Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Kozlevčar

Preverjanje pravilnosti nalog z uporabo sistema Moodle

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Tomaž Dobravec

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu se osredotočite na področje avtomatskega ocenjevanja računalniških programov. Preglejte obstoječe programe za preverjanje pravilnosti in kakovosti programov ter opišite podobnosti in razlike. V diplomskem delu razvijte tudi samostojen modul sistema Moodle za ocenjevanje oddanih programov v različnih programskih jezikih. Modul naj omogoča enostavno administracijo za izvajalca predmeta in pregledno uporabo za študenta. Modul naj bo tesno povezan s sistemom Moodle in naj omogoča vpis ocen direktno v redovalnico. Izdelajte tudi namestitveni program za enostavno vključitev modula v obstoječ sistem Moodle.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Nejc Kozlevčar, z vpisno številko **63120211**, sem avtor diplomskega dela z naslovom:

Preverjanje pravilnosti nalog z uporabo sistema Moodle

IZJAVLJAM

1. da sem pisno zaključno delo študija izdelal samostojno pod mentorstvom doc. dr. Tomaža Dobravca;
2. da je tiskana oblika pisnega zaključnega dela študija istovetna elektronski obliki pisnega zaključnega dela študija;
3. da sem pridobil/-a vsa potrebna dovoljenja za uporabo podatkov in avtorskih del v pisnem zaključnem delu študija in jih v pisnem zaključnem delu študija jasno označil/-a;
4. da sem pri pripravi pisnega zaključnega dela študija ravnal/-a v skladu z etičnimi načeli in, kjer je to potrebno, za raziskavo pridobil/-a soglasje etične komisije;
5. soglašam, da se elektronska oblika pisnega zaključnega dela študija uporabi za preverjanje podobnosti vsebine z drugimi deli s programsko opremo za preverjanje podobnosti vsebine, ki je povezana s študijskim informacijskim sistemom članice;
6. da na UL neodplačno, neizključno, prostorsko in časovno neomejeno prenašam pravico shranitve avtorskega dela v elektronski obliki, pravico reproduciranja ter pravico dajanja pisnega zaključnega dela študija na voljo javnosti na svetovnem spletu preko Repozitorija UL;
7. dovoljujem objavo svojih osebnih podatkov, ki so navedeni v pisnem zaključnem delu študija in tej izjavi, skupaj z objavo pisnega zaključnega dela študija.

V Ljubljani, dne 7. september 2016 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
SPOJ	Sphere online judge	Sphere online judge
PHP	PHP Hypertext Preprocessor	PHP Hypertext Preprocessor
HTML	Hyper Text Markup Language	Jezik za označevanje nadbesedila
SQL	Structured Query Language	Strukturirani povpraševalni jezik
IP	Internet Protocol	Internet protokol
MB	Megabyte	Megabyte
GB	Gigabyte	Gigabyte
IIS	Internet Information Server	Internetni informacijski strežnik
URL	Uniform Resource Locator	Enolični krajevnik vira

Povzetek

Naslov: Preverjanje pravilnosti nalog z uporabo sistema Moodle

V okviru našega diplomskega dela ustvarimo sodniški sistem primeren za uporabo na fakulteti. Sistem je namenjen preverjanju pravilnosti domačih nalog napisanih v programskem jeziku Java. Uporablja se na spletni učilnici Moodle. Sistem smo realizirali v programskem jeziku PHP, z uporabo platforme Docker povečamo varnost sistema.

Na začetku na spletu najdemo nekaj podobnih sistemov, jih opišemo in naštejemo prednosti in slabosti.

V teoretičnem delu opišemo in predstavimo glavne platforme, ki smo jih uporabili za realizacijo sistema. V nadaljevanju podrobno opišemo delovanje in realizacijo programa. V diplomsko nalogo vključimo tudi nekaj izsekov programske kode in opišemo njeno delovanje.

Ker je naš sistem vključen v Moodle, lahko izdelek testiramo kar znotraj Moodla. Izgled samega sistema prikažemo s pomočjo zaslonskih slik.

Ključne besede: Moodle, pravilnost nalog, sodniški sistem, Docker, PHP.

Abstract

Title: Homework checker for Moodle

Throughout our thesis we implement a judge system suitable for faculty application. The system is intended for checking the correctness of homeworks written in Java programming language. It is used in combination with learning management system Moodle. The system is written in PHP programming language and with the use of the platform Docker we enhance its security.

At first we find some similar systems, describe them and list their strengths and weaknesses.

In the theoretical part we describe and introduce the main platforms that we used for the implementation of the system. In continuation we thoroughly describe the functionality and implementation of the program. In the thesis we also include some parts of the source code and explain its functionality.

Because our system is included in Moodle, we can test the product from within. The system's appearance is illustrated with the help of screenshots.

Key words: Moodle, homework checker, judge system, Docker, PHP.

Poglavje 1

Uvod

1.1 Motivacije

Na fakulteti za računalništvo in informatiko je veliko predmetov, kjer je za domačo nalogo potrebno narediti nek program. Pravilnost teh nalog profesorji pregledujejo na več različnih načinov. Nekateri zahtevajo ustni zagovor vsakega posameznika, kar je dokaj zamudno in s tem profesor oz. asistent izgubi vsaj 1 termin vaj za predavanje snovi. Spet drugi imajo na svojem strežniku postavljen sistem za avtomatsko preverjanje teh nalog. Problem teh sistemov je, ker v večini niso uporabniku ali skrbniku prijazni. Na fakulteti ni nekega univerzalnega sistema, ki bi ga lahko vsi enostavno uporabljali preko spletne učilnice - Moodle.

Moodle je odprto kodna programska oprema, namenjena izobraževalnim ustanovam. Vsak profesor je zadolžen za urejanje svojih predmetov, kjer so vpisani študenti, ki ta predmet opravljajo. Moodle omogoča enostavno oddajanje domačih nalog in komunikacijo med študenti in profesorjem oz. asistentom.

1.2 Cilji

Cilj diplomske naloge je narediti nek splošen sistem za preverjanje programskih nalog in ga vključiti v sistem Moodle. To bo bistveno olajšalo točkovanje domačih nalog tako za študente kot za profesorje. Študentje bodo tako lahko nalogo samo oddali na spletno učilnico. Naloga se bo nato avtomatsko preverila, dosežene točke pa se bodo samodejno vpisale v sistem.

Zdajšnji sistemi so bili za skrbnike neprijazni, saj je administracija možna le preko konzole in ne preko uporabniškega vmesnika. Z našim sistemom omogočimo skrbniku enostavno dodajanje in odstranjevanje testnih primerov. Še vedno pa je možno nalogo ročno preveriti in po potrebi spremeniti točke.

1.3 Metodologije

Prva faza diplomske naloge je sestavljena iz praktičnega dela. Na začetku se lotimo realizacije samega sistema, ki ga nato povežemo s sistemom Moodle. Sistem spišemo v programskem jeziku PHP, ker ga je najlažje povezati z Moodlom.

V drugi fazi smo se osredotočili bolj na teorijo izdelanega sistema in ga podrobneje opisali. Opisali bomo delovanje in spisali kratka navodila za uporabo, da ga bojo uporabniki lažje usvojili.

Poglavje 2

Primerjava z obstoječimi sodniškimi sistemi

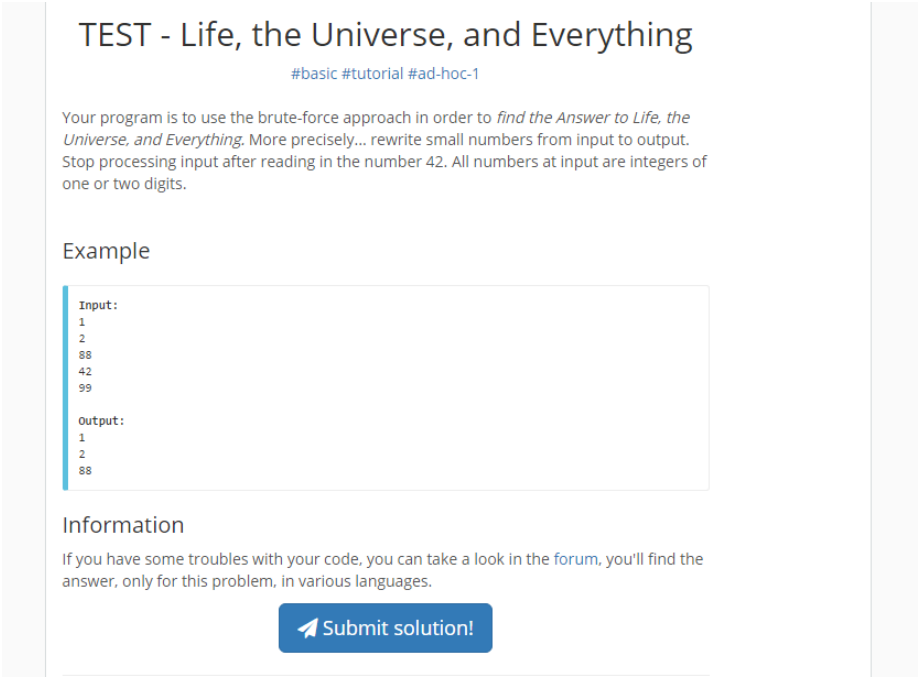
Sodniški sistemi so sistemi za testiranje pravilnosti programske kode. Sistem kodo prevede in zažene na priloženih testih. Izhod programa se preveri s pravilnimi izhodi, ki so bili definirani pred zagonom naloženega programa. Test je pravilen, če se oba izhoda popolnoma ujemata. Na spletu je mogoče najti kar nekaj takih sodniških sistemov. Vseh ne moremo analizirati in testirati, zato bomo to storili zgolj za izbrane sisteme [1]. Osredotočili se bomo na naslednje tri sisteme:

- Sphere online judge,
- Code Chef,
- Top Coder.

2.1 Sphere online judge

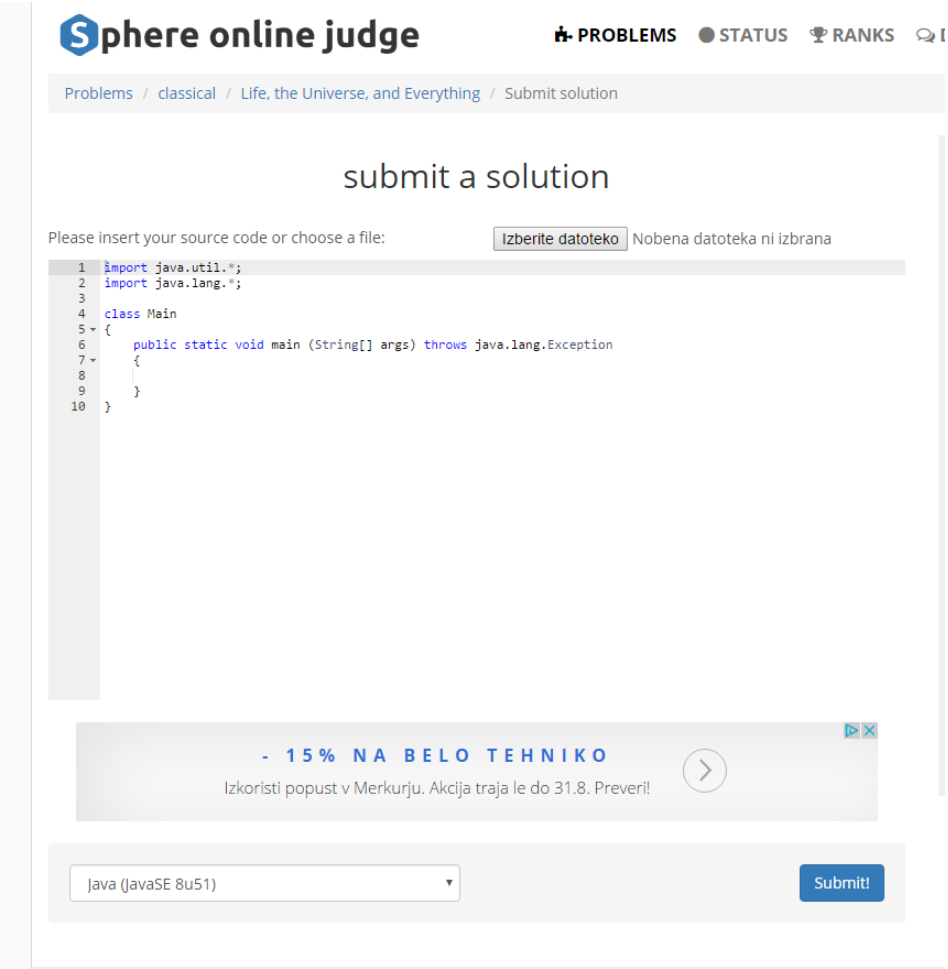
Sphere Online Judge (SPOJ) je sodniški sistem dostopen preko spleta. Za uporabo sistema ni potrebno namestiti dodatne programske opreme. Uporaba sistem je brezplačna, potrebna je zgolj prijava. Na SPOJ je registriranih preko 200 000 uporabnikov, ki imajo na voljo 20 000 nalog za reševanje. Naloge pripravlja za to zadolžena skupnost - običajno so vzete iz končanih programerskih tekmovanj. SPOJ omogoča tudi organizacijo tekmovanj. Tekmovanje lahko teče pod poljubnimi pravili, ki jih zastavi ustvarjalec tekmovanja. Sistem omogoča tudi forum, kjer lahko programerji komunicirajo med seboj [2][3].

Ko se prijavimo v sistem, se nam pokaže seznam problemov za reševanje. Poleg problema je napisana težavnost, odstotek pravilno oddanih nalog in število uporabnikov, ki je to nalogo reševalo. S klikom na željen problem se nam pokažejo navodila naloge s primerom.



Slika 2.1: Prikaz navodil (SPOJ) [4]

SPOJ ima na spletni strani vgrajen tudi preprost urejevalnik besedila, kamor pišemo kodo. Ko izberemo željen programski jezik, se nam v urejevalnik izpiše ogrodje programa. Sistem nam sam ustvari razred in metodo `main`. Če nam ponujeni urejevalnik ne ustreza, lahko kodo pišemo v poljubnem programu in datoteko s programom nato naložimo na portal.



Slika 2.2: Urejevalnik besedila (SPOJ) [5]

Ko napišemo program, ga lahko oddamo v testiranje. Sistem nato na našem programu izvede vse teste. Po končanem testiranju se nam izpiše pravilnost programa. Če so bili vsi testi pravilni, se naša oddaja obarva zeleno, drugače pa oranžno.

Slika 2.3: Prikaz pravilno rešenega problema (SPOJ) [6]

Slabost sistema SPOJ je v slabem urejevalniku besedila, ki ga ponujajo na spletni strani. Največja pomankljiovst urejevalnika je ta, da kode ni mogoče sproti prevesti in zagnati.

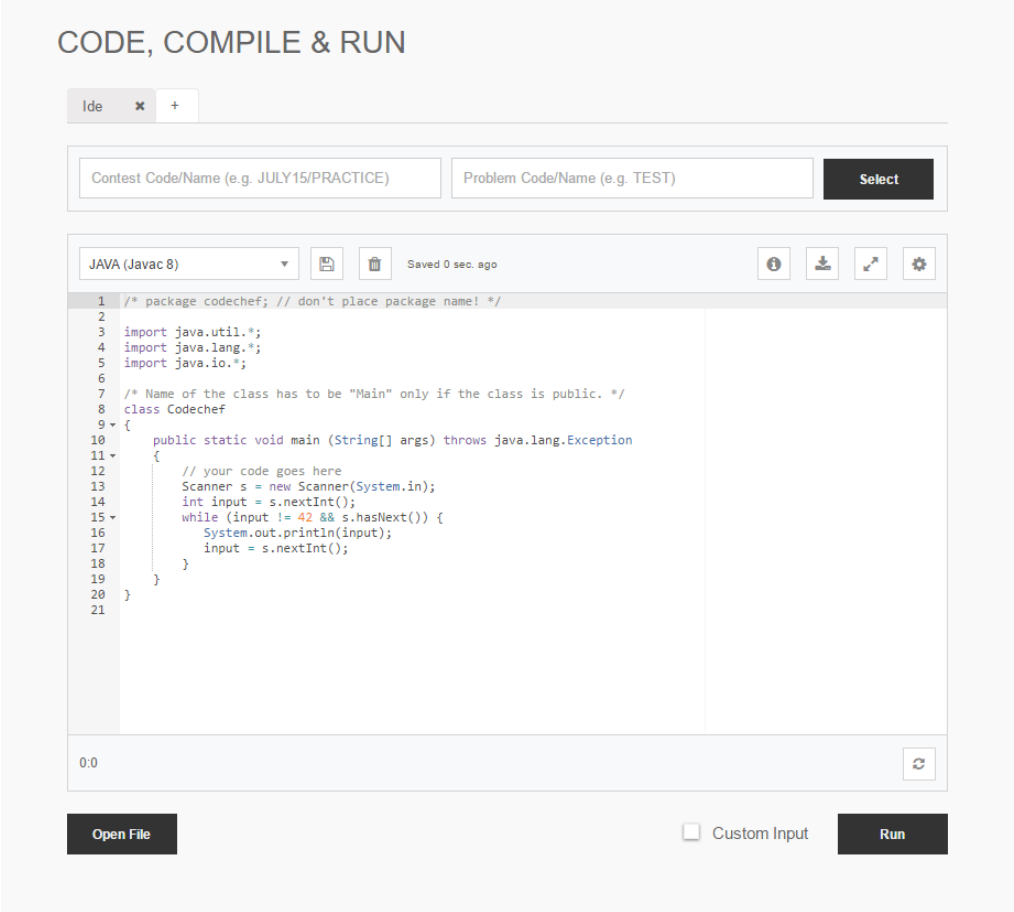
2.2 CodeChef

CodeChef je neprofiten izobraževalni sistem, ki ga je razvilo podjetje Directi. CodeChef podpira preko 50 programskih jezikov in je sprva ciljal predvsem na indijske študente in programerje. CodeChef ima veliko bazo programerjev, ki pomagajo študentom izboljšati znanje programiranja. CodeChef trenutno uporablja več kot 300 000 uporabnikov. Vsak mesec gosti tudi 3 tekmovanja, kjer zmagovalcem podarijo nagrade. [7].

Tako kot za prej opisani SPOJ tudi za CodeChef velja, da si za uporabo tega sistema ni potrebno naložiti dodatne programske opreme. CodeChef ima probleme razvrščene v težavnostne kategorije. Tako imamo na voljo kategorije za začetnike, lahko, srednjo in težko kategorijo ter kategoriji izzivov in problemov, ki so bili na univerzitetnih tekmovanjih. Ko izberemo kategorijo se nam prikažejo vsi problemi, ki so na voljo v tej kategoriji.

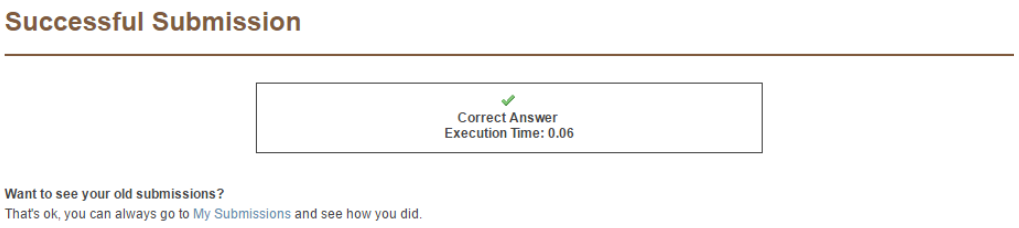
S klikom na problem se nam pokaže besedilo naloge in primer pravilnega para vhodnih in izhodnih podatkov. S klikom na gumb „submit“ lahko začnemo z reševanjem naloge. Prikaže se nam prostor za oddajo rešitve naloge. Tudi tukaj imamo na voljo dve možnosti - v testiranje lahko oddamo datoteko ali pa nalogo kopiramo v priložen urejevalnik besedila.

Namen vgrajenega urejevalnika je zgolj kopiranje izvorne kode vanj, ker podpira malo dodatnih funkcionalnosti. Ima pa CodeChef na drugi povezavi(`https://www.codechef.com/ide`) prav poseben urejevalnik, ki podpira mnogo več funkcij. Kodo lahko prevedemo in zaženemo kar v brskalniku. Kot parameter k izvajanju lahko dodamo poljuben vhod. Omogoča pa tudi funkcijo „auto-complete“, ki nam ponuja predloge pri pisanju kode.



Slika 2.4: Urejevalnik namenjen pisanju kode (CodeChef) [8]

Ko smo s svojim programom zadovoljni in menimo, da je pravilen, ga oddamo v testiranje. Sistem naš program prevede in na njem izvede vse definirane teste. Kot odgovor nam sistem izpiše "Odgovor je pravilen", če so bili testi uspešni ali "Odgovor je napačen", če so bili testi neuspešni.



Slika 2.5: Pravilno delujoč program (CodeChef) [9]

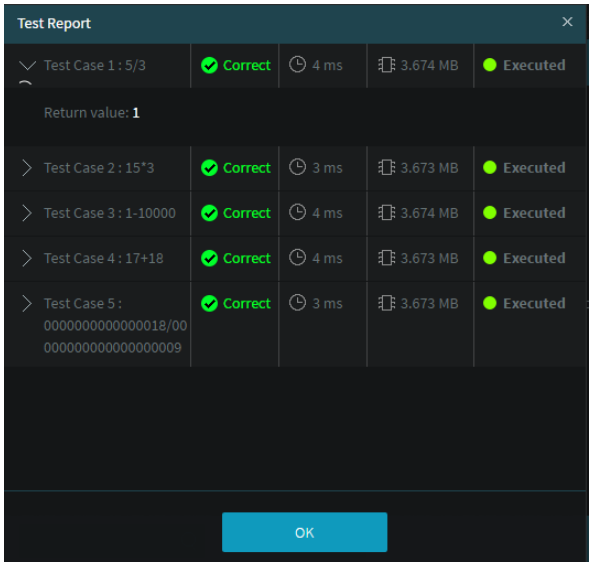
2.3 TopCoder

TopCoder je sistem namenjen povezovanju podjetij s programerji. Vsebuje probleme s področij razvoja, oblikovanje in obdelave podatkov. Uporabniki si z reševanjem realnih problemov izboljšujejo znanje programiranja, hkrati pa lahko zaslužijo tudi nekaj denarja. TopCoder ima trenutno preko 1 000 000 registriranih uporabnikov [10].

Na spletni strani lahko izberemo na katerem natečaju bomo sodelovali in poskušali osvojiti kakšno nagrado. Na voljo imamo tri kategorije: oblikovanje, razvoj programske opreme in analiza podatkov. Ko izberemo eno kategorijo se nam izpišejo vsi izzivi, ki so še vedno odprti. S klikom na izbran izziv se nam odpre opis problema in denarne nagrade za najboljša mesta. Na določene izzive se lahko prijavimo samo pod pogojem, da imamo dovolj visoko oceno.

TopCoder ima tudi tako imenovano tekmovalno areno, kjer so na voljo nekoliko manj zahtevne naloge. Te naloge so bolj podobne nalogam, ki jih uporabljata prej opisana sodniška sistema (SPOJ in CodeChef). Vsaka naloga je vredna določeno število točk. Ko izberem nalogo za reševanje, se nam v ozadju vklopi števec, ki meri porabljen čas za reševanje naloge. Glede na

porabljen čas in vrednost naloge nam sistem dodeli končno število točk. Tudi sam sistem pisanja naloge je tukaj nekoliko drugačen kot pri ostalih dveh sistemih. Pri SPOJ in Code Chef sistemu se preverja izpis programa. Tukaj pa je potrebno logiko programa napisati v prej definirani metodi. Izhod programa pri testiranju nima vpliva na pravilnost. Urejevalnik besedila, ki nam ga ponudi TopCoder je popolnoma osnoven urejevalnik in nima nobenih dodanih funkcionalnosti. Na voljo imamo nekaj vnaprej definiranih testov, ki jih lahko kadar koli izvedemo in kot odgovor dobimo podrobnosti tega testa. Za TopCoder obstaja kar nekaj vtičnikov (ang. plugin) za nekatera najbolj priljubljena razvojna okolja (Eclipse, Intellij IDEA,...).



Slika 2.6: Primer izpisa testov (TopCoder)[11]

Poglavje 3

Uporabljene tehnologije in jeziki

V spodnjem poglavju predstavimo tehnologije in programske jezike, ki smo jih uporabili za realizacijo našega sodniškega sistema. Orodje Docker smo uporabili predvsem zaradi varnosti. Opišemo tudi kakšne vrste zaščite se v njem uporabljajo. Naš sistem je v večini realiziran v programskem jeziku PHP. Jezik je namenjen razvoju spletnih aplikacij. Tudi Moodle je napisan v programskem jeziku PHP, zato je bila vključitev našega sistema v Moodle lažja. Opišemo zgodovino nastanka in na kratko opišemo nekatere sintaktične lastnosti jezika. Moodle je sistem za postavitev spletne učilnice, ki se uporablja tudi na naši fakulteti. Na kratko opišemo postopek namestitve Moodla. Navedemo tudi minimalne zahteve, ki je predpisuje.

3.1 Docker

Docker je brezplačno orodje namenjeno ustvarjanju, razvijanju in poganjanju aplikacij z uporabo tehnologije zabojnikov [12] (ang. container technology). Zabojniki razvijalcu omogočajo združevanje delov, kot so knjižnice in aplikacije v eno neodvisno celoto. S tem dobimo zagotovilo, da bo aplikacija delovala na poljubni napravi, ne glede na nastavitve naprave [13][14].

Docker je na nek način podoben programom za uporabo navideznih računalnikov. Razlika med Dockerjem in programi za uporabo navideznih računalnikov je v tem, da Docker ne zgradi celotnega operacijskega sistema, pač pa le omogoči aplikaciji uporabo istega jedra kot sistem, na katerem aplikacijo poganjamo. To pospeši delovanje aplikacije in zmanjša njeno velikost. Ker je Docker odprto kodni progam, lahko vsak doda svojo dodatno funkcionalnost. Docker je uporaben tako za razvijalce, kot tudi za administratorje aplikacij. Razvijalec se lahko popolnoma osredotoči na pisanje programske kode in se mu ni potrebno ukvarjati s kakšnim sistemom bo imel opravka. Administratorjem doda prilagodljivost in potencialno zmanjša število potrebnih sistemov [15].

3.1.1 Docker in varnost

Naš sistem se bo uporabljal na spletni učilnici na fakulteti, kjer so datoteke in ostale informacije uporabnikov učilnice skupne in zbrane na enem mestu. Zato moramo poskrbeti za ustrezno varnost sistema. Z uporabo Dockerja se študentske naloge izvajajo povsem ločeno od ostalih datotek, ki so shranjene v Moodlu. Tako ni bojazni, da bi lahko kakšno od teh datotek pobrisali oz. spremenili. Docker nam torej prinese veliko mero varnosti.

Zaščita datotek

Za boljšo zaščito so bile v okolje dodane nekatere Linuxove sistemske datoteke, ki pa imajo zgolj bralne pravice. Na srečo večina aplikacij nikoli ne potrebuje pisanja v te datoteke, zato to ne predstavlja večjih težav. Dodane so bile naslednje datoteke:

- . /sys,
- . /proc/sys,
- . /proc/sysrq-trigger,
- . /proc/irq,
- . /proc/bus.

Z dodajanjem teh sistemskih datotek, ki so na voljo samo za branje, privilegirani procesi ne morajo pisati v njih. Tako ne morajo vplivati na originalen sistem. Blokirana je bila tudi pravica za spreminjanje pravic teh datotek.

Docker uporablja tudi tako imenovan „copy-on-write“ datotečni sistem. Tako lahko vsi zabojniki uporabljajo enako datotečno sistemsko sliko kot osnovnik. Ko zabojnik piše vsebino v sliko, se ta vsebina zapiše v datotečni sistem namenjen točno temu zabojniku. To prepreči, da en zabojnik vidi spremembe drugega, tudi če sta pisala po isti datoteki. To pomeni tudi, da zabojnik ne more pisati po datotekah drugih zabojnikov [16].

Imenski prostor

Docker uporablja tudi nekaj svojih imenskih zakonov, ki zagotovijo dodatno varnost. Docker skrije vse procese, ki tečejo na sistemu, z izjemo teh, ki tečejo na našem trenutnem zabojniku. S tem onemogočimo napade na procese, ki niso vidni. Tudi PID (identifikator procesa) 1, ki je ponavadi rezerviran za korenski proces, je proces, ki teče samo na trenutnem zabojniku.

Dodatno zaščito zagotavlja tudi imenski prostor omrežja. Administrator lahko nastavi pravila usmerjanja omrežja za zabojnik. Nastaviti je mogoče celotno IP tabelo, da lahko zabojnik uporablja zgolj določeno vrsto omrežja. 3 najbolj pogoste nastavite:

- dovoljeno je komunicirati zgolj na javnem omrežju,
- dovoljeno je komunicirati zgolj na privatnem omrežju,
- eden lahko komunicira z ostalima dvema, posreduje sporočila v obe smeri med zabojniki, toda blokira neprimerno vsebino [16].

3.2 PHP

PHP je trenutno rekurzivna kratica za „PHP Hypertext Preprocessor“. Izvirno pa je bil akronim za „Personal Home Page Tools“, kar po slovensko

pomeni orodje za osebno domačo stran. PHP je programski jezik za razvoj spletnih aplikacij. Uporablja se na zalednih sistemih. Ima običajno sintaktično strukturo, največ podobnosti pa najdemo s programskima jezikoma C in Perl. Ta podobnost ni naključna, saj so začetki povezni s prav tema dvema jezikoma. Avtor jezika je Rasmus Lerdorf, na Grenlandiji rojen programer. Jezik je bil napisan leta 1994 v programskem jeziku C kot skupina CGI(„Common Gateway Interface“) programov.

PHP interpreter, ki ga poganja Zend Engine, je prosto dostopna programska oprema in jo je mogoče naložiti na skoraj vsak spletni strežnik. Podprt je za skoraj vse operacijske sisteme in platforme. Programski jezik PHP se je do letu 2014 razvijal brez formalno specificiranega standarda. Po leto 2014 pa se je delalo na tem, da se razvijejo formalene PHP specifikacije [17].

3.2.1 Razvoj

Zgodnje izdaje

Razvoj PHP-ja se je začel leta 1995, ko je Rasmus Lerdorf napisal nekaj CGI-programov v programskem jeziku C. Lerdorf jih je nadgradil tako, da je bila mogoča komunikacija s podatkovno bazo. Omogočil pa je tudi delo s spletnimi formami, zato se je ta implementacija imenovala „PHP/Form Interpreter“ ali krajše PHP/FI. Lerdorf je 8. junija 1995 objavil to implementacijo kot prvo verzijo jezika PHP. Prva različica je uporabljala enak način klicanja spremenljivk in upravljanje form kot Perl. Omogočala pa je tudi možnost vgradnje HTMLja. Sintaksa je bila podobna Perl-ovi, toda bolj preprosta, bolj omejena in manj konsistentna[18].

PHP 3 in 4

Zeev Suraski in Andi Gutmans sta leta 1997 ponovno napisala prevajalnik in postavila osnove za PHP 3. Spremenila sta tudi ime jezika v PHP: Hypertext Preprocessor. Po javnem testiranju je bil PHP 3 uradno izdan junija 1998. Leta 1999 sta Suraski in Gutmans ponovno začela pisati novo PHP jedro. Maja 2000 je bil izdan PHP 4, ki ga je poganjal Zend Engine 1.0. Avgusta 2008 je bila zadnja verzija 4.4.9 in PHP 4 se je prenehal razvijati, prenehale so prihajati tudi varnostne posodobitve za to različico[18].

PHP 5

PHP 5 je bil izdan 14. julija, 2004. Poganjal ga je nov Zend Engine II. PHP 5 je podpiral nove funkcionalnosti, kot so objektno pragramiranje, PHP Data Objects (PDO), ki definira lahkoten in konsistenten vmesnik za dostop do podatkovne baze in številne druge izboljšave. PHP 5 je postala edina stabilna verzija v razvoju. Verziji 5.3 in 5.4 sta bili na voljo samo za 32-bitne Windows sisteme, kar so z naslednjo verzijo (5.5) popravili in dodali podporo tudi za 64-bitne Windows sisteme[18].

PHP 7

V verziji 6 je bila načrtovana Unicode podpora, vendar ta verzija ni bila nikoli izdana. Tako je bila naslednja velika nadgradnja verzija 7. Verzija se imenovala tudi phpng, kar pomeni PHP new generation. Avtorji te verzije so bili Dmitry Stogov, Xinchen Hui in Nikita Popov. Osredotočili so se predvsem na optimizacijo Zend Engine-a. Spremembe v tej verziji so bile narejene tako, da so nadgradnje v prihodnosti lažje[18].

Trenutna zadnja stabilna verzija je verzija 7.0.10, ki je bila izdana avgusta 2016. Na voljo pa je tudi različica 7.1.0 beta 3.

3.2.2 Sintaksa

PHP ima običajno sintaktično strukturo, podobno kot C ali Java. Bloki kode so ločeni z zavitimi oklepaji. Beli presledki pa na kodo nimajo vpliva. Vsaka vrstica kode pa se mora zaključiti s podpičjem.

Največja razlika med programskima jezikoma PHP in C je v tem, da se spremenljivke začnejo z znakom dolar ($). Ta lastnost se je prevzela po programskem jeziku Perl. Nizi se v PHP-ju združujejo z znakom pika (.). Vsaka PHP skripta se mora začeti z nizom `<?php` in končati končati z `?>`. PHP kodo lahko vključimo tudi v HTML dokument. To storimo tako, da del PHP kode vpišemo med znakoma `<?php` in `?>`. Z ukazom `echo` izpisujemo željen tekst v HTML dokument [19].

```
<!DOCTYPE html>
<html>
    <head>
        <title>PHP Naslov</title>
    </head>
    <body>
        <?php echo '<p>Danes je lep dan</p>'; ?>
    </body>
</html>
```

Izsek kode 3.1: PHP znotraj HTML dokumenta

PHP razlikuje dve različni vrsti narekovaja, enojni in dvojni. Če uporabimo enojne narekovaje, potem nam niz razume dobesedno. Pri dvojnih narekovajih pa bo prevajalnik spremenljivko zamenjal z dejansko vrednostjo.

```
<?php
$ime = "Nejc";
//Enojni narekovaji
echo 'Moje ime je $ime.'; //IZPIS: Moje ime je $ime.
//Dvojni narekovaji
echo "Moje ime je $ime."; //IZPIS: Moje ime je Nejc.
```

Izsek kode 3.2: Preprost PHP program - primer z narekovaji

Ker PHP podpira tudi objektno programiranje, je mogoče v kodi uporabljati tudi objekte. Razred naredimo tako kot smo navajeni pri ostalih jezikih. Razred se začne z rezervirano besedo `class`, nato sledi ime razreda in logika, ki je obkrožena z zavitimi oklepaji. Atribute in metode objekta kličemo z znakom puščice (->).

```
<?php
class Oseba {
    public $ime;
    public $priimek;

    public function __construct($ime, $priimek = '') {
        $this->ime = $ime;
        $this->priimek  = $priimek;
    }
}
$o = new Oseba("Nejc","Kozlevcar");
echo $o->ime; //Izpis: "Nejc"
echo $o->priimek; //Izpis: "Kozlevcar"
?>
```

Izsek kode 3.3: Primer objektnega programiranja

3.3 Moodle

Moodle je najbolj uporabljen sistem za postavitev spletne učilnice v Sloveniji. Napisan je v programskem jeziku PHP. Po svetu je aktivnih 70569 strani v 232 državah. V Sloveniji je prijavljenih 323 spletnih strani, ki uporabljajo sistem Moodle [20]. Moodle je primeren za uporabo na različnih ravneh izobraževanja. Uporablja se vse od fakultet pa do osnovne šole. Uporabo zasledimo tudi v organizacijah in podjetjih, ki izvajajo neformalno izobraževanje. Lahko pa se uporablja tudi za podporo projektnega dela, saj ponuja kar nekaj možnosti za skupinsko delo.

Moodle je kratica za "Modular Object-Oriented Dynamic Learning Environment", kar v slovenščini pomeni modularno objektno dinamično okolje

za učenje. Deluje na principu odjemalec/strežnik, kar pomeni, da uporabnik potrebuje samo dostop do interneta in spletni brskalnik. To nam omogoča dostop do Moodla od kjer koli in kadar koli [21].

3.3.1 Zgodovina

Avtor Moodla je Martin Dougiamas, nekdanji administrator učnega okolja WebCT. Neprilagodljivost in slaba odzivnost razvijalcev tega sistema so ga pripravili do tega, da je v svojem podiplomskem študiju ustvaril učno okolje, ki je bolj po meri učitelja. Prvo verzijo novega sistema Moodle je predstavil avgusta 2002. Prvotno je bil Moodle narejen za posamezne skupine študentov na nekaterih manjših univerzah, ki si plačljivih proizvodov niso mogle privoščiti. Ravno ta odprtost in brezplačnost sta najbolj pripomogla k razvoju Moodla kot ga poznamo danes. Douglas je prejemal številne predloge, ideje in tudi rešitve, ki so pripomogle k izboljšanju sistema. Trenutno je Moodle na voljo že v več kot 100 jezikih [21].

3.3.2 Vloge v Moodlu

V sistemu Moodle poznamo več vlog uporabnikov. Uporabniške vloge delimo glede na pravice in funkcije uporabnika znotraj sistema. Uporabljajo se naslednje vloge:

- **Administrator strani** - Na strani lahko počne vse, ima vse pravice,
- **Upravljalec** - Malo manjša administrativna vloga, še vedno skoraj vse pravice,
- **Ustvarjalec predmeta** - Lahko ustvari nov predmet,
- **Izvajalec(učitelj)** - Lahko upravlja in dodaja dejavnosti na predmet,
- **Izvajalec(učitelj) brez pravic urejanja** - Lahko ocenjuje v predmetu, toda nima pravic za urejanje predmeta,

- **Udeleženec(študent)** - Lahko dostopa in sodeluje v predmetu,
- **Gost** - Lahko spremlja predmet, toda ne mora sodelovati,
- **Avtenticiran uporabnik** - Pravice, ki jih imajo vsi prijavljeni uporabniki v sistemu,
- **Potrjeni uporabnik na prvi strani** - Pravice samo za začetno stran.

3.3.3 Namestitev

Uporabnik si lahko Moodle brezplačno prenese z uradne spletne strani. Namesti se ga na spletni strežnik, recimo Apache HTTP Server. Podpira tudi različne baze podatkov, kot so MySQL, PostgresSQL ,... Že pripravljena kombinacija spletnega strežnika in podatkovne baze za Moodle obstaja za Microsoft Windows in Mac računalnike. Za ostale Moodlove storitve, kot so gostovanje, usposabljanje, prilagajanje in razvoj vsebin, skrbijo ostali Moodlovi partnerji.

Moodle je podprt za vse večje operacijske sisteme (Unix, Windows, OS X, ...). Edini pogoj je, da podpira PHP in podatkovne baze na spletnem strežniku.

Moodle zahteva vsaj 160MB prostora na disku ter prostor, ki ga potrebujemo ta shranjevanje svojih podatkov. Zaradi tega je priporočeno imeti vsaj 5GB prostora. Predpisani minimum spomina je 256MB, toda zelo priporočljivo je imeti 1GB ali več. Splošno pravilo Moodla je, da lahko podpira 10 do 20 sočasnih uporabnikov za vsak 1GB RAMa. Toda to je odvisno tudi od ostalih specifikacij in vrste uporabe.

Moodle zahteva tudi nekaj obvezne programske opreme. Na računalnik mora biti nameščen operacijski sistem, ki podpira spletni strežnik, PHP in podatkovno bazo. Najbolj priporočljiv spletni strežnik je Appache ali IIS. Minimalna verzija PHP-ja mora biti 5.3.2. Priporočljivi podatkovni bazi sta MySQL in PostgreSQL, ker sta bili najbolj celovito testirani in imata največ dokumentacije in podpore. MSSQL in Oracle sta polno podprti, vendar

dokumentacija in ostala pomoč ni primerljiva z MySQL in PostgrSQL. Verzije podatkvnih baz:

- MySql - 5.0.25 ali novejša
- PostgreSQL - 8.3 ali novejša
- MSSQL - 9.0 ali novejša
- Oracle - 10.2 ali novejša

Uporabnik do portala Moodle dostopa preko spletnega brskalnika. Za to lahko uporablja računalnik, tablico, pametni telefon,... Moodle bi moral delovati na praktično vsakem novejšem brskalniku (Internet Explorer verzije 6 in starejši niso podprti). Operacijski sistem ni pomemben [22][23].

Poglavje 4

Preverjanje pravilnosti nalog

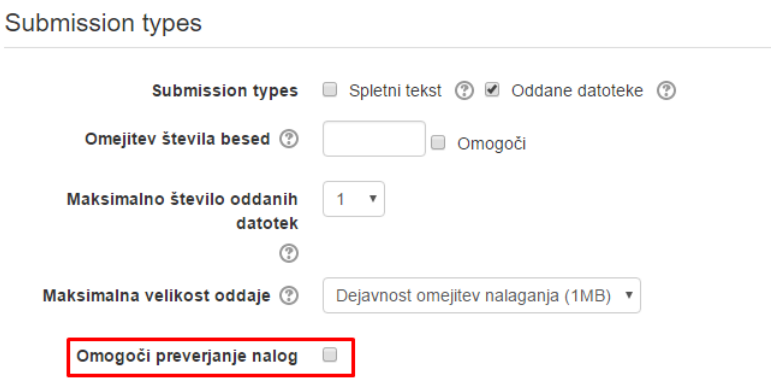
V sistemu Moodle do sedaj še ni vgrajenega avtomatskega preverjanja nalog, z našim sistemom pa se doda ta funkcionalnost. Razširitev omogoča preverjanje pravilnosti algoritmov napisanih v programskem jeziku Java. Uporabnik z učenjem nove funkcionalnosti ne bi smel imeti težav, saj je uporaba enostavna.

4.1 Uporaba

4.1.1 Ustvarjanje nove naloge

Pri dodajanju nove naloge smo v blok „Tip oddaje (ang. Submission types)“, dodali novo lastnost naloge, ki omogoči preverjanje nalog. Če se označi ta možnost, potem se bo naš sistem uporabljal v na novo ustvarjeni nalogi. Izbira preverjanja nalog se shrani v podatkovni bazi. Za realizacijo tega dela je bilo v tabeli „mdl_assign“ potrebno dodati nov stolpec. Privzeta vrednost tega stolpca je 0, tako da nima vpliva na ostale že ustvarjene naloge. (slika 4.1).

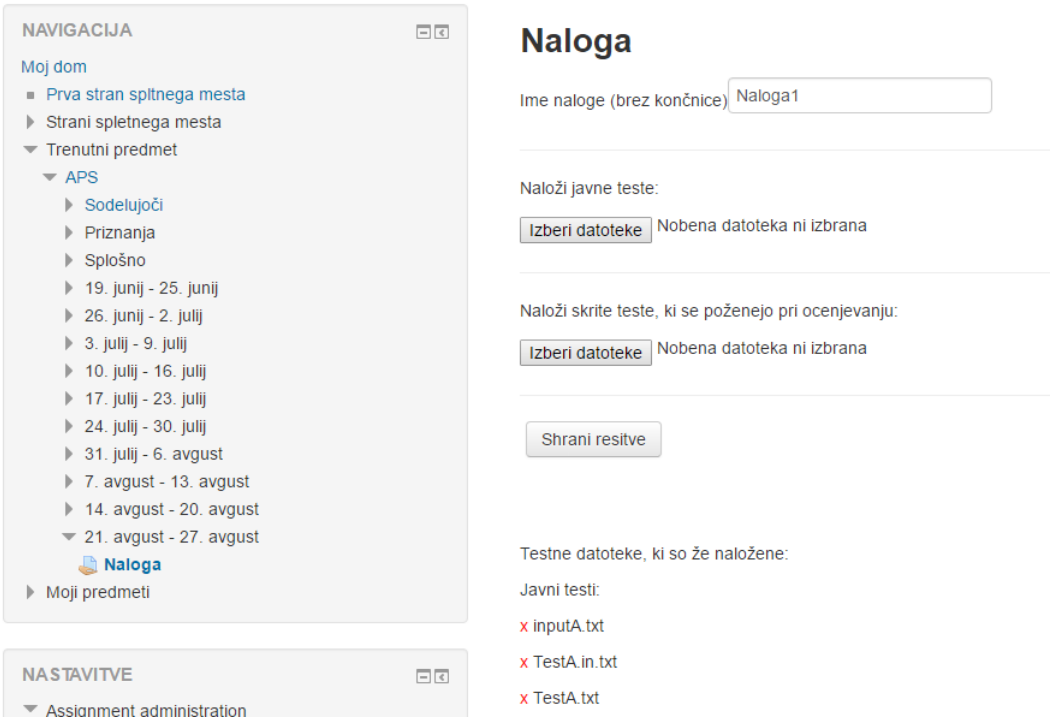
Ko vklopimo možnost preverjanja nalog se nam na začetni strani oddaje pokažeta 2 dodatna gumba (oz. eden, če je vpisan študent). Prvi gumb je namenjen dodajanju rešitev, drugi pa preverjanju in oddajanju nalog.



Slika 4.1: Omogoči preverjanje naloge

4.1.2 Dodajanje testov

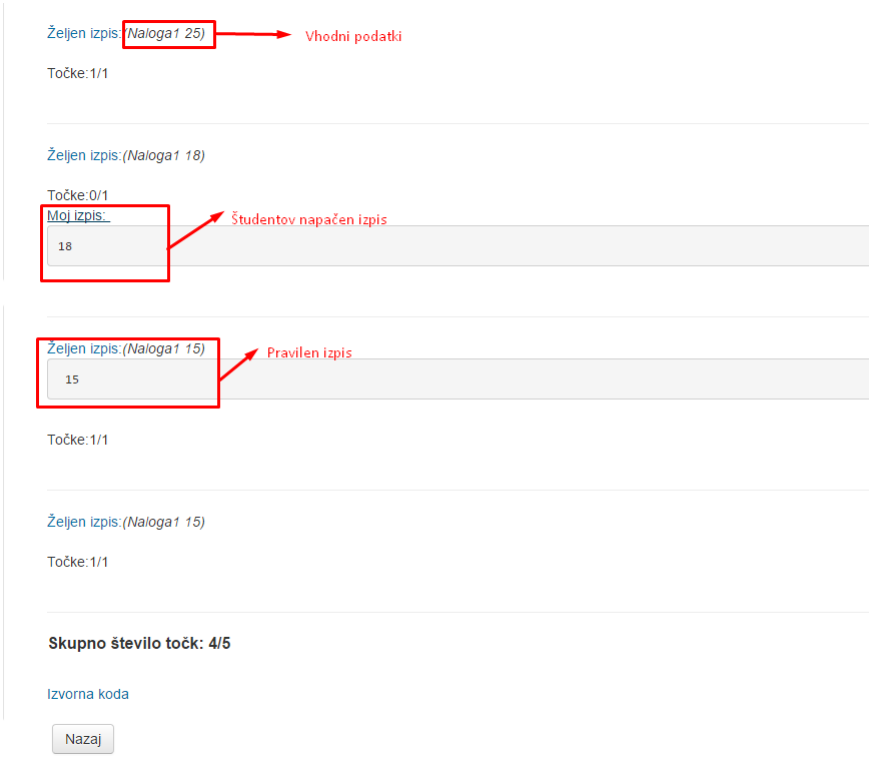
Sistem preverjanja pravilnosti deluje na podlagi ujemanja izhoda programa. Ne zanima nas, kaj se med izvajanjem programa dogaja, toda le, če pri enakih vhodnih podatkih dobimo enake izhodne podatke. Tak način preverjanja se imenuje Black-box testiranje. Profesor mora napisati ime naloge (npr. Naloga1), da sistem prepozna ime javanskega razreda. Nato mora profesor naložiti še testne datoteke. Sistem pozna dve skupini testov. Javni, ki so vidni in se izvedejo, ko študent preveri svojo nalogo. Druga skupina pa so skriti testi. Skriti testi se uporabijo, ko profesor preveri pravilnost nalog za vse študente. Vsak test je sestavljen iz dveh datotek. Prva datoteka predstavlja vhodne, druga pa izhodne podatke. Vhodna datoteka mora imeti končnico „.in“ oz. „.in.txt“ in se mora imenovati enako kot izhodna datoteka. Vsak test se mora začeti z nizom „test“. Vsebina vhodnega dokumenta predstavlja argumente, ki jih program prejme na začetku izvajanja (parametri v metodi main). Kot vhod lahko dodamo tudi vsebino druge datoteke. Ta datoteka pa mora biti dodana na strežnik poleg testov. Vsebina „.in“ datoteke pa se mora začeti z znakom „<“, kateremu sledi ime datoteke. Grafični prikaz dodajanja rešitev je prikazan na sliki 4.2.



Slika 4.2: Primer dodajanja rešitev.

4.1.3 Prikaz pravilnosti naloge

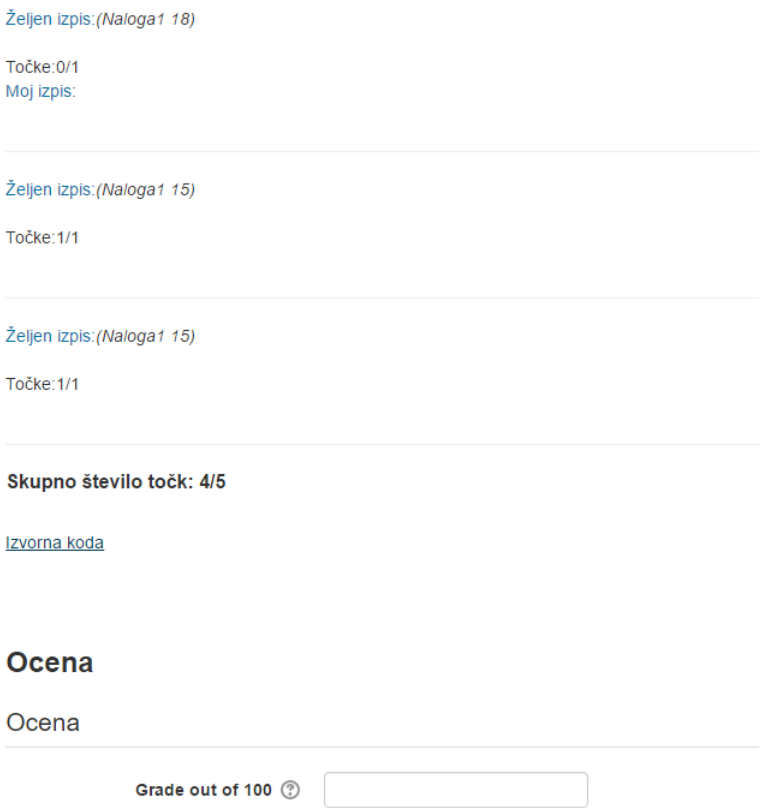
Ko imamo dodane rešitve za določeno nalogo, lahko sistem na podlagi teh datotek preveri, ali je program pravilno opravil z določeno nalogo. Vse kar mora študent narediti je, da odda javanski program, kjer je ime razreda enak imenu, ki ga je navedel profesor. Če se naloga ne prevede, sistem izpiše napako in prikaže podrobnosti te napake. Če pa se naloga prevede, sistem za preverjanje pravilnosti za vsak test odgovori ali je bil test uspešen ali ne. Na zaslon izpiše, kakšen je bil vhod in kakšen je bil željen izhod. Izhod je prvotno skrit, lahko pa ga poljubno prikazujemo in skrivamo. Če je test napačen, se pokaže tudi izhod študentove rešitve. Tudi ta izhod se lahko skrije ali prikaže. Prikaže se tudi število osvojenih točk, ki jih je prejel študent za svoj program. Na koncu strani imamo tudi možnost prikaza celotne kode programa, ki ga je oddal študent. Del forme je prikazan na sliki 4.3.



Slika 4.3: Forma za testiranje

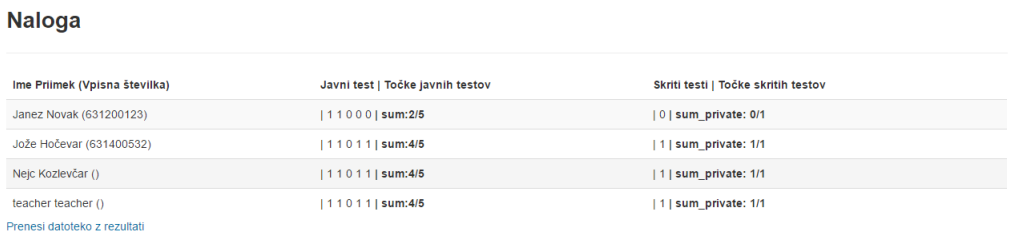
4.1.4 Prikaz točk za študente

Profesor lahko študentske rezultate pregleda na dva načina. Ko odpre stran za ocenjevanje, se profesorju izpiše seznam vseh vpisanih študentov. Če želi pogledati podrobnosti testov za določenega študenta, lahko to stori s klikom na „uredi oceno“. Sistem mu izpiše enako testno stran kot študentu. Vendar ima profesor poleg testne forme izpisan tudi status oddaje ter možnost ocene za študenta. (slika 4.4).



Slika 4.4: Oceni enega študenta

Drugi način je precej hitrejši. Profesor zgolj z enim klikom dobi rezultate vseh študentov v eni tabeli. Slika 4.5. Tabela je sestavljena iz treh delov. Prvi del so osebni podatki. V drugem delu so podatki o javnih testih. Najprej so izpisane točke vsakega testa posebej, nato pa še skupno število točk. Ta del je popolnoma enak kot datoteka, ki se zapiše, ko študent testira svojo nalogo. Ravno zato se ta datoteka tudi zapiše. Sistem lahko tako samo prebere vsebino te datoteke in mu ni potrebno ponovno poganjati vseh testov. Zadnji, tretji del, pa so podatki o skritih testih. Sestavljeni so enako kot javni testi. Na začetku so izpisane točke posameznih testov, na koncu pa so zapisane skupne točke skritih testov. Tudi skriti testi se izvajajo s pomočjo programa Docker. Celotna tabela se zapiše v tekstovno datoteko, ki jo lahko shranimo na svoj računalnik.



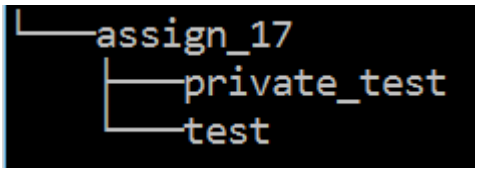
Slika 4.5: Test nad vsemi študenti

Pomemben dodatek našemu sistemu je pregled testov za vse vpisane študente, saj to profesorju prihrani veliko časa. Tako z enim klikom pridobi točke vseh študentov, ne da bi pregledoval vsako nalogo posebej. Pri testiranju se uporabijo tudi skriti testi, ki jih je profesor lahko dodal. Na zaslonu se v tabeli izpišejo podatki o študentu, rezultati javnih in rezultati skritih testov. Rezultat testov je sestavljen iz izpisa točk za vsak test posebej in skupnega števila točk. Če profesor želi shraniti podatke v datoteko, lahko to stori s klikom na povezavo na dnu strani. Primer prikaza ocen je prikazan na sliki 4.5.

4.2 Realizacija

Naš sodniški sistem je v večini realiziran s programskim jezikom PHP, ker je tudi celoten Moodle spisan v tem jeziku. Nekaj funkcionalnosti, ki se izvajajo na klientovi strani, so napisane v jezikih JavaScript in HTML. Pri izvajanju študentskih nalog si zaradi varnosti pomagamo s sistemom Docker.

Ko prvič kliknemo na dodajanje rešitev, se nam na strežniku ustvari nova mapa z imenom „assign_(id naloge)“. Znotraj te mape se ustvarita še 2 novi mapi „test“, kjer so shranjeni javni testi in „private_test“, kjer se nahajajo skriti testi. /slika 4.6).



Slika 4.6: Drevesna struktura testnih map

Ko se nam prikaže stran za dodajanje rešitev (slika 4.2), lahko naložimo rešitve na strežnik. Če so bile datoteke z enakim imenom na strežnik že dodane, jih sistem prepiše z novimi. Datoteke, ki so na strežniku, so izpisane na dnu strani. Poleg njih je rdeč križec, s katerim lahko datoteko odstranimo s strežnika. Brisanje datotek deluje s pomočjo metode AJAX. AJAX nam omogoči brisanje datotek, ne da bi pri tem rabili osvežiti stran.

```
1 function remove(file_name, assign_id) {
2     var req = false;
3     req = new XMLHttpRequest();
4     req.open("GET", "delete_file.php?id=" + assign_id + "&file="
      + file_name, true);
5     req.send(null);
6     return req;
7 }
```

Izsek kode 4.1: Brisanje datotek (JavaScript)

Brisanje je sestavljeno iz dveh delov. Prvi del se izvaja na klientovi strani in je napisan v programskem jeziku JavaScript. V tem delu pošljemo HTML zahtevo z dvema parametroma, identifikacija naloge in ime izbrisane datoteke. Parametra potrebujemo zato, da lahko naš program najde želeno datoteko za brisanje. Parametra se preneseta na PHP stran za brisanje. PHP skripta se izvaja na strežniku. Naš program najprej preveri, če se datoteka nahaja med javnimi testi. Če program datoteko najde, jo izbriše, drugače pa jo izbriše iz skritih testov.

```
1 $file_name = $_GET['file'];
2 $assignid = $_GET['id'];
3 $dir = "local/assign_$assignid/";
4 $testDir = $dir."test/";
5 if (!file_exists($testDir.$file_name)) {
6     $testDir = $dir."private_test/";
7 }
8
9 unlink($testDir.$file_name);
```

Izsek kode 4.2: Brisanje datotek (PHP)

Ko potrdimo izbrane teste, se te datoteke shranijo v PHP-jevo spremenljivko `FILES`. Nas zanimata, trenutna pot in ime datoteke. Z metodo `move_uploaded_file` premaknemo datoteko iz trenutne poti v prej ustvarjeno mapo. Metodo uporabimo na vseh izbranih datotekah. Ko se postopek kopiranja datotek zaključi, nas sistem preusmeri na začetno stran naloge.

```
if (isset($_FILES['upload_test']['tmp_name'])) {
    $tmp_name = $_FILES['upload_test']['tmp_name'];
}

if (isset($tmp_name)) {
    for ($i = 0; $i < $count; $i++) {
        if (!empty($name[$i])) {
            if (move_uploaded_file($tmp_name[$i], $testDir .
    $name[$i])) {

            } else {
                die();
            }
        }
    }
    header("Location: $_SERVER[PHP_SELF]?id=$assignid&action=
    view");
}
```

Izsek kode 4.3: Koda, ki shrani datoteke na željeno mesto

Ko so testi na strežniku, lahko študenti začnejo z oddajo in testom svojih nalog. Študent na prvi strani naloge vidi nov gumb „Preveri nalogo“. S klikom nanj ga sistem preusmeri na naslednjo stran, kjer se mu odpre možnost za oddajo naloge.

Preusmerjanje med različnimi stranmi v Moodlu deluje s pomočjo GET spremenljivke „action“. URL za prikaz prve strani naloge je naslednji: `http://localhost/moodle/mod/assign/view.php?id=8&action=view`. Za prikaz strani z ocenami, se v URL-ju spremeni zgolj spremenljivka „action“. Naslov te strani pa je: `http://localhost/moodle/mod/assign/view.php?id=8&action=grading`. Preusmerjanje po naših straneh deluje na enak način. Del kode ki skrbi za to je prikazan na izseku kode 4.4

```
...
} else if ($action == 'viewsubmitforgradingerror') {
    $o .= $this->view_error_page(get_string('submitforgrading',
    'assign'), $notices);
} else if ($action == 'checkersubmission') {
    $o .= $this->view_checker_submission_page();
} else if ($action == 'tester') {
    $o .= $this->view_tester_submission_page();
} else if ($action == 'adminuploader') {
    $o .= $this->view_admin_uploader_page();
}
```

Izsek kode 4.4: Preusmerjanje v Moodlu

Spremenljivnka `$o` predstavlja HTML kodo, ki se izpiše v brskalniku. V metodi `view_admin_uploader_page()` je napisano, kaj naj določena stan vsebuje. Primer metode vidimo na izseku kode 4.5. Spodnja koda nam prikaže Moodlovo glavo strani, ustvari in prikaže našo formo za oddajanje datotek in doda nogo na spletni strani.

```
...
$o .= $this->get_renderer()->render(new assign_header($this->
    get_instance(),
     $this->get_context(),
     $this->show_intro(),
     $this->get_course_module()->id,
     $title));

$mform = new admin_upload_form();

$o .= $this->get_renderer()->render(new assign_form('
    admin_upload', $mform));
$o .= $this->view_footer();
```

Izsek kode 4.5: Primer prikaza strani v Moodlu

Vsaka Moodlova forma mora biti razširjena iz razreda `moodleform`. Nato je potrebno implementirati abstraktno funkcijo `definition`. Znotraj te funkcije povemo, kaj naj se prikaže na formi. To storimo z metodo `addElement`.

Primer preprostega izpisa je prikazan na izseku kode 4.6.

```
1     $mform->addElement('html', "<p>Nalozi javne teste: </p>");
2     $mform->addElement('html', '<input name = "
      upload_files_public[]" type = "file" multiple = "multiple" />
      ');
3     $mform->addElement('html', '<br><hr>');+
4     $mform->addElement('html', "<p>Nalozi skrite teste, ki se
      pozenejo pri ocenjevanju: </p>");
5     $mform->addElement('html', '<input name = "
      upload_files_private[]" type = "file" multiple = "multiple"
      />');
6     $mform->addElement('html', '<br><hr>');
7     $mform->addElement('html', '<input type = "submit" value = "
      Shrani resitve" >');
```

Izsek kode 4.6: Primer prikaza strani v Moodlu

Pri prvem preverjanju se na strežniku ustvari nova mapa. Znotraj mape „asign_id“ se ustvari mapa z imenom „user_(id uporabnika)“. Tako imamo za vsakega uporabnika, ki odda nalogo, svojo mapo. Zaradi varnosti se naloge študentov izvajajo s pomočjo Dockerja. Zato potrebujemo datoteko z imenom „Dockerfile“, kjer definiramo, kako se bo naša slika Dockerja obnašala. Izsek kode 4.7.

```
1 FROM java:8
2
3 ARG file1
4 ARG assign
5
6 COPY local/$assign /$assign/
7
8 RUN javac $file1 2> error.txt ;exit 0
9 RUN if [ -z $(cat error.txt  &> /dev/null ) ] ; then echo empty;
      else exit 1; fi
10
11 ENTRYPOINT ["java"]
```

Izsek kode 4.7: Vsebina Dockerfile datoteke

Prva vrstica `FROM java:8`, nam pove, da bomo uporabili knjižnico java. Z ukazom ARG ustvarimo spremenljivki, ki jih definiramo pri gradnji Docker slike. To naredimo z naslednjim ukazom:

```
shell_exec("sudo docker build --build-arg file1=$FILE_NAME --
    build-arg assign=assign_$assignid --no-cache -t moodle .");
```

Izsek kode 4.8: Ukaz za gradnjo slike

Spremenljivka $FILE_NAME je ime poti do datoteke, ki jo hočemo pognati. Če ima ukaz izhodni status enak 1 pomeni, da je prišlo pri prevajanju programa do napake. Ukaz COPY kopira vsebino prvega argumenta na lokacijo drugega argumenta. Pri tem je prvi argument dejanska pot na strežniku, druga pot pa je pot znotraj Docker slike. Naslednja 2 ukaza RUN izvedeta prevajane javanskega programa. Drugi RUN ukaz nato vrne status 1, če je prišlo do napake. Zadnji ukaz ENTRYPOINT nam pove, kateri ukaz se bo izvedel, ko bomo klicali ukaz `docker run`. Torej drugi Dockerjev ukaz, ki ga potrebujemo je:

```
sudo docker run -i -t moodle -cp assign_$assignid/user_$userid
    Naloga1 $input
```

Izsek kode 4.9: Ukaz za izvedbo Dockerja

Ta vrstica izvede ukaz, ki smo ga definirali z rezervirano besedo ENTRYPOINT in kot argument temu doda argumente zgornjega ukaza. V našem primeru so argument -cp, assign_$assignid/user_$userid, Naloga1 in $input. Zgornja funkcija vrne izhod našega programa.

Po izvedenem zgornjem ukazu dobimo izhod naloge, ki ga lahko preverimo s pravilno rešitvijo. Sistem nam za vsak test izpiše kakšen je vhod v test in kakšen je željen izhod. Če se rešitev in izhod programa ujemata, dobimo za ta test 1 točko. V nasprotnem primeru dobimo 0 točk, sistem pa nam izpiše tudi naš izpis. Z ukazom `shell_exec` dobimo izpis študentove naloge z vhodnimi parametri. Ta izpis shranimo v spremenljivko `output`. V spremenljivko `solution` pa se shrani rešitev testa. Rešitev dobimo tako, da preberemo vsebino priložene testne datoteke. To storimo z metodo `file_get_contents`. Izsek kode 4.10

```
$output = shell_exec($executable);
$solution = file_get_contents($dir . $FILE_SOLUTION);
$mform->addElement('html', "Tocke:");
if ($output == $solution) {
    $mform->addElement('html', "1/1 <br><br>");
    fwrite($grade, "1 ");
    $my_points++;
} else {
    $mform->addElement('html', "0/1<br>");
    fwrite($grade, "0 ");
    $mform->addElement('html', "<a href=\"javascript:toggle('
    $div_output');\">Moj izpis: </a><br>");
    $mform->addElement('html', "<pre>$output</pre></div><br>");
}
```

Izsek kode 4.10: Preverjanje pravilnosti testov

Na naši formi lahko rešitev testa skrijemo ali prikažemo, kar nam omogoča JavaScript funkcija `toggle`. Vse kar ta funkcija stori je, da spremenil lastnost „display“ v „none“, če hočemo da je vsebina skrita, oziroma v „block“, če hočemo da je vsebina prikazana.

```
window.toggle = function (showHideDiv) {
    var ele = document.getElementById(showHideDiv);
    if (ele.style.display == "block") {
        ele.style.display = "none";
    }
    else {
        ele.style.display = "block";
    }
};
```

Izsek kode 4.11: Funkcija toggle

Ko se vsi testi izvedejo, se nam podrobnosti testov zapišejo v tekstovno datoteko. Tekst je sestavljen iz treh delov. Prvi del so podatki študenta, drugi del so točke za vsak test posebej, zadnji tretji del pa prikazuje število doseženih točk. Vsebina te datoteke je naslednja:

```
Janez Novak (631200123) | 1 1 0 0 0 | sum:2/5
```

Poglavje 5

Zaključek

V okviru diplomskega dela je nastal uporaben sistem za pomoč študentom in profesorjem na fakulteti. Sistem občutno zmanjša zahtevnost preverjanja programerskih nalog, tako za profesorje kot tudi za študente. Študent takoj dobi okvirno oceno izdelka, profesor pa lahko enostavno preveri pravilnost nalog za vse študente. Ker se sistem uporablja v spletni učilnici, ga ni pretirano težko usvojiti, saj se z učilnico srečujemo vsak dan.

Pri sami realizaciji je bilo potrebno posebej paziti na varnost sistema, saj bi lahko kakšna varnostna luknja predstavljala zelo veliko težavo za celotno spletno učilnico.

Naš sodniški sistem za preverjanje nalog ima še veliko možnosti za nadgradnjo, saj je šele nastal. Trenutna verzija sistema podpira možnost preverjanja nalog zgolj za programski jezik java. V prihodnosti bi lahko podpiral testiranje še kakšnega drugega programskega jezika in tako študentom omogočil pisanje nalog v poljubnem jeziku.

Literatura

[1] S.S. Skiena and M.A. Revilla. *Programming Challenges: The Programming Contest Training Manual*. Texts in Computer Science. Springer New York, 2006.

[2] Wikipedia. Spoj. `https://en.wikipedia.org/wiki/SPOJ`. [Dosegljivo; dostop 28 August, 2016].

[3] H. Leung, F. Li, R. Lau, and Q. Li. *Advances in Web Based Learning - ICWL 2007: 6th International Conference, Edinburgh, UK, August 15-17, 2007, Revised Papers*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008.

[4] SPOJ. Spoj navodila. `http://www.spoj.com/problems/TEST/`. [Dosegljivo; dostop 28 August, 2016].

[5] SPOJ. Spoj oddaja. `http://www.spoj.com/submit/TEST/`. [Dosegljivo; dostop 28 August, 2016].

[6] SPOJ. Spoj status. `http://www.spoj.com/status/TEST,n3jk0/`. [Dosegljivo; dostop 28 August, 2016].

[7] CodeChef. Codechef. `https://www.codechef.com/aboutus`. [Dosegljivo; dostop 28 August, 2016].

[8] CodeChef. Codechef urejevalnik besedila. `https://www.codechef.com/ide`. [Dosegljivo; dostop 28 August, 2016].

[9] CodeChef. Codechef status testov. https://www.codechef.com/status/TEST,n3jk0. [Dosegljivo; dostop 28 August, 2016].

[10] TopCoder. Topcoder. https://www.topcoder.com/about/. [Dosegljivo; dostop 28 August, 2016].

[11] TopCoder. Topcoder arena. https://arena.topcoder.com/#/u/practiceCode/1331/2110/2272/2/1331. [Dosegljivo; dostop 28 August, 2016].

[12] OTS. Delavanice ots. http://www.ots.si/delavnice/. [Dosegljivo; dostop 27 August, 2016].

[13] K. Matthias and S.P. Kane. *Docker: Up & Running*. O'Reilly Media, 2015.

[14] P. Raj, J.S. Chelladhurai, and V. Singh. *Learning Docker*. Packt Publishing, 2015.

[15] Opensource. Docker. https://opensource.com/resources/what-docker. [Dosegljivo; dostop 29 August, 2016].

[16] Opensource. Docker security. https://opensource.com/business/14/9/security-for-docker. [Dosegljivo; dostop 29 August, 2016].

[17] Wikipedia. Php. https://en.wikipedia.org/wiki/PHP. [Dosegljivo; dostop 19 August, 2016].

[18] Wikipedia. Php. https://en.wikipedia.org/wiki/PHP#Release_history. [Dosegljivo; dostop 19 August, 2016].

[19] K. Tatroe, P. MacIntyre, and R. Lerdorf. *Programming PHP*. O'Reilly Media, 2013.

[20] Moodle. Moodle strani. https://moodle.net/sites/. [Dosegljivo; dostop 18 August, 2016].

[21] Moodle. Moodle slovenija. http://www.fm-kp.si/zalozba/ISSN/1854-4231/2_267-272.pdf. [Dosegljivo; dostop 19 August, 2016].

[22] Moodle. Zahteve za inštalacijo. https://docs.moodle.org/22/en/Installing_Moodle#Requirements. [Dosegljivo; dostop 29 August, 2016].

[23] W. Rice. *Moodle 2.0 E-Learning Course Development*. Community experience distilled. Packt Pub., 2011.

[24] Frank Appel. *Testing with JUnit*. PACKT Publishing, 2015.