **Poprava**,
- **Uporabniki**,

- **Kandidati**,
- **Nastavitve**,
- **Dovoljenja**,
- **Analiza**.

V naslednjih poglavjih bom podrobneje opisal omenjene kontrolerje in s tem celotno implementacijo spletne aplikacije e-Asistent.

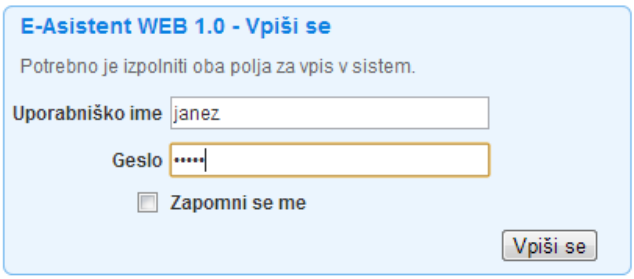
4.4.1 Vpis

Kontroler Vpis skrbi za kontroliran vstop v sistem preko preverjanja uporabniškega imena in gesla. Vsebuje naslednji akciji:

- **index:** Akcija se imenuje *index*, tako da lahko do nje dostopamo samo z imenom kontrolerja. Ta funkcija prikaže pogled *vpis/index.php*, ki je na sliki 4.3. Ko vpišemo uporabniško ime in geslo, se ta funkcija zopet izvrši in preveri, ali imamo posredovane podatke prek metode POST. Obstoj uporabnika in validacijo preverimo prek modela *Uporabnik*. Če uporabnik obstaja, ga zapišemo v sejno spremenljivko in usmerimo v kontroler **Vprašanja**. Če je obkljukana tudi možnost *Zapomni se me*, uporabnika shranimo v piškotek, kar mu omogoča avtomatski vstop v sistem nadaljnjih sedem dni.
- **izpis:** Akcija uporabnika izpiše iz sistema z izbrisom sejne spremenljivke uporabnika oz. piškotka. Uporabnika zopet usmerimo na kontroler **Vpis**.

4.4.2 Vprašanja

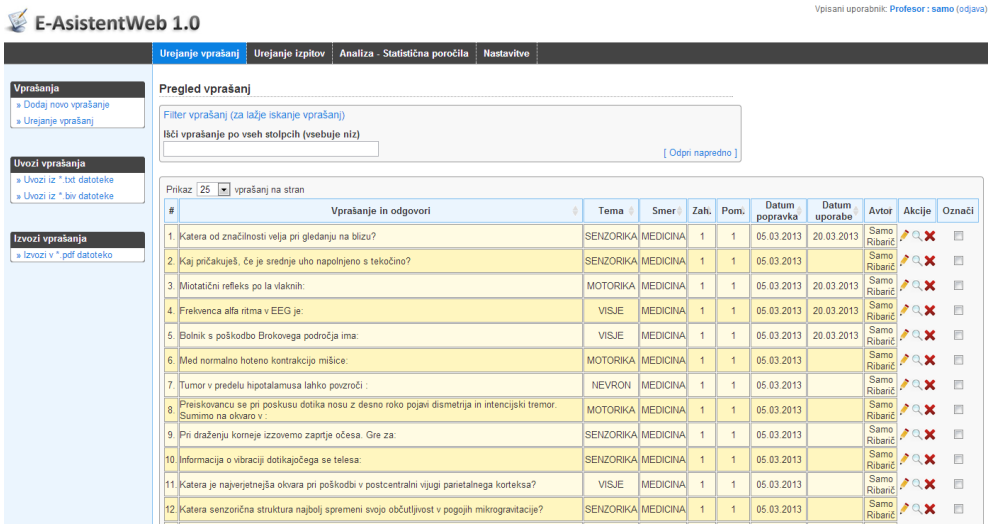
Kontroler Vprašanja vsebuje akcije CRUD na izpitnih vprašanjih. Akcije CRUD vsebujejo še kontrolne funkcije, ki so podedovane iz osnovnega razreda *Controller*, ki sem jih opisal v 4.5.2. Te funkcije upoštevajo dovoljenja



Slika 4.3: Pogled *vpis/index.php*

in posredujejo samo tiste podatke, ki so dovoljeni trenutnemu uporabniku. Implementiran je tudi uvoz vprašanj preko datotek. Kontroler vsebuje naslednje akcije:

- **index:** Prikaže pogled *vprašanja/index.php*. Nahaja se na sliki 4.4.
- **podrobnosti(id):** Ob kliku na ikono *podrobnosti* (slika leče na sliki 4.4) se izvrši ta akcija. Posreduje se ji tudi parameter *id* vprašanja, ki ga je že izpisala akcija *index* v tabeli za vsako vprašanje. Preko modela *Vprašanje* dobimo določeno vprašanje, ki ga posredujemo pogledu *vprašanja/podrobnosti.php*, kjer prikažemo njegove lastnosti.
- **dodaj:** Akcija pokaže pogled *vprašanja/dodaj.php*. Nahaja se na sliki 4.5. Ob potrditvi formularja najprej preveri, ali so ji posredovani podatki preko metode POST. Preko modela *Vprašanje* preveri validacijo in ustvari nov objekt modela, kar posledično doda novo vprašanje v bazo. Akcijo sproži pritisk na gumb *Dodaj novo vprašanje*.
- **uredi(id):** Akcija se sproži ob kliku na ikono *uredi* (svinčnik na sliki 4.4). Akcija prejme parameter *id* vprašanja, preko katerega dobimo določeno vprašanje iz baze. Podobno kot pri akciji *dodaj* se ob potrditvi formularja preveri, ali so posredovani podatki preko metode POST. Preveri se sama validacija in izvede posodobitev vprašanja.



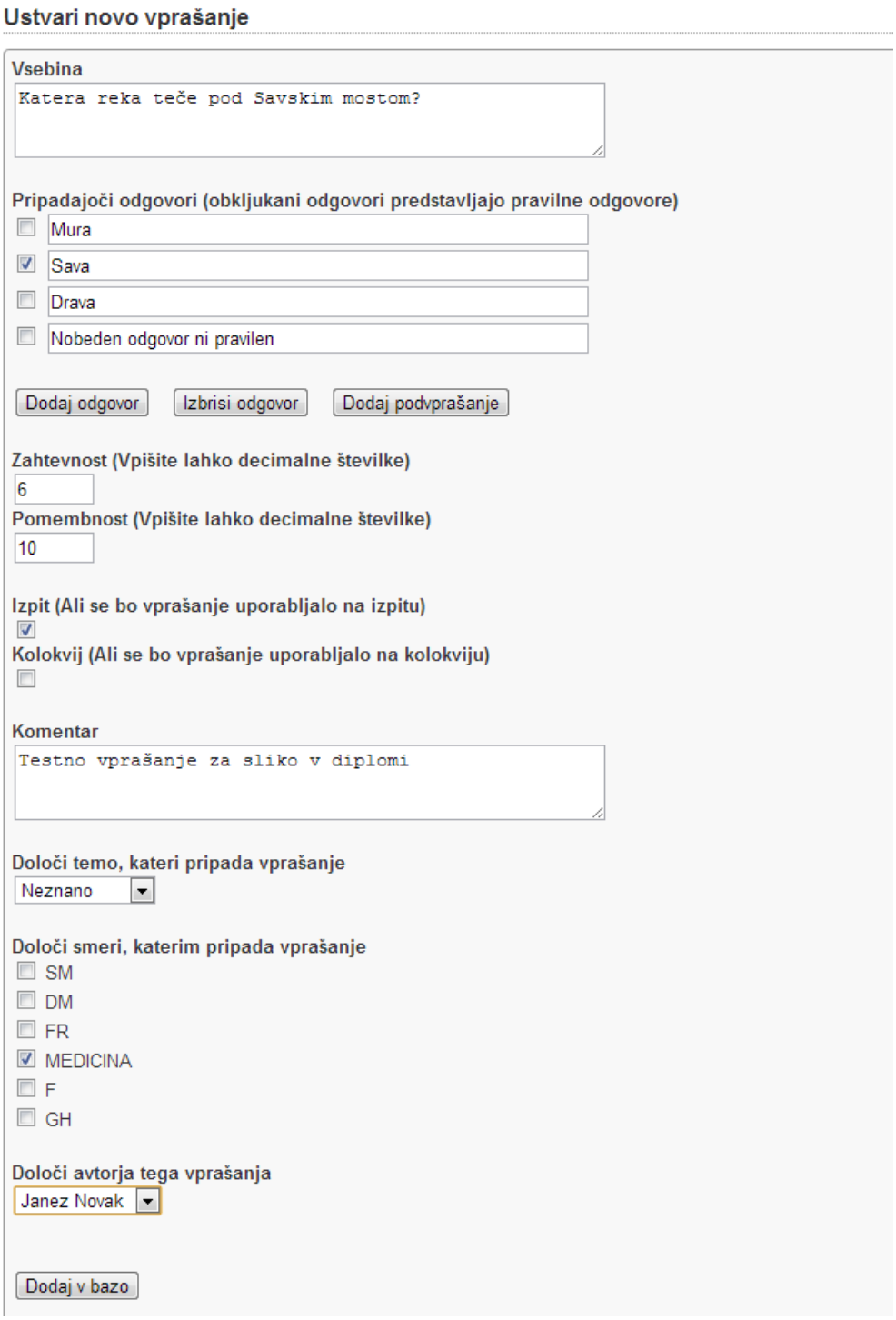
Slika 4.4: Pogled *vprašanja/index.php*

- **zbriši(id):** Akcija se sproži ob kliku na ikono *izbriši* (rdeči križec na sliki 4.4). Akcija prejme parameter *id* vprašanja, ki ga izbrišemo iz baze.

- **zbriši več:** Akcija se sproži ob kliku na gumb *Izbriši označena vprašanja* pod tabelo. Preko metode POST pošljemo *id*-je obkljukanih polj, ki so enaki istoležnim identitetam vprašanj. Izbrišejo se vsa vprašanja, ki jih paremetri vsebujejo.

- **pokaži(id):** Akcija se sproži, ko uporabnik klikne na vsebino vprašanja v tabeli, in pojavijo se pripadajoči odgovori tega vprašanja. Parameter *id* vprašanja se pošlje preko metode POST z uporabo tehnologije AJAX. Akcija vrne niz kode v HTML z odgovori, zapisanimi v HTML znački DIV, ki se doda pod vprašanjem tabele.

- **uvozi:** Akcija se sproži ob kliku na gumb *Uvozi iz *.txt* ali *Uvozi iz *.biv*. Akcija pokaže pogled *vprašanje/uvozi.php*. Ko uporabnik izpolne vnosna polja in potrdi formular, akcija preko funkcij PHP prebere datoteko, shranjeno v tabeli $_FILES. Prepozna ustrezen format datoteke



Slika 4.5: Pogled *vprašanja/dodaj.php*

in prek modela *Vprašanje* ustvarja objekte tega modela za vsako vrstico v datoteki, ki predstavlja vprašanje.

- **vprašanjaDataTable:** Akcijo sproži objekt *DataTable*, ki izhaja iz zunanje knjižnice *jQuery DataTables*. Ob vsaki osvežitivi ali akciji na tabeli na sliki 4.4 se kliče ta akcija preko metode POST z uporabo tehnologije AJAX. Izvede se poizvedba po vprašanjih glede na dovoljenja trenutnega uporabnika s pomočjo podedovanih funkcij osnovnega razreda *Controller*. Akcija vrne izpis vprašanj v formatu, imenovanem JSON, ki ga tabela DataTable dekodira v primeren izpis.

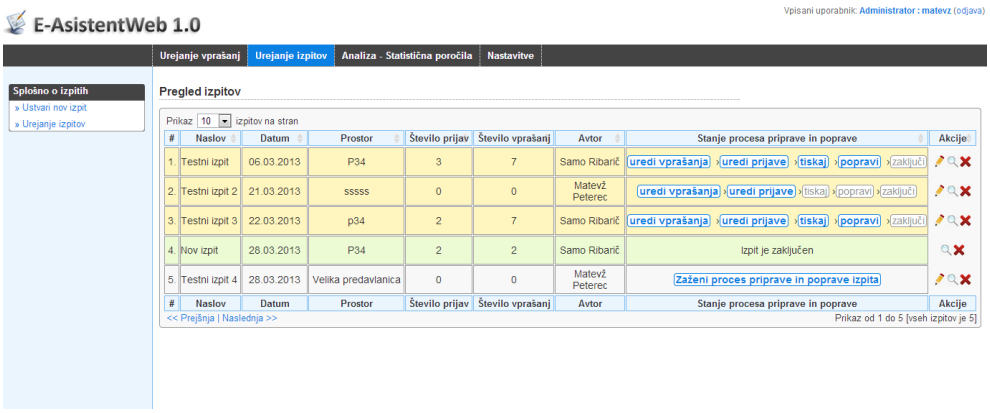
4.4.3 Priprava

Kontroler skrbi za akcije CRUD na izpitih in za pripravo izpitov, kot so vnosi vprašanj v izpit, prijava in odjava kandidatov na izpit in tiskanje izpitnih pol. Tudi tukaj je upoštevanje dovoljenj uporabnika za akcije realizirano z uporabo podedovanih funkcij osnovnega razreda *Controller*. Vsebuje naslednje akcije:

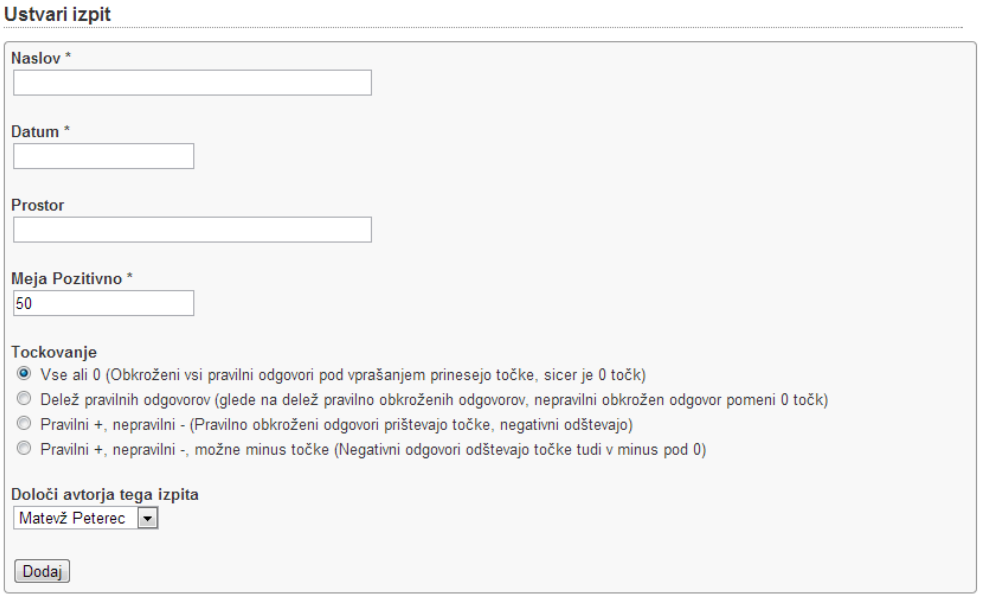
- **index:** Akcija se izvrši, ko obiščemo naslov *e-Asistent-web.com/index.php/priprava*. Naloži model *Izpit*, preko katerega dobimo vse izpite (objekti modela), ki so uporabniku dovoljeni. Izpite pošlje pogledu *priprava/index.php* in ga izpiše. Pogled je predstavljen na sliki 4.6.

- **dodaj:** Akcija se sproži, ko pritisnemo na gumb *Ustvari nov izpit*. Pokaže pogled *priprava/dodaj.php*, ki prikaže formular, kot je na sliki 4.7. Ob potrditivi formularja se preveri, ali so poslani izpolnjeni podatki preko metode POST. Ustvari nov objekt modela *Izpit* z izpolnjenimi podatki in ga shrani. Uporabnika preusmeri v pogled akcije (index).

- **podrobnosti(id):** Akcija se izvrši, ko pritisnemo na ikono *podrobnosti izpita* (leča na sliki 4.6). Naloži objekt modela *Izpit* preko prejetega parametra *id* izpita. Objekt modela se pošlje pogledu *priprava/podrobnosti.php*, ki prikaže vse lastnosti izpita uporabniku.



Slika 4.6: Pogled *priprava/index.php* predstavlja tabelo izpitov, kjer ima vsak izpit akcije, kot so *uredi izpit, podrobnosti izpita, izbriši izpit.* V predzadnjem stolpcu tabele je predstavljeno stanje procesa priprave in poprave izpita, ki kaže, v katerem stanju je izpit. S kliki na modro označena stanja lahko vstopamo v stanja procesa priprave in poprave.



Slika 4.7: Pogled *priprava/dodaj.php*

- **uredi(id):** Akcija se izvrši, ko pritisnemo na ikono *uredi izpit* (svinčnik na sliki 4.6). Pokaže pogled *priprava/uredi.php*, podobno kot je na sliki 4.7. Pogledu pošlje še objekt modela *Izpit*, ki ga naloži preko prejetega parametra *id*. Ob potrditvi formularja se preveri, ali so posredovani podatki preko metode POST, in izpit posodobimo. Validacija se izvrši preko modela *Izpit*.

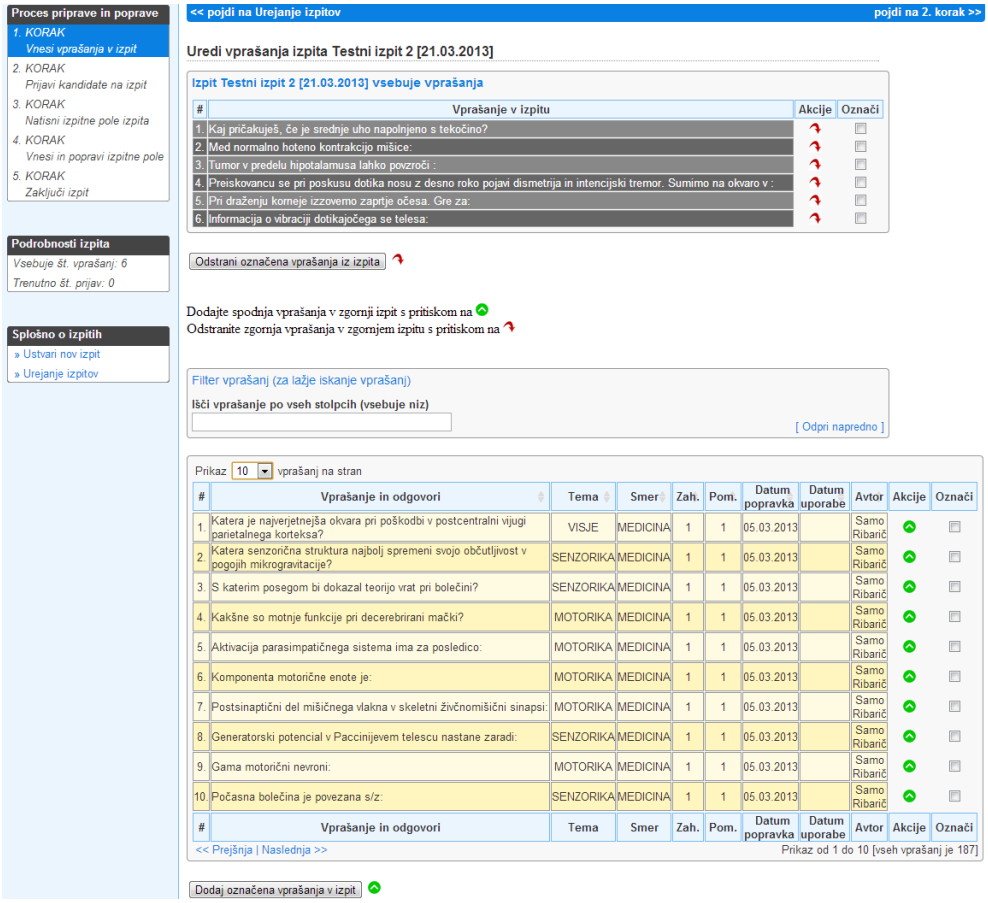
- **zbriši(id):** Akcija se izvrši, ko pritisnemo na ikono *izbriši izpit* (križec na sliki 4.6). Naloži objekt modela *Izpit* preko prejetega parametra *id* in sproži funkcijo *delete*, ki jo definira razred AR.

- **urediVprašanja(id):** Akcija se izvrši ob kliku na *Zaženi proces priprave in poprave izpita* ali na *Uredi vprašanja* v tabeli izpitov na sliki 4.6. Naloži objekt modela *Izpit* glede na podani parameter *id*, ki ga poda pogledu *priprava/uredi_vprasanja.php*. Pogled prikaže vprašanja, ki jih ta izpit vsebuje. V tem pogledu lahko uporabnik tudi dodaja in odstranjuje vprašanja iz izpita. Na sliki 4.8 je predstavljen pogled urejanja vprašanj v izpitu.

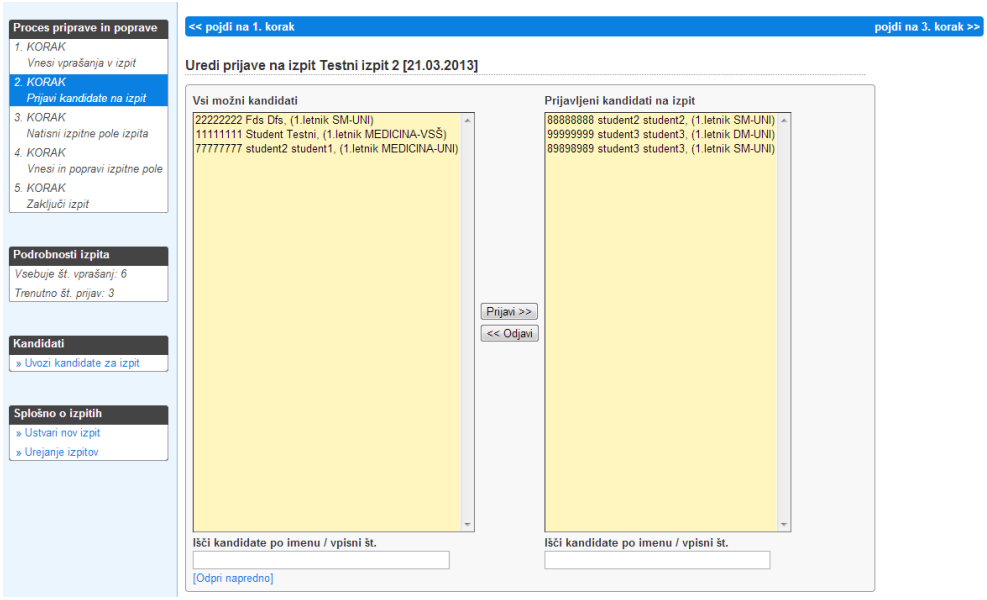
- **dodajVprIzpitu:** Akcija se izvrši ob kliku na ikono *dodaj vprašanje v izpit* (zelena puščica na sliki 4.8). Preko metode GET se pošlje *id* vprašanja v vrstici pritisnjene zelene puščice in *id* izpita. Akcija izvede ukaz SQL, ki vstavi v vmesno tabelo *vprasanje_tbl_has_izpit_tbl* oba *id*-ja in ponovno naloži pogled.

- **odstraniVprIzpitu:** Akcija se izvrši ob kliku na ikono *odstrani vprašanje iz izpita* (rdeča puščica na sliki 4.8). Podobno kot akcija **dodajVprIzpitu** se tudi ta izvrši preko metode GET *id* vprašanja in *id* izpita, izvede se stavek SQL in izbrišemo vrstico v vmesni tabeli *vprasanje_tbl_has_izpit_tbl*.

- **dodajVprsIzpitu:** Akcija se izvrši ob kliku na gumb *Dodaj označena vprašanja v izpit*. Označena polja so poimenovana z *id*-jem vprašanja in *id*-jem izpita v spodnji tabeli vprašanj. Ob kliku na gumb se akciji



Slika 4.8: Pogled *priprava/uredi_vprasanja.php*. Spodnja tabela predstavlja vprašanja, ki so na voljo uporabniku. Zgornja tabela je tabela vprašanj, ki jih izpit že vsebuje. S puščicami se lahko vprašanja dodaja ali odstrani iz izpita.



Slika 4.9: Pogled *priprava/uredi_prijave.php*. Levi seznam predstavlja vse kandidate v sistemu, desni pa kandidate, prijavljene na določen izpit. Z gumboma *Prijavi* in *Odjavi* prijavljamo ali odjavljamo študente z izpita.

preko metode POST pošlje imena označenih polj, kjer se izvrši dodajanje *id*-jev v vmesno tabelo *vprasanje_tbl_has_izpit_tbl*.

- **odstraniVprsIzpitu:** Akcija se izvrši ob kliku na gumb *Odstrani označena vprašanja iz izpita*. Podobno kot pri **dodajVprsIzpitu** so označena polja poimenovana tudi tu. Akcija izbriše vrstice z *id*-ji iz vmesne tabele *vprasanje_tbl_has_izpit_tbl*.

- **urediPrijave(id):** Akcija se izvrši ob kliku na *pojdi na 2. korak*, ki je na sliki 4.8, ali na modro označeno stanje *uredi prijave* v tabeli, ki je na sliki 4.6. Pokaže pogled *priprava/uredi_prijave.php*, ki je predstavljen na sliki 4.9. Pokaže kandidate izpita, ki je določen preko poslanega parametra *id*.

- **prijavaOdjava(id):** Akcija se izvrši ob kliku na gumb *Prijavi* in gumb *Odjavi*. Celoten vmesnik je formular z dvema gumboma tipa *Submit*.

Glede na ustrezne podatke, poslane preko metode POST, se preveri, kateri gumb je akcijo sprožil. Pri kliku za prijavo se izvrši kreiranje novega objekta modela *Prijava*, v katerega se shrani *id* študenta in *id* izpita za vsako vrstico v desnem seznamu formularja. Vsak objekt shranimo, kar pomeni nov vnos v tabelo *prijava_tbl*. Pri kliku na *Odjavi* se označene prijave (akciji so posredovani *id*-ji prijav) v desnem seznamu izbrišejo iz tabele *prijava_tbl*. Ob vsakemu klicu akcije se pogled zopet osveži.

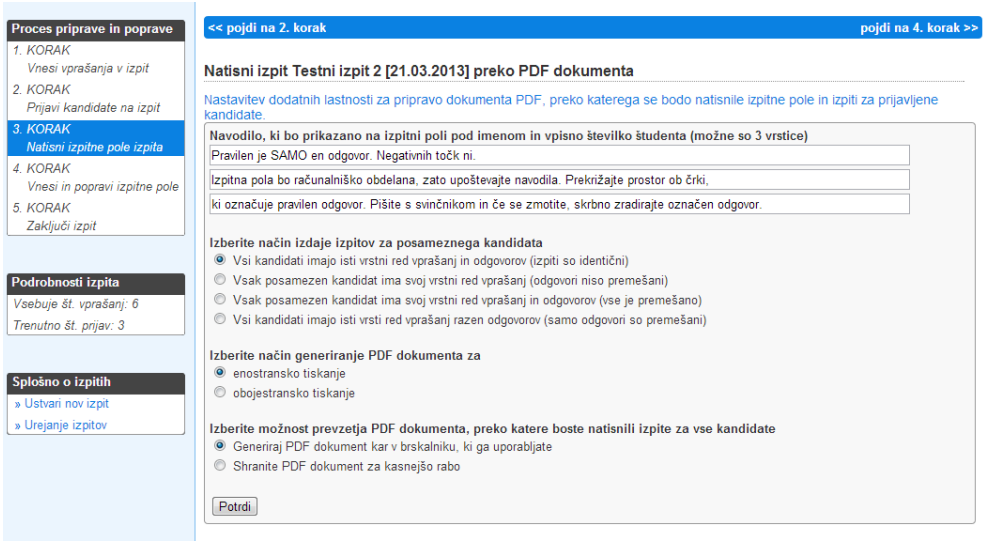
- **natisni(id):** Akcija se izvrši ob kliku na *pojdi na 3. korak* na sliki 4.9 ali na modro označeno stanje *natisni* v tabeli izpitov na sliki 4.6. Pošlje se še parameter *id* izpita, da akcija ve, za kateri izbit bo zgeneriran dokument PDF. Prikaže pogled *priprava/natisni.php*, ki je na sliki 4.10, kjer je potrebno izpolniti formular za kreiranje dokumenta PDF. Preko dokumenta PDF natisnemo izpite in izpitne pole za vsakega prijavljenega kandidata.

- **ustvariPDF(id):** Ob potrditvi formularja na sliki 4.10 se izvrši ta akcija. Z uporabo zunanje knjižnice FPDF se ustvari dokument PDF. Naloži se model *Prijava* in prebere se vse prijave v bazi. Preko prijav dobimo tudi pripadajoče objekte modelov *Izpit* in *Student* (to omogoča preslikovalnik ORM, ki jo Yii podpira). Te prijave pišemo v dokument s pomočjo funkcij omenjene knjižnice, tako barkodo (barkoda vsebuje *id* prijave) na izpitni poli kot samo vsebino pripadajočih vprašanj na izpitu. Primer izpitne pole je na sliki 4.11.

4.4.4 Poprava

Kontroler Poprava skrbi za popravo poskeniranih izpitnih pol in ocenjevanje preko javanskega vmesnika. Njegove akcije so naslednje:

- **popravi(id):** Akcija se izvrši ob kliku na *Pojdi na 4.korak*, ki je na sliki 4.10, ali pa na modro označeno stanje *popravi* v tabeli izpitov



Slika 4.10: Pogled *priprava/natisni.php*. Prva tri polja predstavljajo tri vrstična navodila na izpitni poli. Določimo lahko, ali bo možno mešanje vprašanj in odgovorov za vsakega kandidata posebej. Izberemo lahko način tiskanja, da aplikacija zna upoštevati način generiranje dokumenta PDF. Na koncu še izberemo, kako bomo dokument prevzeli.

student2 student2
88888888 **Podpis:**

00000032

Izpit: Testni izpit 2

Pravilen je SAMO en odgovor. Negativnih točk ni.
Izpitna pola bo računalniško obdelana, zato upoštevajte navodila. Prekrižajte prostor ob črki, ki označuje pravilen odgovor. Pišite s svinčnikom in če se zmotite, skrbno zradirajte označen odgovor.

1	a	b	c	d	e	f	g	h	i	j
2	a	b	c	d	e	f	g	h	i	j
3	a	b	c	d	e	f	g	h	i	j
4	a	b	c	d	e	f	g	h	i	j
5	a	b	c	d	e	f	g	h	i	j
6	a	b	c	d	e	f	g	h	i	j
7	a	b	c	d	e	f	g	h	i	j
8	a	b	c	d	e	f	g	h	i	j
9	a	b	c	d	e	f	g	h	i	j
10	a	b	c	d	e	f	g	h	i	j
11	a	b	c	d	e	f	g	h	i	j
12	a	b	c	d	e	f	g	h	i	j
13	a	b	c	d	e	f	g	h	i	j
14	a	b	c	d	e	f	g	h	i	j
15	a	b	c	d	e	f	g	h	i	j
16	a	b	c	d	e	f	g	h	i	j
17	a	b	c	d	e	f	g	h	i	j
18	a	b	c	d	e	f	g	h	i	j
19	a	b	c	d	e	f	g	h	i	j
20	a	b	c	d	e	f	g	h	i	j
21	a	b	c	d	e	f	g	h	i	j
22	a	b	c	d	e	f	g	h	i	j
23	a	b	c	d	e	f	g	h	i	j
24	a	b	c	d	e	f	g	h	i	j
25	a	b	c	d	e	f	g	h	i	j
26	a	b	c	d	e	f	g	h	i	j
27	a	b	c	d	e	f	g	h	i	j
28	a	b	c	d	e	f	g	h	i	j
29	a	b	c	d	e	f	g	h	i	j
30	a	b	c	d	e	f	g	h	i	j
31	a	b	c	d	e	f	g	h	i	j
32	a	b	c	d	e	f	g	h	i	j
33	a	b	c	d	e	f	g	h	i	j
34	a	b	c	d	e	f	g	h	i	j
35	a	b	c	d	e	f	g	h	i	j
36	a	b	c	d	e	f	g	h	i	j
37	a	b	c	d	e	f	g	h	i	j
38	a	b	c	d	e	f	g	h	i	j
39	a	b	c	d	e	f	g	h	i	j
40	a	b	c	d	e	f	g	h	i	j
41	a	b	c	d	e	f	g	h	i	j
42	a	b	c	d	e	f	g	h	i	j
43	a	b	c	d	e	f	g	h	i	j
44	a	b	c	d	e	f	g	h	i	j
45	a	b	c	d	e	f	g	h	i	j
46	a	b	c	d	e	f	g	h	i	j
47	a	b	c	d	e	f	g	h	i	j
48	a	b	c	d	e	f	g	h	i	j
49	a	b	c	d	e	f	g	h	i	j
50	a	b	c	d	e	f	g	h	i	j

51	a	b	c	d	e	f	g	h	i	j
52	a	b	c	d	e	f	g	h	i	j
53	a	b	c	d	e	f	g	h	i	j
54	a	b	c	d	e	f	g	h	i	j
55	a	b	c	d	e	f	g	h	i	j
56	a	b	c	d	e	f	g	h	i	j
57	a	b	c	d	e	f	g	h	i	j
58	a	b	c	d	e	f	g	h	i	j
59	a	b	c	d	e	f	g	h	i	j
60	a	b	c	d	e	f	g	h	i	j
61	a	b	c	d	e	f	g	h	i	j
62	a	b	c	d	e	f	g	h	i	j
63	a	b	c	d	e	f	g	h	i	j
64	a	b	c	d	e	f	g	h	i	j
65	a	b	c	d	e	f	g	h	i	j
66	a	b	c	d	e	f	g	h	i	j
67	a	b	c	d	e	f	g	h	i	j
68	a	b	c	d	e	f	g	h	i	j
69	a	b	c	d	e	f	g	h	i	j
70	a	b	c	d	e	f	g	h	i	j
71	a	b	c	d	e	f	g	h	i	j
72	a	b	c	d	e	f	g	h	i	j
73	a	b	c	d	e	f	g	h	i	j
74	a	b	c	d	e	f	g	h	i	j
75	a	b	c	d	e	f	g	h	i	j
76	a	b	c	d	e	f	g	h	i	j
77	a	b	c	d	e	f	g	h	i	j
78	a	b	c	d	e	f	g	h	i	j
79	a	b	c	d	e	f	g	h	i	j
80	a	b	c	d	e	f	g	h	i	j
81	a	b	c	d	e	f	g	h	i	j
82	a	b	c	d	e	f	g	h	i	j
83	a	b	c	d	e	f	g	h	i	j
84	a	b	c	d	e	f	g	h	i	j
85	a	b	c	d	e	f	g	h	i	j
86	a	b	c	d	e	f	g	h	i	j
87	a	b	c	d	e	f	g	h	i	j
88	a	b	c	d	e	f	g	h	i	j
89	a	b	c	d	e	f	g	h	i	j
90	a	b	c	d	e	f	g	h	i	j
91	a	b	c	d	e	f	g	h	i	j
92	a	b	c	d	e	f	g	h	i	j
93	a	b	c	d	e	f	g	h	i	j
94	a	b	c	d	e	f	g	h	i	j
95	a	b	c	d	e	f	g	h	i	j
96	a	b	c	d	e	f	g	h	i	j
97	a	b	c	d	e	f	g	h	i	j
98	a	b	c	d	e	f	g	h	i	j
99	a	b	c	d	e	f	g	h	i	j
100	a	b	c	d	e	f	g	h	i	j

e-AsistentWeb ver.:1.0.0, Up.: Inštitut Matevž Petercčč, Neka fakulteta, Univerza v Ljubljani 14.11.2012 9:20:20

Slika 4.11: Primer izpitne pole, ki jo prejmejo kandidati poleg izpita.

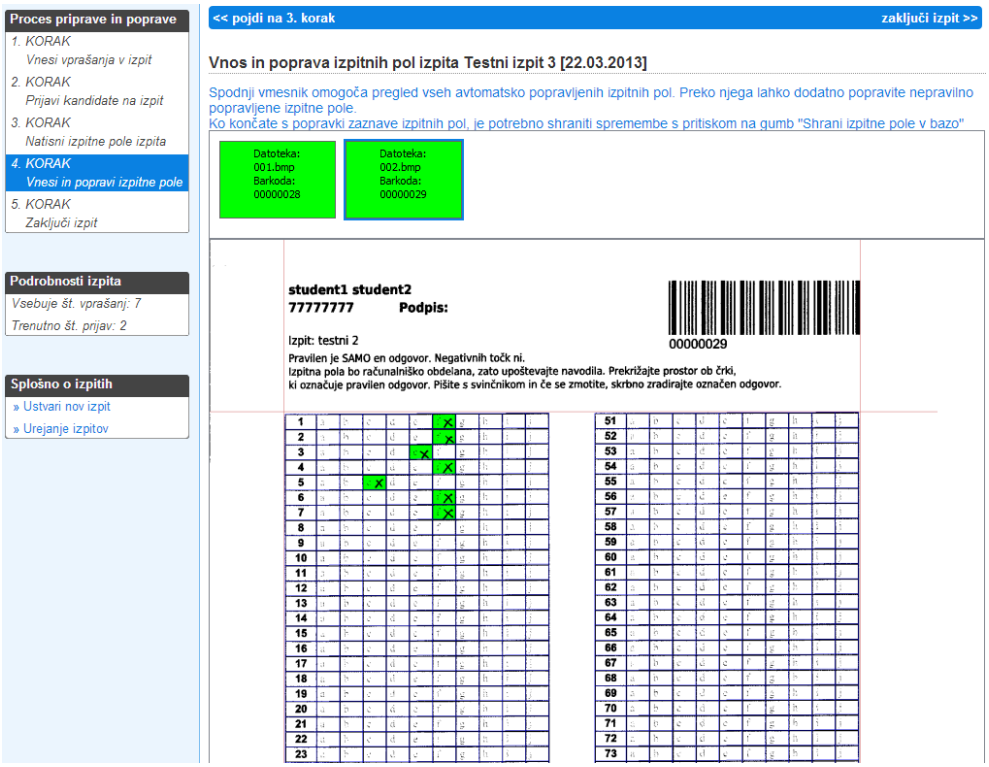
na sliki 4.6. Preko poslanega parametra *id* izpita preveri, ali je izpit v stanju tiskanja. Če je, potem pokaže pogled *poprava/popravi.php*, kateremu pošlje tudi parametre izpita. Pogled vsebuje javanski aplet (angl. *Java applet*), ki je integrirana preko HTML značke APPLET. Naloži se na uporabniški računalnik in služi kot vmesnik za razpoznavo optično prebranih izpitnih pol v obliki 1-bitnih slik formata BMP. Preko njega uporabnik izbere direktorij, kamor je shranil slike izpitnih pol, in vmesnik jih prebere. Ob branju vsako sliko procesira in zaznava obkljukane križce v tabeli izpitne pole (pola je predstavljena na sliki 4.11). Ko konča s procesom zaznavanja, nudi uporabniku tudi "ročno"popravljanje po posameznih izpitnih polah. Vmesnik je predstavljen na sliki 4.12. Podrobneje o implementaciji vmesnika pišem v 4.5.1.

- **oceniIzpit:** Akcijo izvršuje javanski vmesnik ob kliku na *Shrani popravljene izpitne pole*. Gre skozi svoj seznam popravljenih izpitov (v seznamu ima hranjene obkljukane križce) in jih preko metode POST pošilja akciji, ki dobi označene križce in barkodo za vsak izpit posebej. Preko barkode najde ustrezno prijavo v podatkovni bazi in jo posodobi z nizom označenih odgovorov in izračunom točk. Ko konča, pošlje vmesniku odziv, da je akcija končana in je pripravljena na naslednji popravljen izpit. Komunikacija med njima poteka na asinhron način.

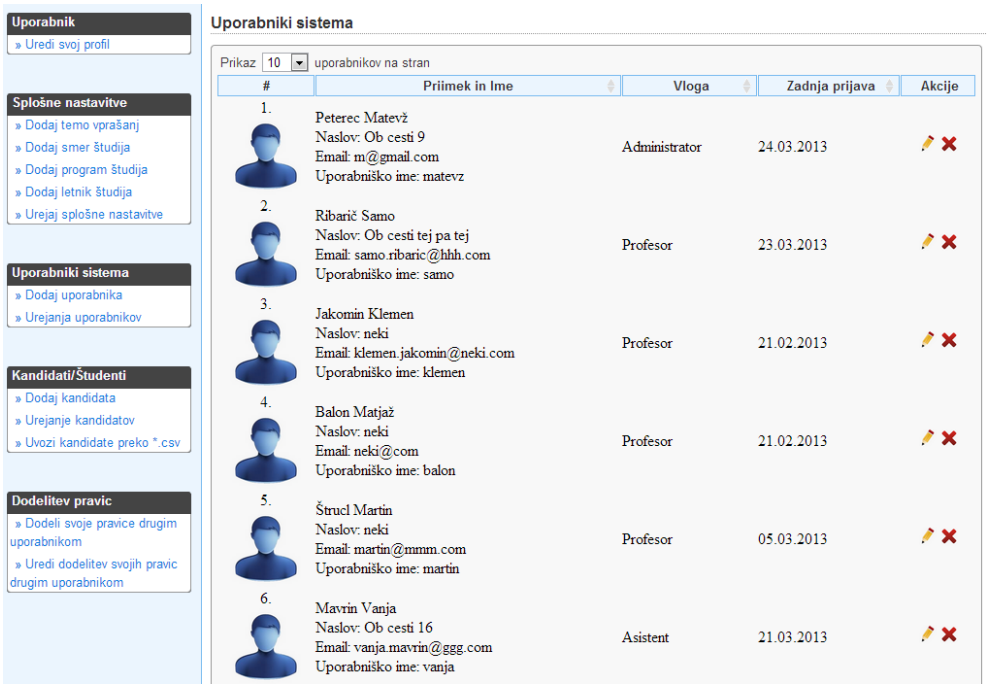
- **zakljuciIzpit(id):** Akcija se izvrši ob kliku na *Zaključi izpit* na sliki 4.12. Atribut objekta modela *Izpit*, ki je najden preko posredovanega parametra *id*, nastavi na stanje zaključenega izpita. Uporabnika preusmeri na pogled *priprava/index.php*. Zaključenega izpita ni več mogoče urejati. Primer zaključenega izpita je na sliki 4.6.

4.4.5 Uporabniki

Kontroler skrbi za akcije CRUD na uporabnikih sistema. Dostop do njegovih akcij je dovoljen le administratorju, kar je realizirano s preverjanjem vloge



Slika 4.12: Primer uporabe javanskega vmesnika v procesu poprave. V zgornjem delu vmesnika so v zelenih okvirjih označene izpitne pole, ki so bile zaznane brez težav. Napačne zaznane pole bi bile obdane z rdečimi okvirji.

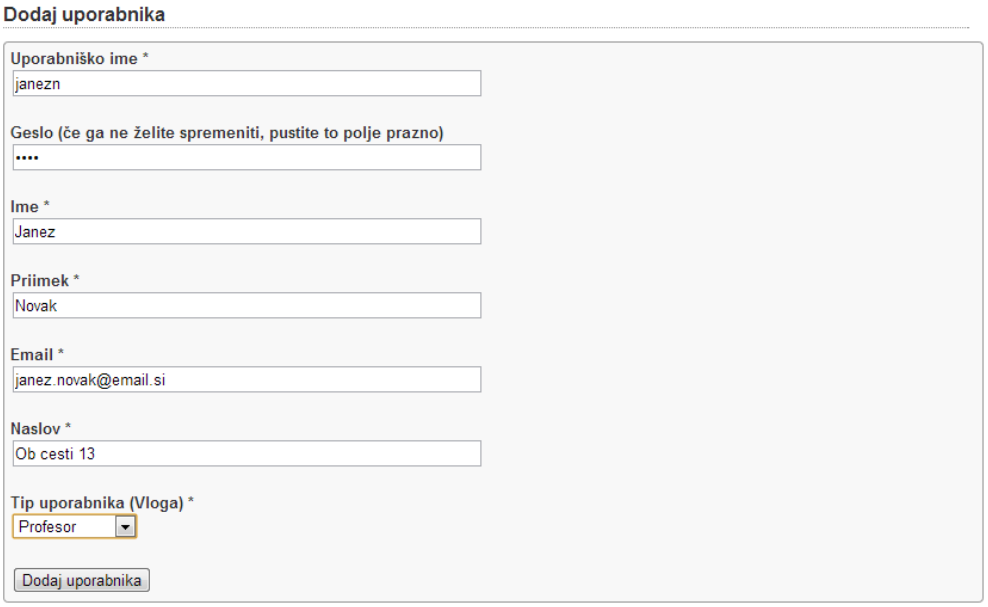


Slika 4.13: Pogled *uporabniki/index.php*. Predstavlja tabelo uporabnikov sistema in viden le administratorju.

trenutno vpisanega uporabnika preko sejne spremenljivke. Akcije so:

- **index:** Preveri se vloga trenutno vpisanega uporabnika v sejni spremenljivki. Če vloga uporabnika ni administrator, uporabnika preusmerina naslov *nastavitve/index.php*. Pošlje se model *Uporabnik* pogledu *uporabniki/index.php*. Pogled predstavi tabelo uporabnikov v sistemu, prikazana je na sliki 4.13.

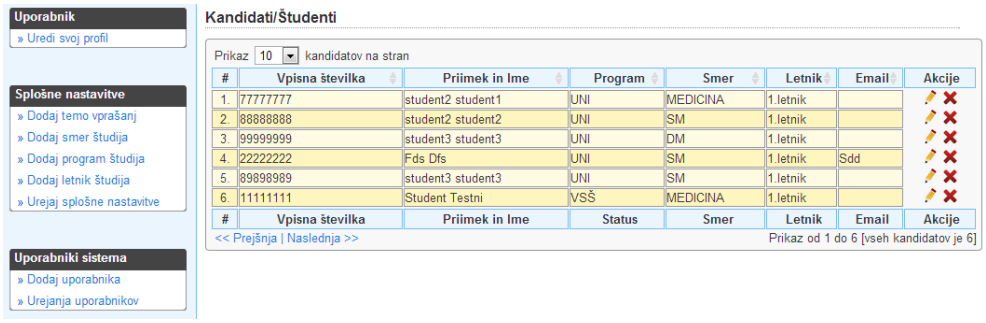
- **dodaj:** Akcija se izvrši ob kliku na *Dodaj uporabnika*. Pokaže se pogled *uporabniki/dodaj.php*, ki vsebuje formular, kot je na sliki 4.14. Če so akciji posredovani izpolnjeni podatki formularja preko metode POST, se izvrši kreiranje objekta modela *Uporabnik* in klicanje funkcije *save*. V tabelo *uporabnik_tbl* se shrani nov uporabnik. Validacija je implementirana preko modela *Uporabnik*.



Slika 4.14: Pogled *uporabniki/dodaj.php*. Vsebuje formular za dodajanje novih uporabnikov.

- **uredi(id):** Akcija se izvrši ob kliku na ikono *uredi uporabnika* (slika svinčnika na sliki 4.13), ki predstavlja povezavo na naslov *uporabniki/uredi/(določen id uporabnika)*. Pojavi se enak formular kot na sliki 4.14, ko se pokaže pogled *uporabniki/uredi.php*. Objekt modela *Uporabnik* dobi preko posredovanega *id* parametra in pogledu posreduje uporabniške lastnosti, ki jih zapolni v vnosna polja formularja. Ob potrditvi se izvršijo operacije enako kot pri akciji **dodaj**, le da se uporabnik posodobi.

- **zbrisi(id):** Akcija se izvrši ob kliku na ikono *izbriši uporabnika*. Preko posredovanega parametra *id* izbrišemo uporabnika iz tabele *uporabnik_tbl*. Dobi objekt modela *Uporabnik* in kliče funkcijo *delete*.



Slika 4.15: Pogled *kandidati/index.php*. Vsebuje tabelo vseh kandidatov v sistemu, kjer so mogoče akcije urejanja in brisanja kandidatov.

4.4.6 Kandidati

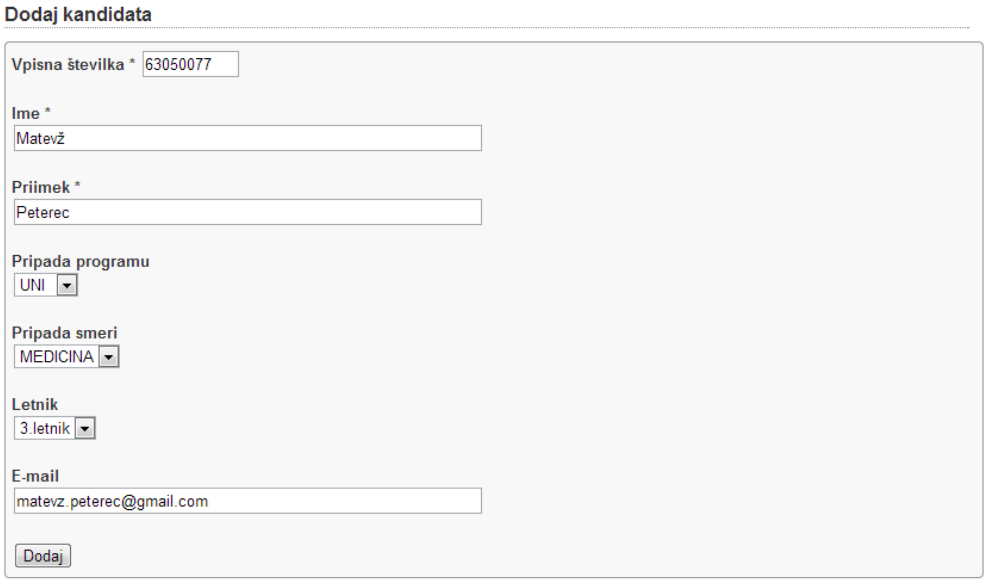
Kontroler skrbi za akcije CRUD na kandidatih oz. študentih. Omogoča tudi uvoz kandidatov prek datoteke CSV.

- **index:** Akcija se izvrši ob kliku na *Urejanje kandidatov* v levem podmeniju, ko se nahajamo v nastavitvah. Naloži se model *Kandidat*, preko katerega dobimo vse kandidate iz tabele *student_tbl*, in se pošlje pogledu *kandidati/index.php*, ki predstavi tabelo vseh kandidatov. Ta tabela je na sliki 4.15.

- **dodaj:** Akcija se izvrši ob kliku na *dodaj kandidata* v levem podmeniju na sliki 4.15. Pojavi se formular, ki ga vsebuje pogled *kandidati/dodaj.php* na sliki 4.16. Dodajanje je realizirano na enak način kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija dodaj**).

- **uredi(id):** Akcija se izvrši ob kliku na ikono *uredi kandidata* (slika svinčnika na sliki 4.15). Urejanje je realizirano enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija uredi**).

- **zbrisi(id):** Akcija se izvrši ob kliku na ikono *izbriši kandidata* (križec na sliki 4.15). Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija izbriši**).



Slika 4.16: Pogled *kandidati/dodaj.php*. Vsebuje formular, preko katerega dodajamo nove kandidate.

- **uvoziCSV:** Akcija se izvrši ob kliku na *Uvozi kandidate preko *.csv* v levem podmeniju na sliki 4.15. Prikaže se pogled *kandidati/uvozi.php*, ki vsebuje formular. Prikazan je na sliki 4.17. Ob potrditvi formularja preveri, ali je nastavljena tabela $_FILES, ki vsebuje naloženo datoteko CSV v začasnem prostoru. Za vsako prebrano vrstico v datoteki kreira objekte modela *Kandidat* z vsemi potrebnimi lastnostmi in jih shranjuje. Po končanem uvozu se izpiše niz uspešnih in neuspešnih uvozov, ki se po končani akciji prikaže uporabniku z dodanimi parametri nizov *success* in *error*. Format datoteke je:

```
vpisna;priimek;ime;email;smer;letnik;program
99060199;Janez;Novak;jn@gmaill.com;Medicina;3.letnik;uni
99060152;Jure;Kovač;temp@gmaill.com;Medicina;3.letnik;uni
...
```

Slika 4.17: Pogled *kandidati/uvozi.php*. Vsebuje formular, kjer naložimo datoteko CSV kandidatov. Primer uspešnega uvoza kandidatov.

4.4.7 Dovoljenja

Kontroler skrbi za akcije CRUD na dovoljenjih. Vsak Administrator ali Profesor lahko ustvari dovoljenje. Profesor lahko dovoli branje, urejanje, brisanje in ustvarjanje svojih vprašanj, izpitov ali procesov priprave in poprave izpitov drugim uporabnikom. Administrator lahko ustvarja dovoljenju v imenu katerega koli uporabnika in jih dodeli drugim uporabnikom.

- **index:** Akcija se izvrši ob kliku na *Uredi dodelitev svojih pravic drugim uporabnikom* v levem podmeniju, ko se nahajamo v nastavitvah. Pošlje model *Dovoljenje* pogledu *dovoljenja/index.php*, preko katerega dobimo vsa dovoljenja, ki jih je uporabnik ustvaril. Preveri se vloga uporabnika. Za vlogo Profesor se pošljejo le njegova dovoljenja. Administratorju se prikažejo vsa dovoljenja. Na sliki 4.18 je predstavljena tabela dovoljenj.
- **dodaj:** Akcija se izvrši ob kliku na *Dodeli svoje pravice drugim uporabnikom* v levem podmeniju, ko se nahajamo v nastavitvah. Prikaže pogled *dovoljenja/dodaj.php*, ki vsebuje formular z meniji, kot na sliki 4.19.

Slika 4.18: Pogled *dovoljenjai/index.php*

Preveri se vloga vpisanega uporabnika. Za vlogo Profesor se izbira uporabnikov, ki dovoljujejo drugim, nastavi le na eno izbiro, ki je ta Profesor sam. Pri vlogi Administratorja je možna izbira med vsemi uporabniki. Ko formular potrdimo, se ustvari objekt modela *Dovoljenje*, ki se shrani in s tem zapiše dovoljenje v tabelo *dovoljenja_tbl*.

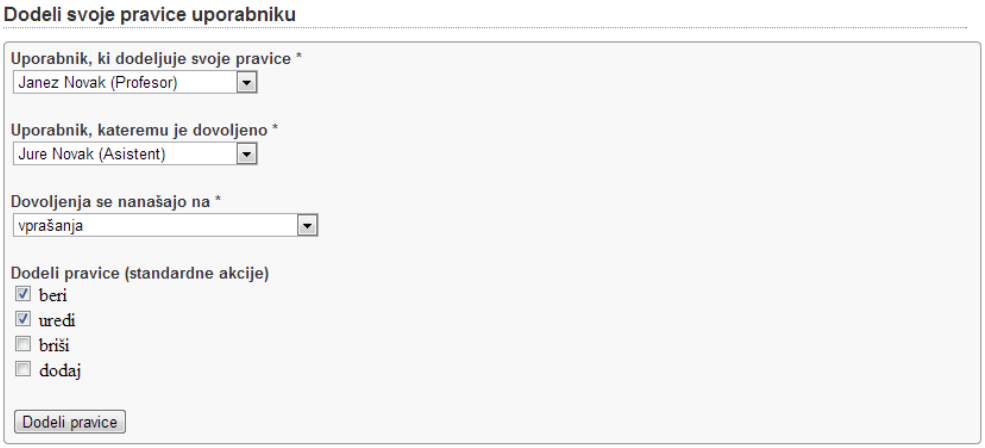
- **uredi(id):** Akcija se izvrši ob kliku na ikono *uredi dovoljenje* (slika svinčnika na sliki 4.18). Urejanje je realizirano na enak način kot pri kontrolerju uporabniki (glej **Uporabniki, akcija uredi**).

- **zbrisi(id):** Akcija se izvrši ob kliku na ikono *izbriši dovoljenje* (slika križca na sliki 4.18). Brisanje je realizirano kot pri kontrolerju uporabniki (glej **Uporabniki, akcija izbriši**).

4.4.8 Nastavitve

Kontroler skrbi za akcije CRUD na možnih temah vprašanj, smereh, programih in letnikih.

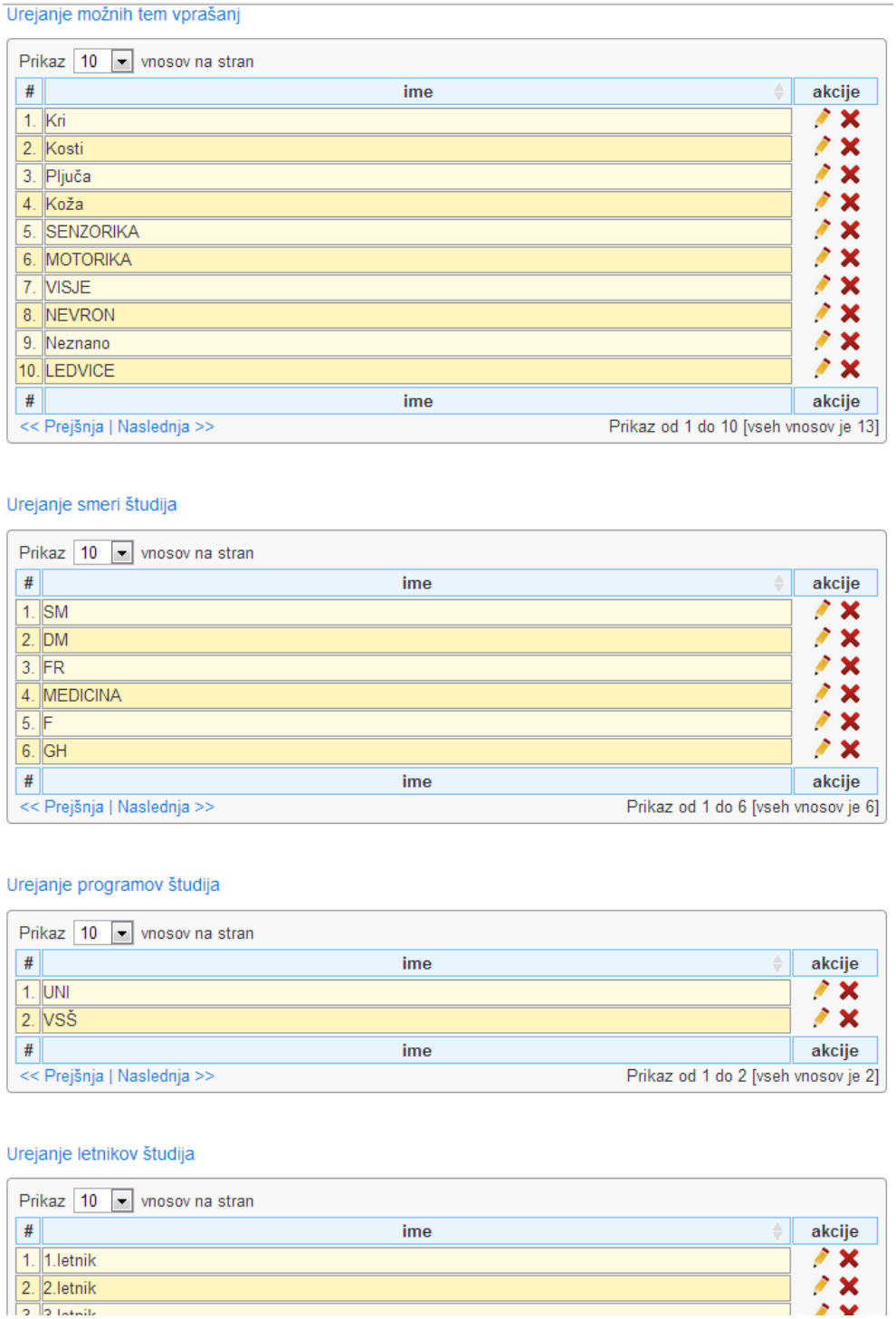
- **index:** Akcija se izvrši ob kliku na *Nastavitve* v glavnem meniju. Prikaže pogled *nastavitve/index.php*, ki vsebuje štiri tabele. Pogledu se pošljejo štiri modeli: *Tema*, *Smer*, *Program* in *Letnik*. Preko njih



Slika 4.19: Pogled *dovoljenja/dodaj.php*

se izvršijo poizvedbe v podatkovni bazi, ki se predstavijo v tabelah z akcijami *uredi* in *izbriši*, kot je prikazano na sliki 4.20.

- **dodajTemo:** Akcija se izvrši ob kliku na *Dodaj temo* v levem podmeniju, ko se nahajamo v nastavitvah. Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija dodaj**).
- **urediTemo(id):** Akcija se izvrši ob kliku na ikono *uredi temo* (slika svinčnika na sliki 4.20). Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija uredi**).
- **zbrisiTemo(id):** Akcija se izvrši ob kliku na ikono *izbriši temo* (slika križca na sliki 4.20). Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija izbriši**).
- **dodajSmer:** Akcija se izvrši ob kliku na *Dodaj smer* v levem podmeniju, ko se nahajamo v nastavitvah. Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija dodaj**).
- **urediSmer(id):** Akcija se izvrši ob kliku na ikono *uredi smer* (slika svinčnika na sliki 4.20). Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija uredi**).



Slika 4.20: Pogled *nastavitve/index.php*

- **zbrisiSmer(id):** Akcija se izvrši ob kliku na ikono *izbriši temo* (slika križca na sliki 4.20). Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija izbriši**).
- **dodajProgram:** Akcija se izvrši ob kliku na *Dodaj program* v levem podmeniju, ko se nahajamo v nastavitvah. Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija dodaj**).
- **urediProgram(id):** Akcija se izvrši ob kliku na ikono *uredi program* (slika svinčnika na sliki 4.20). Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija uredi**).
- **zbrisiProgram(id):** Akcija se izvrši ob kliku na ikono *izbriši temo* (slika križca na sliki 4.20). Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija izbriši**).
- **dodajLetnik:** Akcija se izvrši ob kliku na *Dodaj letnik* v levem podmeniju, ko se nahajamo v nastavitvahi. Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija dodaj**).
- **urediLetnik(id):** Akcija se izvrši ob kliku na ikono *uredi letnik* (slika svinčnika na sliki 4.20). Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija uredi**).
- **zbrisiLetnik(id):** Akcija se izvrši ob kliku na ikono *izbriši temo* (slika križca na sliki 4.20). Akcija se izvršuje enako kot pri kontrolerju Uporabniki (glej **Uporabniki, akcija izbriši**).

4.4.9 Analiza

Kontroler skrbi za prikaz in izračun statističnih podatkov uporabnikov sistema, posameznih kanditatov, vprašanj in izpitov.

- **index:** Akcija se izvrši ob kliku na *Analiza - Statistična poročila* v glavnem meniju. Preusmeri na akcijo **statistikaUpor** s parametrom *id*, ki ga dobi iz sejne spremenljivke trenutno vpisanega uporabnika.

- **uporabnik:** Akcija je enaka kot **index**.

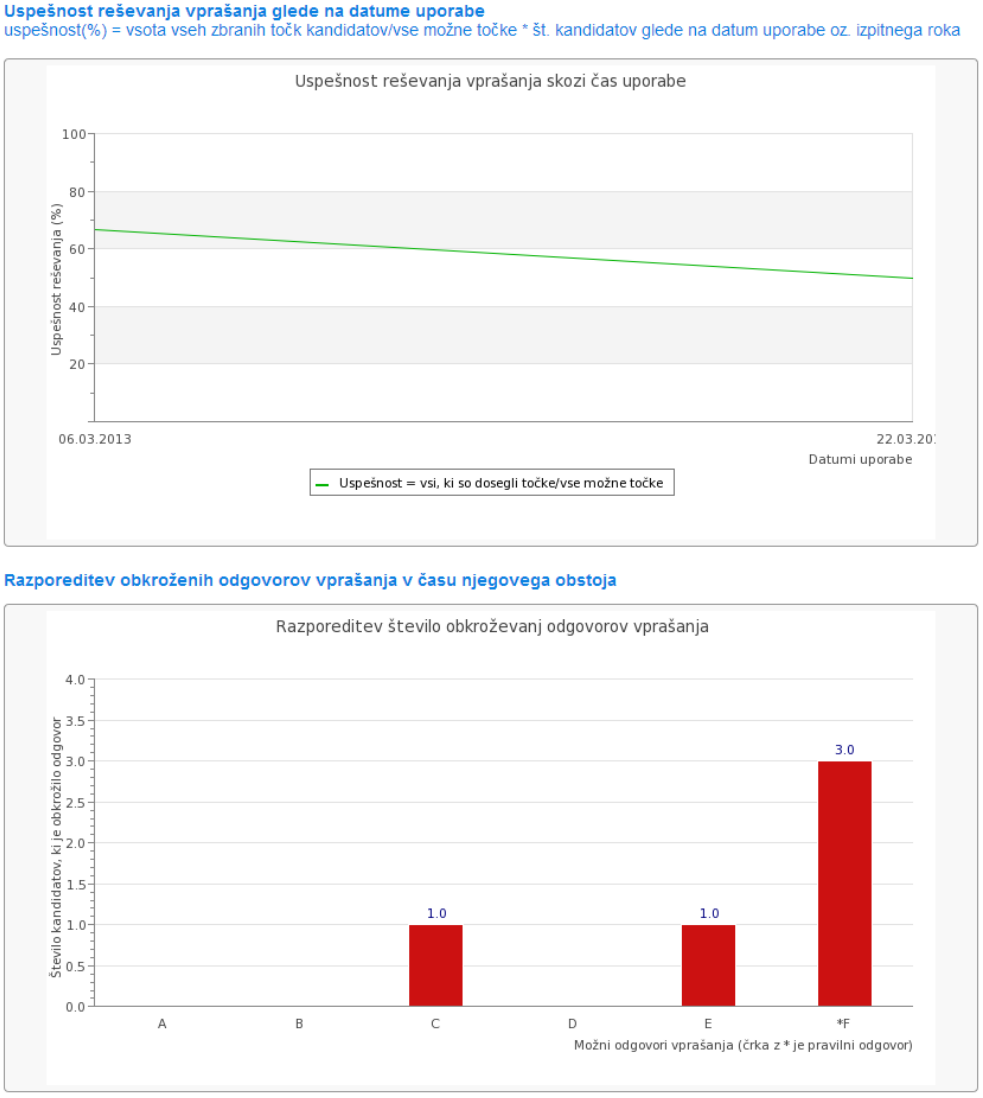
- **vprašanja:** Akcija se izvrši ob kliku na *Statistika vprašanja* v levem podmeniju, ko se nahajamo v analizi. Prikaže pogled *analiza/vprasanja.php*. Pogled vsebuje tabelo vprašanj, implementirano na enak način kot v kontrolerju vprašanja (glej **kontroler Vprašanja, akcija vprasanjaDataTable**). Ob vsaki osvežitvi tabele se kliče akcija **vprasanjaDataTable** preko tehnologije AJAX, ki vrne izpis vprašanj. Vprašanja imajo na razpolago le akcijo *statistični podatki vprašanja*, preko katere dostopamo do **statistikaVpr**.

- **izpiti:** Akcija se izvrši ob kliku na *Statistika vprašanja* v levem podmeniju, ko se nahajamo v analizi. Prikaže pogled *analiza/izpiti.php*. Pogled vsebuje tabelo izpitov, implementirano na enak način kot pri kontrolerju Priprava (glej **kontroler Priprava, akcija index**). Izpiti v tabeli imajo na voljo le akcijo *statistika izpita*.

- **kandidati:** Akcija se izvrši ob kliku na *Statistika kandidata* v levem podmeniju, ko se nahajamo v analizi. Prikaže pogled *analiza/kadidati.php*. Pogled vsebuje tabelo kandidatov, implementirano na enak način kot pri kontrolerju Kandidati (glej **kontroler Kandidati, akcija index**). Kandidati v tabeli imajo na voljo le akcijo *statistika kandidata*.

- **statistikaVpr(id):** Akcija se izvrši ob kliku na ikono *statistični podatki vprašanja* v tabeli vprašanj. Izvede potrebne statistične izračune za vprašanje, ki je določeno preko posredovanega parametra *id*. Z uporabo zunanje knjižnice *JpGraph* se narišejo grafi statistične obdelave in shranijo v slike PNG. Prikaže pogled *analiza/statistika_vpr.php*, ki vsebuje sliko grafov preko HTML značke IMG. Te grafe prikazuje slika 4.21.

- **statistikaIzp(id):** Akcija se izvrši ob kliku na ikono *statistični podatki izpita* v tabeli izpitov. Podobno kot pri akciji **statistikaVpr** se izvrši



Slika 4.21: Pogled *analiza/statistika_vpr.php*

akcija po enakih korakih, da ustvari potrebne statistične grafe. Prikaže pogled *analiza/statistika_izp.php*, ki vsebuje slike grafov. Slika 4.22 predstavlja grafe.

- **statistikaUpor(id):** Akcija se izvrši takoj, ko obiščemo akcijo **index** ali **uporabnik**. Izvede se izračun statističnih podatkov trenutnega uporabnika, graf je prikazan na sliki 4.23. Prikaže se pogled *analiza/statistika_upor.php*.

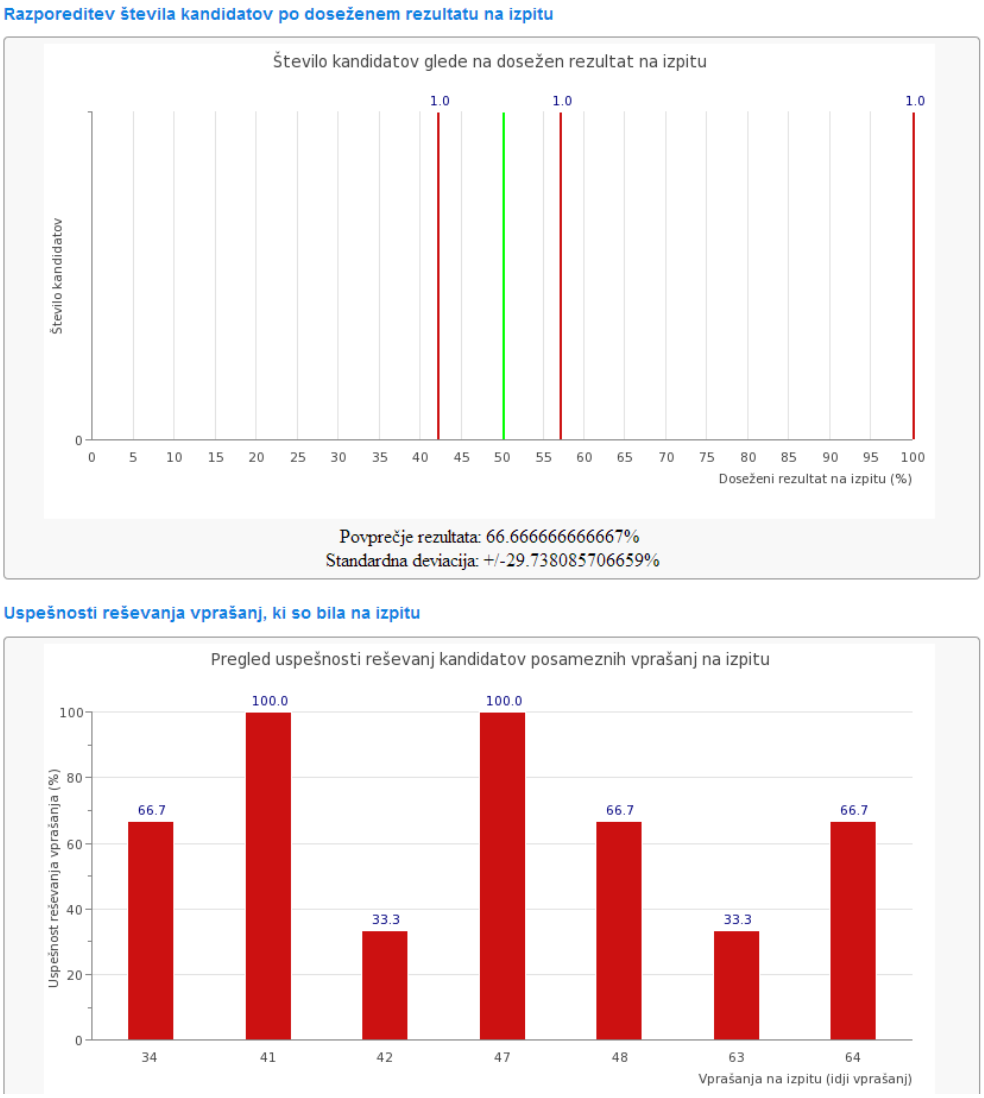
- **statistikaKand(id):** Akcija se izvrši ob kliku na ikono *statistični podatki kandidata* v tabeli kandidatov. Podobno kot pri akciji **statistikaVpr** se izvrši akcija po enakih korakih, da ustvari potrebne statistične grafe. Prikaže pogled *analiza/statistika_kand.php* s slikami grafov.

4.5 Ključni izzivi pri realizaciji aplikacije

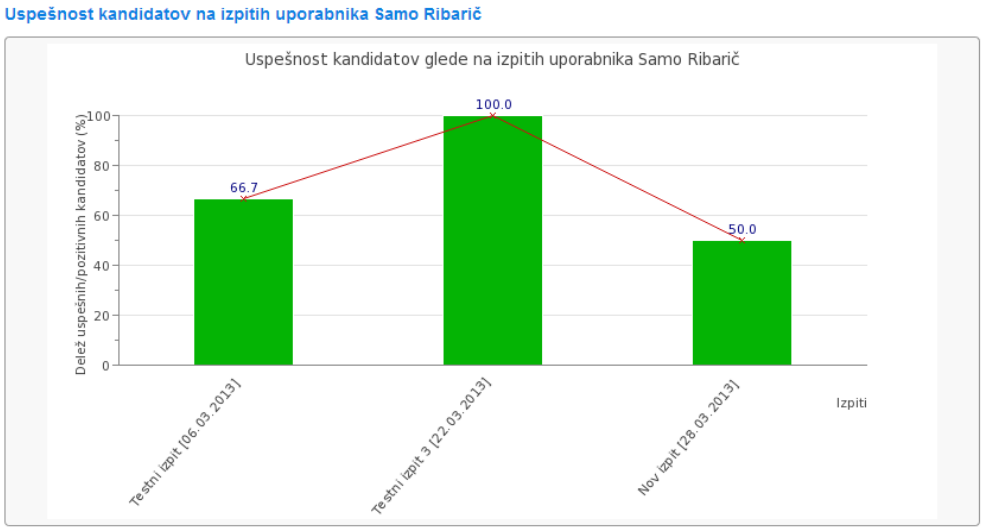
4.5.1 Izdelava vmesnika za zaznavo in popravo izpitnih pol

V sklopu starih programov e-Asistent je tudi program, ki je sprocesiral poskenirane slike izpitnih pol in zaznal označene odgovore v tabeli. Slabost programa je bila, da je bil uporabnik po uvozu slik v program prisiljen pogledati vsako izpitno polo ne glede na pravilno oz. napačno zaznavo. Deloval je z nalaganjem slike na platno (angl. *canvas*). Ko se je naložila, se je izvedel algoritem zaznave robov, barkode in označenih križcev v tabeli (izpitna pola je prikazana na sliki 4.11). Ob pravilni zaznavi je uporabnik pritisnil na gumb *naslednja izpitna pola* in zopet se je ponovilo enako zaporedje akcij. Če je bila zaznava napačna, je s kliki na ustrezna mesta v tabeli spreminjal označbe kandidata. Nov vmesnik je moral simulirati omenjeni program in pohitriti pregled in ročno popravo skozi vse izpitne pole.

Prvi problem je bil, na kakšen način naj implementiram vmesnik v spletno



Slika 4.22: Pogled *analiza/statistika_izp.php*



Slika 4.23: Pogled *analiza/statistika_upor.php*

aplikacijo. Za izdelavo zahtevnejših grafičnih vmesnikov je v zadnjih časih zelo popularna uporaba tehnologije HTML5 v povezavi s skriptnim jezikom Javascript. Implementacija preko omenjenih tehnologij predstavlja problem predvsem pri starejši brskalnikih, ki ne podpirajo HTML5. Na Medicinski fakulteti imajo še vedno dokaj staro računalniško opremo in po večini še prevladujejo operacijski sistemi Windows XP. Uporabniki največkrat uporabljajo še brskalnik Internet Explorer verzije manj kot 9, kar pomeni nekompatibilnost s omenjeno novo tehnologijo. Odločil sem se implementirati vmesnik v obliki javanskega apleta, vendar vseeno z občutkom, da bo potrebno prej ali slej uporabiti drugačen pristop. V zadnjih časih se brskalniki nekako izogibajo javanskih programov. Njihovo izvajanje je na splošno onemogočeno zaradi varnosti. V brskalnikih je zato treba izvajanje javanskih programov omogočiti.

Drugi problem je predstavljala pohitritev zaznave pravilnih izpitnih pol. Potrebno je bilo spremeniti način zaznave slik in se znebiti prve faze risanja slike na platno (angl. *canvas*). Uporabnikom je bilo potrebno omogočiti pregled nad pravilno in nepravilno zaznanimi izpitni polami. Nov vmesnik vse-

buje prikaz vseh izpitnih pol, barva njihovega okvirja pa predstavlja napačno oz. pravilno zaznavo. Enobitno sliko formata BMP prebere neposredno iz datoteke ("po bitih"). Prebrane piksle shrani v matriko oz. dvodimenzionalno tabelo, ki predstavlja celo sliko. Velikost matrike je velikost slike tako po dolžini kot širini. Nad to matriko se najprej izvede algoritem rotacije, v primeru če je slika poskenirana preveč postrani. Sledi zaznava glavnih robov okoli tabele. Potem se določijo robovi med celicami glede na razmerje glavnih robov. Vse piksle v posamezni celici poimenujemo po številki vrstnega reda celice. Izvede se še zaznava križcev v posamezni celici glede na delež črnih pikslov znotraj okvirja. Zaznane označbe shranjuje v seznam za vsako izpitno polo, shranjuje tudi zaznano barkodo tipa Codabar. Po končani zaznavi zopet sledi branje naslednje slike v matriko in proces se ponovi za vsako datoteko v izbranem direktoriju.

Tretji problem je bil, kako dostopati do podatkovne baze in pisati zaznave v ustrezno tabelo MySQL. Lahko bi dostopal do baze kar s knjižnico JDBC API, ki omogoča izvajanje poizvedb in stavkov SQL v različnih tipih baz, tudi MySQL preko pogona MySQL Connector/J. Odjemalec bi preko javanske aplikacije dostopal neposredno do podatkovne baze. Vrata 3306 (angl. *port 3306*) bi morala biti stalno odprta na strežniku (vrata 3306 veljajo za bazo MySQL), kar pa predstavlja nevarnost vdora na nivoju baze. Nov vmesnik kliče akcijo *oceniIzpit* (opisana v 4.4.4) preko protokola HTTP in ji pošlje zaznane podatke posameznega izpita preko metode POST. Potem čaka na njen odziv (odsek kode je na sliki 4.24). Ko akcija zapiše zaznave v podatkovno bazo, se odzove vmesniku, ki pošlje zaznane podatke naslednjega izpita.

4.5.2 Vpeljava uporabniških vlog v sistem in izmenjava dovoljenj med uporabniki

Ena izmed razširitev starega sistema e-Asistent je bila vpeljava uporabniških vlog v sistem. Vsakemu uporabniku pripada ena od treh vlog: Asistent, Profesor ali Administrator. Upoštevanja vlog nisem mogel implementirati

```
private int sendPostData(String uri, String barcode,
        ArrayList<Integer> crosses)
{
   try {
       String data1 = URLEncoder.encode("barcode", "UTF-8") +"="
           + URLEncoder.encode(barcode, "UTF-8");
       String data2 = URLEncoder.encode("crosses", "UTF-8") + "=
          " + URLEncoder.encode(crosses.toString()+"", "UTF-8");
       String data = data1 +"&" +data2;
       String inLine;

       URL u1 = new URL(uri);
       URLConnection uc1 = u1.openConnection();

       uc1.setDoOutput(true);
       OutputStreamWriter out = new OutputStreamWriter(uc1.
           getOutputStream());
       out.write(data);
       out.flush();

       BufferedReader in = new BufferedReader( new
           InputStreamReader( uc1.getInputStream() ) );
       int q=-1;
       while ((inLine = in.readLine()) != null) {
          q=Integer.parseInt(inLine);
          this.qN=(q>=0)? q:100;
       }
       in.close();
       return q;
   }
   catch( Exception e ) {
       e.printStackTrace();
   }
   return -1;
}
```

Slika 4.24: Koda v programskem jeziku Java, vmesnikova funkcija za pošiljanje podatkov zaznave posameznega izpita preko metode POST

na način, ki ga je ponujal Yii, saj vdelan sistem vlog upošteva le delitev že vnaprej določenih vlog. Potrebno je bilo implementirati tudi možnost, da nek uporabnik drugemu uporabniku dovoli izvajati akcije na svojih objektih, kot so vprašanja, izpiti ali procesi priprave izpita. Osredotočil sem se predvsem na kontrolerje *VprasanjaController*, *IzpitiController*, *PripravaController* in *PopravaController*. Vsi ti kontrolerji vsebujejo akcije, pri katerih je potrebno preveriti, ali določen uporabnik lahko izvaja akcijo glede na lastno vlogo ali glede na posredovano dovoljenje nekoga drugega.

V vsaki akciji je potrebno preveriti vlogo uporabnika. Če vloga dopušča izvajanje akcije, potem se preverjanje konča in akcija se izvede. Ko je izvajanje akcije za vlogo prepovedano, je potrebno preverjanje oz. branje v tabeli *dovoljenje_tbl* podatkovne baze. Če je v tabeli dovoljenj zapisana vrstica, ki uporabniku dovoljuje izvajanje te akcije na določenih objektih, potem se akcija izvede.

Glede na to, da so funkcije preverjanja dovoljenj enake za vse akcije omenjenih kontrolerjev, sem jih implementiral kar v starševskem razredu *Controller* (razred izpeljan še iz osnovnega razreda *CController*), iz katerega so izpeljani vsi obstoječi kontrolerji. Na sliki 4.25 je implementirana funkcija preverjanja.

4.5.3 Implementacija zaporedja procesa priprave in poprave izpita

Ko uporabnik v sistemu ustvari izpit, lahko prične s procesom priprave in poprave izpita. Proces je sestavljen iz štirih stanj: urejanje vprašanj v izpitu, prijava in odjava kandidatov, tiskanje izpitov in poprava izpitov. Stanja si morajo slediti po zaporednem vrstnem redu. Zamislite si scenarij, ko uporabnik natisne izpitne pole in izpite, potem se vrne v stanje urejanja vprašanj, odstrani dve vprašanji in doda novo vprašanje v izpit, v stanju prijave in odjave prijavi še dodatnega kandidata itd. Pride do zmede, saj so izpitne pole že natisnjene z določenimi vprašanji in že pripadajo določenim kandidatom.

```
/* Ali je uporabniku dovoljeno izvajanje akcije */
/* @param $action katera akcija */
/* @param $ownerOfModel_id lastnik objekta */
/* @param $user_id uporabnik, ki hoce izvesti akcijo */
/* @param $ime_ObjektPravicTb_id za kateri objekt se gre, ali
   vprasanja, izpiti...*/
protected function accessGranted($action,$ownerOfModel_id,
   $user_id,$ime_ObjektPravicTbl){
     if(Yii::app()->user->vloga=="Administrator")
                return true;
      if($ownerOfModel_id!=$user_id){
            $sql='SELECT dovoljenje_tbl.* FROM dovoljenje_tbl
               LEFT JOIN objekt_pravic_tbl ON dovoljenje_tbl.
                objekt_pravic_tbl_id_objekt_pravic=
                objekt_pravic_tbl.id_objekt_pravic WHERE
                dovoljenje_tbl.uporabnik_tbl_id_upor='.
                $ownerOfModel_id.' AND dovoljenje_tbl.
                uporabnik_tbl_id_upor_dovoljeno='.$user_id.' AND
                objekt_pravic_tbl.ime="'.$ime_ObjektPravicTbl.'
                "';
            $dovoljenje=Dovoljenje::model()->findBySql($sql);
            if($action=='dodaj'){
                return ($dovoljenje['ustvari']==1)? true:false;
            }else if($action=='uredi'){
                return ($dovoljenje['ureja']==1)? true:false;
            }else if($action=='zbrisi'){
                return ($dovoljenje['brise']==1)? true:false;
            }else if($action=='vidi'){
                return ($dovoljenje['vidi']==1)? true:false;
            }else{
                return false;
            }
        }else
            return true;
 }
```

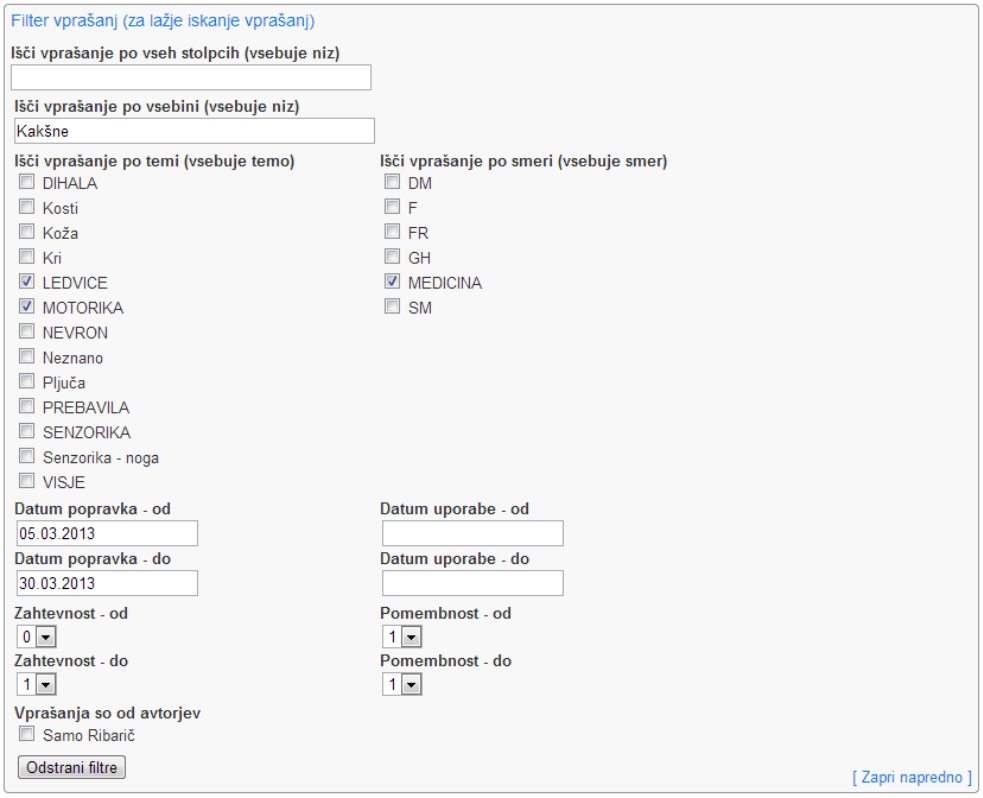
Slika 4.25: Funkcija preveri, ali je uporabniku dovoljeno izvajati akcijo na določenem objektu nekega lastnika, kot so vprašanja, izpiti, procesi priprave in poprave izpitov.

Kontrolerja *PripravaController* in *PopravaController* skrbita za zaporedni vrstni red izvajanje teh akcij. Tabela *izpit_tbl* podatkovne baze vsebuje atribut *status*, v katerem se zapisuje zaporedna številka stanja, v katerem se nahaja. Ko dostopamo do akcije *urediVprašanja*, se nastavi status izpita na 1. Ko gremo na naslednjo akcijo *urediPrijave*, preverimo, ali je izpit v statusu 1, dovolimo izvajanje te akcije ter nastavimo status na 2. Pri akciji *natisni* preverimo, ali je izpit v statusu 2. Če je, nastavimo status na 3, vendar le v primeru, ko je dokument PDF (preko njega natisnemo izpite in izpitne pole) posredovan uporabniku. Akcija *popravi* preveri, ali ima izpit status 3, in nastavi status na 4. Ko zaključimo izpit se nastavi status na 5, kar predstavlja konec procesa.

Uporabniku je omogočeno, da vstopi v katerokoli stanje, če ga je že obiskal. Vendar mora v tem primeru vsa naslednja stanja zopet začeti znova (saj izvedba izbrane akcije zopet nastavi status na primerno vrednost). Na sliki 4.6 je predstavljena tabela, v kateri lahko s klikom na modro označene okvirje vstopamo v katerokoli stanje.

4.5.4 Možnost filtriranja izpitnih vprašanj

Za uporabnika je pomembno, da ima možnost filtriranja vprašanj po različnih lastnostih. Tako lažje najde določena vprašanja. To je še posebej pomembno za Asistenta, ki skrbi za vprašanja različnih Profesorjev. Za prikaz tabele vprašanj uporabljam zunanjo knjižnico JQuery Datatables, ki ima vgrajeno logiko za filtriranje vrednosti v tabeli z uporabo skriptnega jezika Javascript. Glede na to, da je vprašanj zelo veliko, sem uporabil način filtriranja s strani strežnika (angl. *server-side*). Ko se izvrši filtrirna funkcija (*fnFilter()*) na tabeli HTML, se z uporabo tehnologije AJAX kliče funkcija *vprasanjaDataTable* v kontrolerju *Vprašanja*. Ta izpiše filtrirane vrednosti. Parametri filtrirne funkcije se omenjeni funkciji pošljejo preko metode GET. V funkciji *vprasanjaDataTable* se izvrši poizvedba SQL, ki se mora dinamično kreirati glede na poslane parametre filtrirne funkcije. Na sliki 4.26 je prikaz filtra vprašanj.



Slika 4.26: Filter vprašanj

Poglavje 5

Zaključek

Z uporabo odprtokodnega spletnega ogrodja Yii sem uspel ustvariti spletno aplikacijo, ki se lahko primerja s starim sistemom. Nova spletna aplikacija e-Asistent je zmožna zajeti vse glavne funkcionalnosti starega sistema, hkrati pa z implementacijo uporabniških vlog in kontroliranim procesom priprave in poprave izpitov razširi organizacijske funkcionalnosti. Novi e-Asistent je že pripravljen za uporabo, saj je zmožen zaključiti celoten cikel priprave izpitov, hraniti njihove rezultate, prikazovati statistične izračune na podlagi rezultatov skozi celotno zgodovino in urejati izpitna vprašanja.

Aplikacijo sem poskušal razvijati po metodologiji slapovnega življenjskega modela razvoja, vendar je v praksi to težko izvedljivo. Zaradi sprotnega usklajevanja med mano in somentorjem pri zajemanju zahtev sem se moral kdaj pa kdaj vračati na prvotne faze razvoja. K temu so pripomogle tudi potrebe po dodatnih zahtevah še med fazo implementacije. Na primer pri implementaciji dodeljevanja dovoljenj med uporabniki smo se šele kasneje zavedli, da k pripravi izpitov velikokrat pristopijo tudi asistenti in drugi sodelavci v inštitutih. V osnovi se je razvoj najbolj približal slapovnemu modelu. Svojega projekta nisem že na začetku razbil na podmnožice določenih funkcionalnosti in jih obdelal po vseh fazah (analiza, načrtovanje, implementacija in testiranje) ter se lotil obdelave naslednje podmnožice, kot je značilno za iterativni model razvoja [5]. Danes velja, da vsak razvoj projekta po za-

porednih fazah, ki jim sledimo, kolikor je le mogoče, velja za slapovni oz. zaporedni model razvoja [9].

Novi spletni sistem ima še veliko možnosti za razširitev svojih funkcionalnosti. Ena izmed razširitev bi bila možnost elektronskega reševanja izpitov. Potrebno bi bilo v sistem vpeljati kandidate kot akterje sistema. Kandidati bi preko vmesnika reševali izpite podobno, kot sedaj rešujejo kvize v sistemu Moodle. Nadaljnji razvoj bo tudi na komunikaciji med sistemoma Moodle in e-Asistent. E-Asistent bi lahko uporabljal podatke iz spletne učilnice, recimo vprašanja iz kvizov, uporabnike kot kandidate in učitelje itd. Sistem bi lahko izboljšali tudi z uporabo nove tehnologije za gradnjo vmesnika za zaznavo izpitnih pol. Poskusil bi ga implementirati z uporabo kombinacije jezikov HTML5 in Javascript. Izboljšal bi tudi sistem vpisa uporabnikov, vsak uporabnik bi dobil svoj certifikat, preko katerega bi dostopal v sistem. S tem bi se tudi varnost bistveno povečala.

Med delom sem tudi dopolnil svoje znanje o delovanju internetnih zahtev med odjemalcem in strežnikom. Seznanil sem se z uporabo spletnih ogrodij, predvsem s tehnologijami, ki jih različna ogrodja uporabljajo (ORM, MVC, preslikovanje naslovov URL itd). Naučil sem se uporabljati spletno ogrodje Yii, pri čemer mi je poleg literature veliko pomagala tudi podpora s strani različnih forumov, kar dokazuje popularnost ogrodja. Z gradnjo vmesnika za zaznavo izpitnih pol sem si razširil znanje programskega jezika Java in njenih knjižnic, predvsem na grafičnem nivoju. Naučil sem se tudi načina komuniciranja med strežnikom in apleti, ko sem moral implementirati način proženja akcij s strani apleta. Seznanil sem se tudi s tehnologijami, kot so AJAX, format JSON, knjižnica JQuery, in dopolnil znanje o skriptnem jeziku Javascript ter strežniškem jeziku PHP.

Literatura

[1] Barrz W. Boehm, *A Spiral Model of Software Development and Enchancement*, dostopno na: http://weblog.erenkrantz.com/ jerenk/phase-ii/Boe88.pdf

[2] Alistair Cockburn, *Writing effective use cases*, Addison-Wesley, 2000

[3] Jason Cole, Helen Foster, *Using Moodle, 2nd edition*, O'Reily Media, 2007

[4] Amy Fowler, *Painting in AWT and Swing*, dostopno na: http://www.oracle.com/technetwork/java/painting-140037.html

[5] Martin Fowler, *UML Distilled: A brief guide to the standard object modeling language*, 2003

[6] Bartosz Porebski, Karol Przystalski, Leszek Nowak, *Building PHP Applications with Symfony, CakePHP and Zend Framework*, Wiley Publishing, 2011

[7] Elizabeth Sugar Boese, *An introduction to programming with Java Applets, 3rd Edition*, Jones & Bartlett Learning, 2009

[8] Jeffery Winesett, *Agile Web Application Development with Yii 1.1 and PHP5*, Packt Publishing, 2010

[9] Donald Yeates, Tony Wakefield, *Systems Analysis and Design, Second edition*, 2004