Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jurij Drobnič

Testiranje spletne aplikacije za organizacijo tekmovanj z orodjem Selenium

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana 2014

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: 00456 / 2013
Datum: 12.4.2013

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

Kandidat: **JURIJ DROBNIČ**

Naslov: **TESTIRANJE SPLETNE APLIKACIJE ZA ORGANIZACIJO TEKMOVANJ Z ORODJEM SELENIUM**
TESTING WEB APPLICATION FOR TOURNAMENT MANAGEMENT WITH SELENIUM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi predstavite sistematičen razvoj celovite spletne aplikacije za organizacijo tekmovanj, ki omogoča upravljanje z različnimi tekmovanji, igrami in udeleženci tekmovanj. Ob razvoju prikažite testiranje spletne aplikacije z orodjem Selenium, pri čemer izpostavite načrtovanje testiranja, pripravo primernega testnega okolja ter samo izvedbo testiranja.

Mentor:

viš. pred. dr. Igor Rožanc

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Jurij Drobnič, z vpisno številko **63070074**, sem avtor diplomskega dela z naslovom:

Testiranje spletne aplikacije za organizacijo tekmovanj z orodjem Selenium

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
- 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, dne 6. februarja 2014 Podpis avtorja:

Zahvaljujem se viš. pred. dr. Igorju Rožancu za pomoč in mentorstvo pri izdelavi diplomskega dela. Zahvaljujem se tudi družini za podporo pri mojem študiju, Andreju Komanu pa za dobre ideje in nasvete pri izdelavi naloge.

Kazalo

Povzetek

Diplomska naloga obsega razvoj spletne aplikacije in njeno testiranje z avtomatskimi testi Selenium. Selenium je zbirka orodij za avtomatizacijo testiranja aplikacij v spletnih brskalnikih. Sestavlja ga Selenium IDE, Selenium Remote control, Selenium Webdriver, Selenium Grid in Selenium Grid 2. Ljudje igramo različne igre in v njih tudi tekmujemo. Zato je bila razvita aplikacija Tournament Organizer, ki omogoča upravljanje in organizacijo tekmovanj.

Diplomska naloga zajema pregled in razlago uporabljenih tehnologij, načrtovanje aplikacije, specifikacijo zahtev ter prikaz posameznih modulov aplikacije Tournament Organizer. Testiranje aplikacije obsega definicijo ciljev testiranja, načrtovanje in implementacijo testnega projekta, kjer so vsi avtomatski testi za testiranje naše aplikacije. Jenkins je orodje za sprotno integracijo pri razvoju programske opreme. Za lažji razvoj naše aplikacije in avtomatizacijo ponavljajočih se nalog smo ga vključili na koncu in povezali vse naše delo v celoto. V zaključku je analiza testiranja in sklepne ugotovitve.

Ključne besede: Selenium, organizacija tekmovanj, Jenkins, spletna aplikacija.

Abstract

This thesis consists of web application development and its testing. Testing automated with Selenium framework. Selenium is a group of tools for automating web application testing. Tools that are in Selenium framework are Selenium IDE, Selenium Remote Control, Selenium WebDriver, Selenium Grid and Selenium Grid 2.

People like to play games and also compete in them. That is why application Tournament Organizer for tournament management and organization was developed. Thesis consists of technology overview and explanations, application development planning, application specification overview and description of all the modules of the application. Testing portion of this work starts with our goals we want to achieve with testing. After that we look into test planning and test project implementation. Jenkins is a continuous integration tool. We used it to automate repetitive tasks and combined both our project into single administration point. At the end we analyze our testing results and finish with finishing thoughts.

Keywords: Selenium, tournament management, Jenkins, web application.

Poglavje 1

Uvod

Količina programske opreme, s katero se ljudje srečujemo, se iz dneva v dan povečuje. Programska oprema je prisotna na skoraj vseh področjih. Ena izmed pomembnejših lastnosti programske opreme je kakovost. Visoka kakovost je zelo pomembna, saj lahko napake povzročijo velike finančne izgube ali celo izgube življenj. Kakovost lahko dosežemo le s pravilnim pristopom k razvoju programske opreme. Del tega procesa je testiranje. Če si testiranje dobro zastavimo, lahko število napak v progamu zelo znižamo, vseh napak pa na žalost skoraj nikoli ne bomo odkrili.

Da smo lahko prikazali testiranje aplikacije z ogrodjem Selenium smo v diplomski nalogi razvili aplikacijo za organizacijo tekmovanj `Tournament Organizer`. Omogoča nam dodajanje, urejanje in brisanje iger, igralcev, ekip ter tekmovanj. Igralce lahko povezujemo v ekipe in jih prijavljamo na tekmovanja. Aplikacija je napisana v `Javi EE` in teče na strežniku `JBoss AS 7`. Vsi podatki aplikacije se shranjujejo v podatkovno bazo `MySql`. Za lažji in hitrejši razvoj aplikacije smo uporabili `JavaServer Faces 2` tehnologijo v povezavi s knjižnico `Primefaces`. Da bi zagotovili čimbolj pravilno delovanje aplikacije smo razvili testni projekt, ki vsebuje teste, ki preverijo funkcionalnost aplikacije. Za testiranje smo uporabili ogrodje Selenium, za testiranje spletnih aplikacij, ki posnema uporabo aplikacije kot jo vidi končni uporabnik. Tudi naš testni projekt Selenium je napisan v programskem jeziku `Java`,

vendar ne potrebuje strežnika za poganjanje.

V prvem delu diplomske naloge smo opisali tehnologije in orodja, ki smo jih uporabili pri razvoju aplikacije Tournament Organizer ter pri testnem projektu. Predstavili smo programski jezik *Java*, *JavaServerFaces 2*, knjižnico *Primefaces*, strežnik *JBoss AS 7*, podatkovno bazo *MySQL*, orodje za upravljanje s podatkovno bazo *MySql Toad for MySQL*, orodje *Maven*, *Apache Subversion*, *TortoiseSVN*, brskalnik *Firefox* z vtičnikom *Firebug*, testno ogrodje *JUnit*, ogrodje *Selenium* ter orodje za sprotno integracijo *Jenkins*.

V drugem delu naloge pa smo se posvetili načrtovanju in razvoju aplikacije za organizacijo tekmovanj. Razložili smo vso funkcionalnost, ki jo vsak modul aplikacije podpira ter malo bolj podrobno opisali strukturo projekta aplikacije. Nato smo na kratko predstavili testiranje programske opreme na splošno.

Temu sledi zadnji del diplomske naloge, ki vsebuje načrtovanje testiranja aplikacije Tournament Organizer. V tem poglavju smo opisali testne scenarije, ki smo jih želeli z našimi Selenium testi pokriti. Sledi pregled vseh petih paketov testnega projekta. Zadnje poglavje govori o povezavi obeh projektov z orodjem za sprotno integracijo Jenkins, ter generiranju poročil testiranja.

Sledi zaključek, ki pokaže ugotovitve in opredeli možnosti za nadaljni razvoj in izboljšave.

Poglavje 2

Uporabljene tehnologije in orodja

2.1 Java

Java je objektno usmerjen programski jezik. Razvili so ga v podjetju Sun Mycrosystems [4]. Prva različica Jave je izšla leta 1995. Leta 2010 je podjetje Oracle kupilo podjetje Sum Microsystems [5], ki od takrat Javo tudi dopolnjuje. Programi napisani v javi so prenosljivi, kar pomeni, da lahko tečejo na različnih platformah. To je mogoče, ker se koda prevede v Java vmesno kodo (ang. bytecode) namesto da bi se prevedla v strojno kodo. Java se potem izvaja v Javanskem izvajalnem okolju (ang. Java virtual machine - JVM). Tako vsaka naprava, ki ima vgrajeno podporo za JVM lahko poganja programe, ki so napisani v jeziku Java. To pa seveda ni brez pomankljivosti. Ker gre izvajanje kode preko JVM so performanse slabše kot pri programskih jezikih, ki se direktno prevedejo v strojno kodo. Tej omejitvi se nekatere naprave izognejo, ker poganjajo javo na strojni implementaciji JVM-ja namesto na programski. Primer take uporabe so nekateri procesorji ARM [4].

Obstajajo tri različne implementacije Jave. Prva je Java SE, ki izvira iz prvotne implementacije jezika [6]. Vsebuje nekaj največkrat uporabljenih knjižnic, kot na primer *java.io* in *java.math*. Oracle ponuja dva paketa Jave

SE: Java JRE in Java JDK. Java JRE vsebuje vse potrebno, da lahko poganjamo aplikacije napisane v programskem jeziku Java, Java JDK pa je namenjena razvijalcem programske opreme saj vsebuje tudi nekatera potrebna orodja za razvoj aplikacij.

Druga implementacija je Java EE, ki razširja Javo SE. Ne gre le za programski jezik, ampak za bolj celovito platformo. Java EE vsebuje različne aplikacijske vmesnike (ang. API-je) (npr. vmesnik za povezavo s podatkovno bazo (ang. JDBC), spletne storitve, označevalni jezik XML) in pravila za njihovo medsebojno integracijo [7]. Java EE platformo se uporablja za poslovne aplikacije, saj te potrebujejo komunikacijo med različnimi podsistemi. Vse to upravlja aplikacijski strežnik (npr. GlassFish, JBoss ipd.).

Tretja implementacija pa je JavaME. Namenjena je za vgrajene sisteme (npr. moibilne naprave, digitalni sprejemniki). JavaME naprave imajo implementiranega enega od razpoložljivih profilov. Profili predstavljajo konfiguracijo. Profil Connected Device Configuration izvira iz Jave SE in vsebuje večino knjižnic. Okrnjena verzija profila CDC pa je Connected Limited Device Configuration in vsebuje le nujne knjižnice za delovanje [8].

2.2 JavaServer Faces 2.0 in Primefaces

`JavaServer Faces 2.0` je MVC ogrodje za izdelovanje aplikacij v programskem jeziku Java [1]. Omogoča hiter razvoj aplikacij, saj vsebuje veliko komponent, ki jih razvijalec lahko uporabi s poslovno logiko aplikacije. Tu gre izpostaviti predvsem veliko zbirko grafičnih elementov, podporo za tehnologijo AJAX ter validacijo vhodnih podatkov na odjemalcu. Obstaja veliko knjižnic, ki še razširijo funkcionalnost. Ena izmed teh je knjižnica Primefaces ki doda še dodatne grafične gradnike. Nekateri gradniki so recimo: koledar, različni grafi, drsniki, dialogi, vnosna polja... Značke (ang. tags), ki pripadajo knjižnici Primefaces imajo pred imenom značke črko `p` tiste, ki so že v JavaServer Faces pa s črko `f`. Na primeru lahko vidimo kombinacijo značk JSF in značk Primefaces:

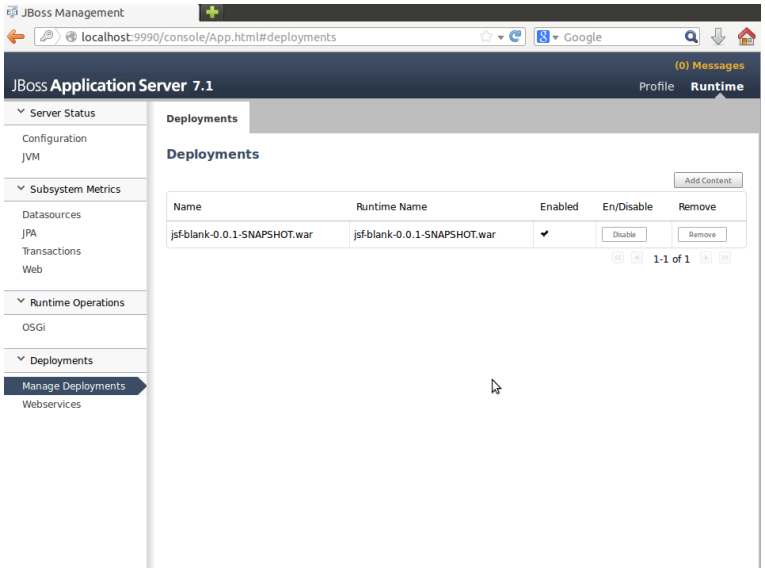
```
<p:inputText label="Name" id="name" required="true" >
  <f:validateLength minimum="2" maximum="50" />
</p:inputText>
```

2.3 Strežnik JBoss AS 7

Strežnik `Jboss AS 7` je aplikacijski strežnik, ki ga razvija podjetje Red Hat. Strežnik je namenjen Java EE aplikacijam. Obstaja zmogljivejša plačljiva verzija strežnika vendar je za naše potrebe zadostovala odprtokodna različica. Strežnik lahko teče v dveh načinih: načinu z enim strežnikom (ang. standalone) ter v načinu, kjer lahko več strenikov povežemo v gručo (ang. domain) [9]. Standalone je namenjen aplikacijam, ki bodo uporabljale le en strežnik. Domain pa označuje način, kjer imamo več sistemov povezanih v gručo in tako porazdelimo obremenitev strežnikov, da lahko zagotovimo kvalitetno in odzivno storitev za veliko uporabnikov. V diplomski nalogi smo uporabili način standalone. S strežnikom lahko upravljamo preko ukazne vrstice vendar je za enostavna opravila, kot so menjava aplikacije z novo verzijo, zelo uporaben tudi grafični vmesnik [10], ki ga prikazuje slika 2.1.

2.4 MySQL in Toad for MySQL

MySQL je odprtokodni sistem za upravljanje z relacijskimi podatkovnimi bazami [2]. Projekt je trenutno v lasti podjetja Oracle. MySQL je pogosta izbira za hranjenje podatkov pri spletnih aplikacijah. Uporabljajo ga tudi večje aplikacije, ko so: Wordpress, Joomla, Twitter, Google in drugi [2]. MySQL je privzeto možno uporabljati le iz ukazne vrstice saj nima grafičnega načina. Ker pa je včasih lažje delati v grafičnem načinu, so bile razvite aplikacije, ki omogočajo grafični prikaz za MySQL. Ena izmed njih je Toad For MySQL. Omogoča upravljanje s povezavami na več podatkovnih baz, kreiranje tabel, izvajanje SQL stavkov... Omogoča tudi grafični prikaz podatkov v podatkovni bazi, da lahko brez poizvedbe pregledamo vsebino tabel.



Slika 2.1: Spletni vmesnik strežnika JBoss AS 7

2.5 Maven

Maven je orodje, ki omogoča nastavljanje projekta iz enotne datoteke *pom.xml* [11]. Preko te datoteke se upravlja z vsemi knjižnicami in njihovimi verzijami, ki so uporabljene v projektu. Tako je zelo enostavno spremeniti verzijo knjižnice ki jo uporabljamo saj le spremenimo ta podatek v *pom.xml*. Prav tako lahko tu nastavljamo celoten proces izgradnje programskega paketa. Vsak maven projekt gre pri izgradnji paketa čez naslednje zaporedje postopkov [12] izgradnje:

- preverjanje (ang. validate)
- prevajanje (ang. compile)
- testiranje (ang. test)
- pakiranje (ang. package)
- integracijsko testiranje (ang. integration test)

- potrditev (ang. verify)
- nalaganje v lokalni repozitorij (ang. install)
- prenos kode (ang. deploy)

Primer dodajanja knjižnice v datoteko *pom.xml*:

```
<dependency>
  <groupId>org.primefaces</groupId>
    <artifactId>primefaces</artifactId>
  <version>4.0-SNAPSHOT</version>
</dependency>
```

2.6 Apache Subversion in TortoiseSVN

Apache Subversion (v nadaljevanju SVN) je orodje za nadzor nad različicami programske opreme [3]. Gre za repozitorj, kjer shranjujemo datoteke in imamo dostop do zgodovine teh datotek. Če želimo pregledati ali uporabiti starejšo različico datoteke lahko to storimo brez ovir. Repozitorij istočasno uporablja več razvijalcev saj sistem poskrbi, da je vsebina in stanje datotek vedno pravilno [3]. V primeru, ko dve osebi poskusita shraniti svoji spremembi, ki se medsebojno izključujeta nas sistem o tem obvesti, da lahko ročno razrešimo neskladje (ang. conflict). Zaradi lažjega dela z orodjem SVN je bilo razvito tudi orodje TortoiseSVN, ki se integrira v Windows Explorer [20]. S tem nam ni potrebno upravljati s SVNjem preko ukazne vrstice ampak grafično, kar je ob velikem številu datotek lahko bolj pregledeno. TortoiseSVN je na voljo pod GNU General Public License [21].

2.7 Brskalnik Firefox in vtičnik Firebug

Mozzila Firefox je odprtokodni spletni brskalnik [22]. Razvija in posodablja ga podjetje Mozzila. Brskalnik je podprt na operacijskih sistemih Windows,

Mac OS X, Linux in Android [13]. Na voljo je veliko vtičnikov, ki razširijo funkcionalnost.

Vtičnik Firebug razširi funkcionalnost brskalnika Mozzila Firefox [23]. Omogoča pregled strukture spletne strani in urejanje stilov spletnih elementov v realnem času. V vtičnik sta vključena tudi konzola za razhroščevanje javascripta in možnost analiziranja nalagalnih časov spletne strani.

2.8 JUnit

JUnit je eno izmed testnih ogrodij za testiranje enot v programskem jeziku Java [14]. Razvila sta ga Kent Beck in Erich Gamma [14]. Spada v skupino tesnih ogrodij xUnit. JUnit in vsa ogrodja iz te skupine so namenjena testiranju majhnih delov kode, ki jim pravimo enote [15]. Sem spada testiranje razredov in funkcij. Test je sestavljen iz več korakov. Najprej se izvede funkcija, ki je označena z `@BeforeClass`. Ta se izvede pred vsem, kar je v testnem razredu, izvede pa se le enkrat. Sledi izvajanje testov. Pred vsakim testom se izvede funkcija označena z `@Before`, ko se test izvede, pa se sproži še funckija označena z `@After`. Funkcija, ki predstavlja test, je označena z notacijo `@Test`, mora biti javna in ne sme vračati vrednosti. Na koncu vseh izvedenih testov se izvede funkcija označena z `@AfterClass`. Če želimo test začasno umakniti iz cikla testov lahko pred testom uporabimo navodilo `@Ignore` in tako test preskočimo. V test ni potrebno vključiti vseh korakov. Če imamo zelo enostaven testni primer lahko uporabimo le `@Test` fazo. V diplomski nalogi smo uporabili zadnjo različico ogrodja JUnit 4.

Koraki izvajanja testov v JUnit so:

- `@BeforeClass`
- `@Before`(za vsak test)
- `@Test`
- `@After`(za vsak test)

- `@AfterClass`

2.9 Selenium

Selenium je zbirka več orodij, ki omogočajo avtomatizacijo testiranja spletnih aplikacij [18]. V paket Selenium spadajo: Selenium WebDriver, Selenium Remote Control, Selenium IDE, Selenium Grid in Selenium Grid 2.

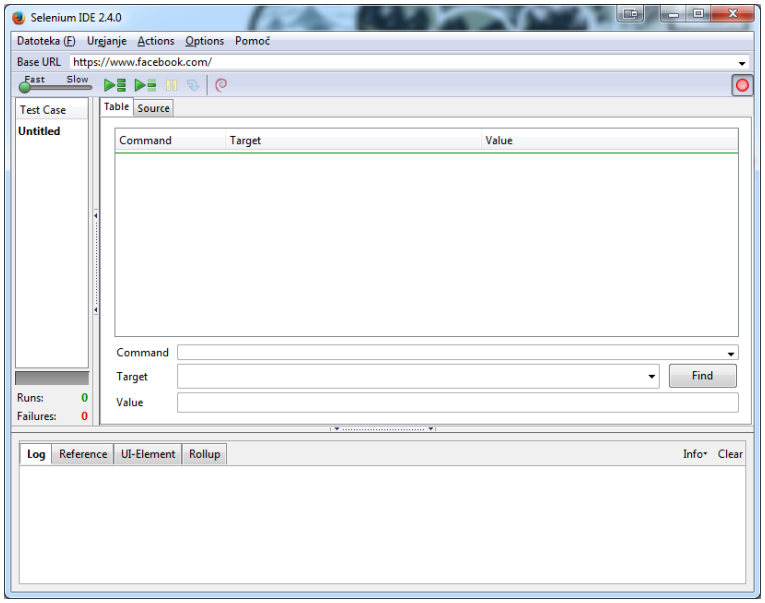
Začetki projekta Selenium segajo v leto 2004, ko je Jason Huggins za potrebe testiranja svoje spletne aplikacije razvil javascript knjižnico [16]. Knjižnica mu je omogočala, da je preko nje komuniciral s spletno aplikacijo in brskalnikom ter tako izvajal različne testne scenarije v različnih spletnih brskalnikih. Ta knjižnica je postala jedro orodja Selenium Remote Control.

Zaradi omejitev javascripta se je kasneje razvila nova različica Seleniuma imenovana Selenium WebDriver, ki združuje projekt Selenium s projektom WebDriver. Glavna razlika med Selenium WebDriver in Selenium Remote Control je ta, da Selenium WebDriver namesto izvajanja javascripta komunicira z brskalnikom preko WebDriver APIja in se tako izogne omejitvam javascripta.

Selenium IDE je vtičnik za brskalnik Mozzila Firefox [18]. Omogoča snemanje uporabnikovih akcij (klikov in ostalih interakcij z elementi spletne aplikacije). Zaporedje akcij lahko preko Selenium IDE vtičnika tudi ponovo poženemo in tako zaporedje akcij izvedemo samodejno. Ena izmed zelo uporabnih funckij je tudi izvoz teh korakov v enega izmed podprtih jezikov in testnih ogrodij.

Podprti so naslednji jeziki:

- Ruby (RSpec/WebDriver, Test::Unit/WebDriver, Rspec/Remote Control, Test::Unit/Remote Control) [17]
- Python 2 (unittest/WebDriver, unittest/Remote Control) [17]
- Java (JUnit 4/WebDriver, JUnit 4/Remote Control, JUnit 3/Remote Control, TestNG/Remote Control) [17]



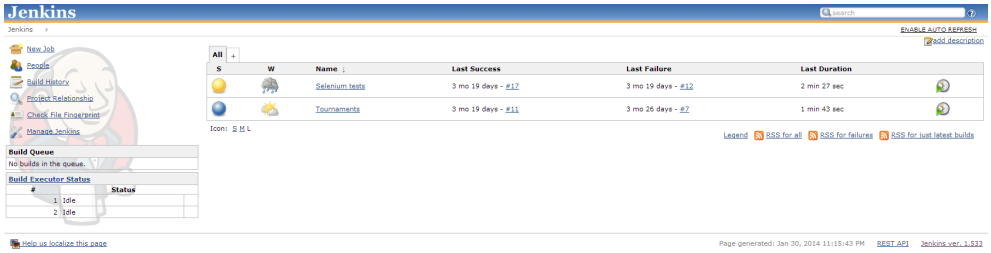
Slika 2.2: Okno orodja SeleniumIDE

- C# (Nunit/WebDriver, NUnit/Remote Control) [17]

Selenium IDE je prikazan na sliki 2.2. Je zelo uporabno orodje, vendar se ga uporablja večinoma za prototipiranje in ne za izdelavo večjih testnih scenarijev. Za demonstracijo lahko s tem vtičnikom naredimo test v nekaj minutah, ga izvozimo v želen jezik in poženemo. Selenium IDE je zelo dober začetek za nekoga, ki se prvič ukvarja s tovrstnim testiranjem.

2.10 Jenkins

Jenkins je odprtokodno orodje za sprotno integracijo [24]. Skupna integracija označuje postopek v razvoju programske opreme, kjer se na enem mestu zbere in poveže delo vseh razvijalcev [25]. Jenkins je v programskem jeziku java in je naslednik znanega orodja Hudson [24]. Za orodje Jenkins je na voljo zelo veliko vtičnikov, ki omogočajo uporabniku, da si priredi orodje točno tako, kot želi. Dobra lastnost orodja je tudi ta, da podpira orodja za nadzor različic, med njimi tudi Subversion [24], ki smo ga uporabili v diplomski



Slika 2.3: Spletni vmesnik orodja Jenkins

nalogi. Jenkins nam služi kot stična točka konfiguracije strežnika, aplikacije in testiranja. Lahko poganja skripte, zgradi paket aplikacije z mavenom, pošilja aplikacijo na strežnik, poganja teste ter generira poročila. Jenkinsa uporabljamo preko spletnega vmesnika, ki je prikazan na sliki 2.3.

Poglavje 3

Testiranje programske opreme

Testiranje programske opreme je skupina praks, s katero zagotavljamo kakovost in pravilno delovanje programske opreme. S testiranjem odkrivamo napake v programski opremi, ki smo jih vnesli med razvojem. Nemogoče je pričakovati, da bomo s testiranjem odkrili vse napake lahko pa se njihovo število s pravilnim pristopom zniža. S testiranjem dokazujemo pravilnost delovanja v določenih pogojih. Ko govorimo o testiranju programske opreme, govorimo o testiranju programske kode. Ta del je zelo pomemben vendar obstaja tudi testiranje na drugih korakih pri procesu razvoja programske opreme. Napaka, ki jo naredimo že med načrtovanjem, je lahko zelo draga, saj so vsi nadaljni koraki zaradi tega napačni. Temu se izognemo s testiranjem na vseh nivojih procesa razvoja programske opreme.

3.1 Testiranje po principu bele in črne škatle

Če se osredotočimo na testiranje kode, lahko ločimo dva pristopa testiranja. Prvi pristop je testiranje bele škatle, drugi pa testiranje črne škatle. Pri testiranju po principu bele škatle testiramo notranjo strukturo in funkcionalnost programske kode [19]. Za to je potrebno poznavanje programskega jezika v katerem je koda napisana, kot tudi kaj koda počne. Znanje programiranja je potrebno za razvoj testov. Pristop bele škatle se najpogosteje uporablja pri

testiranju enot saj želimo, da z izbiro testnih podatkov pokrijemo vse poti izvajanja. Za zadostno pokritje kode moramo zelo dobro poznati kodo, ki jo testiramo.

Pri testiranju po pristopu črne škatle se osredotočimo na vprašanje kaj mora nek program narediti ne zanima pa nas kako to naredi. Ta pristop je uporaben pri testiranju programske opreme na nivoju, kot ga vidi in uporablja končni uporabnik [19]. Še ena dobra lastnost tega pristopa je ta, da človeku, ki testira ni potrebno poznati programskega jezika in strukture kode. V nekaterih primerih ni potrebno niti znanje programiranja.

Na prvi pogled spada testiranje spletnih uporabniških vmesnikov z orodjem Selenium v kategorijo črne škatle. Za zelo osnovno testiranje to drži, vendar se kaj hitro znajdemo v situaciji, da za bolj robustno testno okolje potrebujemo tudi nekaj znanja o aplikaciji, ki jo testiramo. Primer za to je, ko želimo po testu obnoviti podatkovno bazo v tako stanje, kot je bila pred testom. S tem zagotovimo, da je test vedno ponovljiv. Primer je recimo, ko v aplikacijo dodamo uporabnika z imenom Uporabnik1 in mora biti ime enolično, ga moramo po testu pobrisati iz podatkovne baze. Vse to bi lahko poenostavili tako, da v istem testu uporabnika dodamo in tudi pobrišemo, vendar nastane težava, ko nam zataji funkcionalnost brisanja uporabnika v aplikaciji. Tako vidmo, da je testiranje z orodjem Selenium v osnovi testiranje črne škatle vendar ga je zaradi robustnosti pametno razširiti z elementi testiranja bele škatle.

Poglavje 4

Razvoj aplikacije TournamentOrganizer

4.1 Ideja aplikacije

Ker se ljudje v prostem času radi zabavamo z igranjem iger in se v njih tudi radi pomerimo na tekmovanjih, smo se odločili razviti aplikacijo, ki nam omogoča organizacijo tekmovanj v različnih igrah.

Ideja je, da bi lahko preko aplikacije upravljali z igrami tekmovanj, igralci in ekipami, ki se tekmovanj udeležujejo, ter s tekmovanji. Če želimo, da bodo ljudje aplikacijo uporabljali mora na enostaven in intuitiven način omogočati dodajanje, brisanje in urejanje vseh elementov aplikacije. To smo dosegli z jasnim uporabniškim vmesnikom, pogovornimi okni za dodajanje, urejanje in povezovanje elementov med seboj, ter dobrimi opozorili o izvedenih akcijah in napakah.

Ker so nekatere igre ekipne, v drugih pa tekmujejo posamezniki, smo se odločili podpreti oba načina in razvili dva pogleda, enega za ekipna tekmovanja in drugega za tekmovanja posameznikov. Igralce in ekipe urejamo ločeno, vendar je mogoče igralce povezati v ekipe. Igralec lahko pripada le eni ekipi naenkrat. Poleg vseh akcij, ki jih uporabnik lahko izvaja je pomembna uprabniku prijazna predstavitev podatkov s katerimi dela. Vsi podatki so predsta-

vljeni v tabelah. Želeli smo, da ima uporabnik možnost hitrega dostopa do pomembnih informacij, zato smo implementirali tudi dodatno funkcionalnost pregleda podatkov. Sem spada pregled vseh tekmovanj, ki se jih bo udeležil določen igralec ali ekipa, pregled vseh udeležencev določenega tekmovanja ter vseh igralcev, ki so del določene ekipe. To je predstavljeno preko pogovornega okna, ki omogoča tudi odstranitev določene povezave. Primer za to je, ko pregledujemo igralce določene ekipe in opazimo da nekdo ne spada v to ekipo, ga lahko odjavimo iz ekipe kar preko tega pogovornega okna in tako prihanimo nekaj klikov in časa.

4.2 Specifikacija zahtev

Aplikacija Tournament Organizer je sestavljena iz šestih modulov. Modul, ki je uporabljen v vseh ostalih je namenjen navigaciji. Postavljen je na vrhu vsake podstrani aplikacije. Omogoča navigiranje po vseh ostalih modulih. Ostali moduli morajo podpreti funkcionalnost aplikacije. Omogočati morajo predstavitev vseh podatkov v obliki tabel. Vsak modul mora podpreti dodajanje, brisanje in urejanje podatkov, ki spadajo k njemu. Upravljanje z entitetami mora biti poenoteno. Dodajanje in urejanje poteka preko pogovornih oken, ki vsebujejo vsa potrebna polja, ki jih mora uporabnik izpolniti. Uporabnikove akcije morajo biti izvedene hitro in tabela podatkov se mora po spremembi posodobiti z novimi vrednostmi. Pregled vseh igralcev določene ekipe, pregled tekmovalcev nekega tekmovanja in pregled tekmovanj nekega igralca morajo biti podprti preko pogovornih oken, ki se odprejo pri zahtevi pregleda teh podatkov. Ta pogovorna okna morajo omogočati tudi gumbe za odstranitev vnosa v vsaki vrstici, za lažjo administracijo aplikacije. Aplikacija mora uporabnika stalno obveščati o izvedenih akcijah in možnih napakah. Na primer, če želi uporabnik izbrisati entiteto, ki je uporabljena v drugem modulu mora biti uporabnik o tem obveščen.

Aplikacijo razdelimo na šest modulov: navigacija, igre, igralci, ekipe, ekipna tekmovanja in tekmovanja posameznikov.

4.3 Načrtovanje modulov aplikacije

Aplikacija Tournament Organizer je namenjena administraciji organizacije tekmovanj. Zaradi enostavnosti in razširjenosti spletnih tehnologij smo načrtovali aplikacijo, ki bo aplikacija tekla na spletnem strežniku, uporabniki pa bodo do aplikacije dostopali preko spletnih brksalnikov. Tehnologije, ki smo jih uporabili, smo že predstavili v prejšnjih poglavjih. Če povzamemo, smo aplikacijo napisali v programskem jeziku `Java EE`, ki teče na aplikacijskem strežniku `JBoss AS 7`. Podatki se shranjujejo v podatkovni bazi `MySQL`.

4.3.1 Modul Navigacija

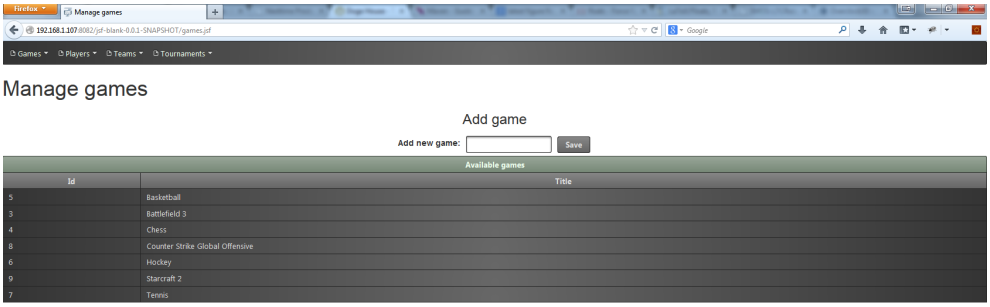
Modul navigacija predstavlja enostaven vrstični spustni menu, ki vsebuje gumbe do vseh ostalih modulov aplikacije. Postavljen je na vrh spletne strani in prisoten v vseh ostalih modulih.

4.3.2 Modul Igre

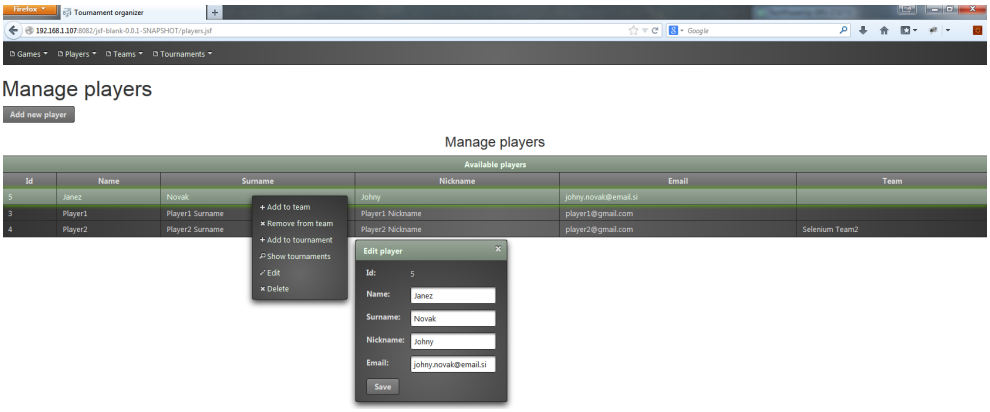
Modul Igre omogoča pregled vseh iger, ki so na voljo organizatorju tekmovanja. Igra je določena z naslovom igre. Igre je mogoče dodajati preko vnosne forme, ki je postavljena nad tabelo obstoječih iger. Igre je možno tudi brisati in urejati. Z desnim klikom na izbrano igro se nam odpre spustni menu na katerem lahko izbiramo med urejanjem in brisanjem igre. V primeru urejanja igre se nam odpre pogovorno okno v katerem spremenimo naslov igre in ga shranimo. O vseh izvedenih akcijah smo na zaslonu sproti obveščeni. Slika 4.1 prikazuje zaslon modula Igre.

4.3.3 Modul Igralci

Ta modul omogoča pregled vseh igralcev, ki smo jih dodali v sistem. Shranjeni so v tabeli. Vsak igralec je določen z imenom, priimkom, vzdevkom, naslovom email in imenom ekipe, če je v katero prijavljen. S klikom na gumb za dodajanje igralca se nam odpre pogovorno okno, v katerem izpolnimo po-



Slika 4.1: Zaslon modula `Igre`



Slika 4.2: Zaslon modula `Igralci`

datke o igralcu in ga tako shranimo v podatkovno bazo. Slika 4.2 prikazuje menu, ki se odpre ob desnem kliku na igralca ter formo za urejanje igralca. Desni klik na izbranega igralca nam odpre menu, ki vsebuje naslednje akcije:

Urejanje

Odpre se pogovorno okno, kjer spremenimo željeno vrednost.

Brisanje

Izbriše igralca iz podatkovne baze.

Dodajanje igralca v ekipo

Odpre se pogovorno okno, ki vsebuje menu z vsemi ekipami, ki so dodane v podatkovno bazo. Izberemo ekipo in shranimo spremembe.

Brisanje igralca iz ekipe

Odstrani igralca iz ekipe v katero je prijavljen.

Dodajanje igralca na tekmovanje

Odpre se pogovorno okno, ki vsebuje menu z vsemi tekmovanji posameznikov, ki so dodana v podatkovno bazo. Izberemo tekmovanje in shranimo spremembe.

Izpis vseh tekmovanj v katere je ta igralec prijavljen

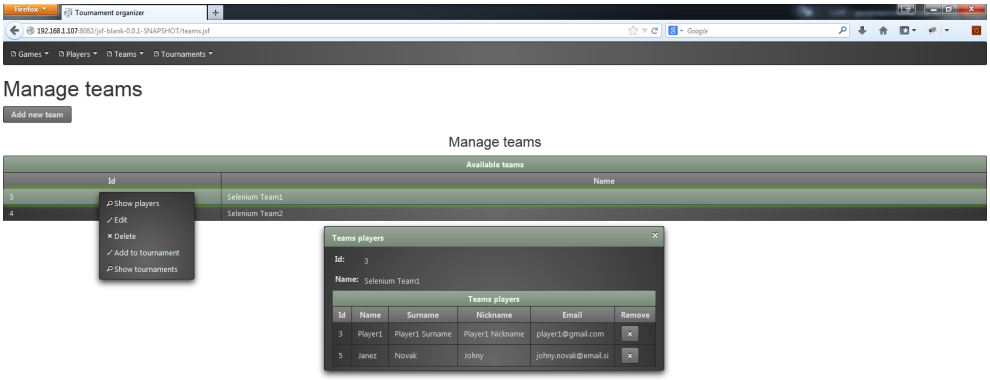
Odpre se pogovorno okno v katerem je tabela vseh tekmovanj na katere je igralec prijavljen. Ta pogled omogoča tudi odjavo igralca od tekmovanja s klikom na gumb odstrani, ki je prisoten za vsako tekmovanje.

4.3.4 Modul Ekipe

Modul omogoča pregled vseh ekip v sistemu. Enako kot igralci so tudi te predstavljene v tabeli. Vsaka ekipa je določena z imenom. S klikom na gumb za dodajanje ekipe se nam odpre pogovorno okno, ki ga izpolnimo z imenom ekipe in jo nato shranimo v podatkovno bazo. Desni klik na ekipo nam odpre meni, prikazan na sliki 4.3, z naslednjimi akcijami:

Izpis vseh igralcev ekipe

Odpre se pogovorno okno, ki nam pokaže vse igralce, ki so prijavljeni v to ekipo. Ta pogled omogoča tudi odstranitev določenega igralca iz ekipe, kot kaže slika 4.3.



Slika 4.3: Zaslon modula `Ekipe`

Urejanje

Odpre se pogovorno okno, kjer spremenimo ime ekipe.

Brisanje

Izbriše ekipo iz podatkovne baze.

Dodajanje ekipe na tekmovanje

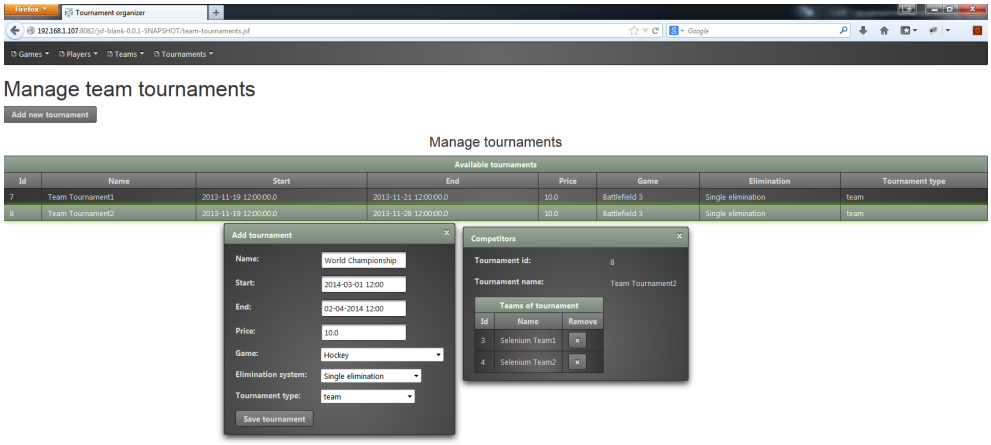
Odpre se pogovorno okno, ki vsebuje vsa ekipna tekmovanja. Izberemo želeno tekmovanje in shranimo.

Izpis vseh tekmovanj v katere je ekipa prijavljena

Izpiše vsa tekmovanja, na katera je ekipa prijavljena. Omogoča tudi odjavo od tekmovanja.

4.3.5 Modul Ekipna tekmovanja

Modul omogoča pregled vseh ekipnih tekmovanj. Predstavljena so v tabeli. Vsako tekmovanje je predstavljeno z imenom, datumom in uro začetka, datumom in uro konca, ceno, igro, načinom izločanja ter ali je tekmovanje ekipno ali za igralce. Tekmovanje dodamo preko vnosne forme, prikazane na sliki



Slika 4.4: Zaslon modula `Tekmovanja`

4.4 Desni klik na tekmovanje nam odpre menu, ki odpre menu z naslednjimi akcijami:

Urejanje

Odpre se pogovorno okno, ki omogoča urejanje atributov tekmovanja.

Brisanje

Izbriše tekmovanje.

Izpis vseh tekmovalcev

Odpre se pogovorno okno, ki v tabeli prikaže vse prijavljene ekipe, kot kaže slika 4.4. Ta pogled omogoča tudi odjavo ekip od tekmovanja.

4.3.6 Modul Tekmovanja posameznikov

Modul omogoča pregled vseh tekmovanj posameznikov. Predstavljena so v tabeli. Vsako tekmovanje je predstavljeno z imenom, datumom in uro začetka, datumom in uro konca, ceno, igro, načinom izločanja ter podatkom,

ali je tekmovanje ekipno ali za posameznike. Desni klik na tekmovanje nam odpre menu, ki odpre menu z naslednjimi akcijami:

Urejanje

Odpre se pogovorno okno, ki omogoča urejanje atributov tekmovanja.

Brisanje

Izbriše tekmovanje.

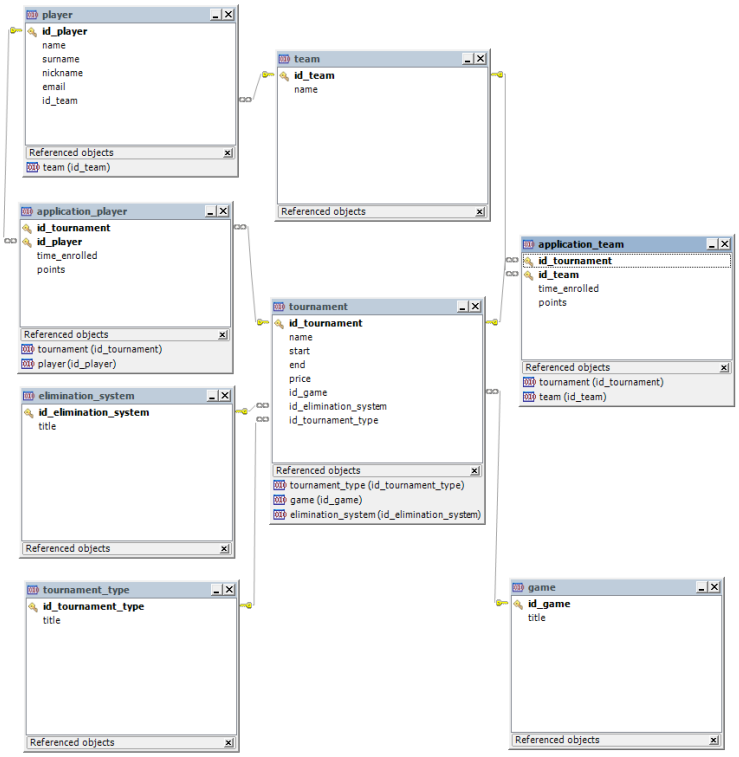
Izpis vseh tekmovalcev

Odpre se pogovorno okno, ki v tabeli prikaže vse prijavljene igralce. Ta pogled omogoča tudi odjavo igralcev od tekmovanja.

4.4 Podatkovni model in podatkovna baza

Vsi podatki aplikacije so shranjeni v eni relacijski podatkovni bazi, katere struktura je prikazana na sliki 4.5. Sestavljena je iz osmih tabel:

- `Game` predstavlja igre iz modula Igre.
- `Player` predstavlja igralca iz modula Igralci.
- `Team` predstavlja ekipo iz modula Ekipe.
- `Elimination_system` predstavlja strukturo tekmovanja.
- `Tournament_type` predstavlja možne vrste tekmovanja. V tabeli sta dva vnosa team in solo, prvi za ekipna tekmovanja drugi pa za solo.
- `Tournament` predstavlja tekmovanje iz modulov Ekipna in Solo tekmovanja.
- `Application_player` povezuje tabeli player in tournament. Predstavlja prijavo igralca na tekmovanje.



Slika 4.5: Podatkovni model

- `Application_team` povezuje tabeli team in tournament. Predstavlja prijavo ekipe na tekmovanje.

4.5 Struktura projekta

Projekt je zastavljen tako, da ima logiko aplikacije ločeno od predstavitve. Logika aplikacije je razdeljena v dva paketa, paket `beanDto` in paket `beanPackage`. Paket `beanDto` vsebuje razrede, ki predstavljajo entitete iz podatkovne baze, paket `beanPackage` pa funkcionalnosti modulov aplikacije. `AbstractBean` iz paketa `beanPackage` je sestavljen iz metod za uporabo podatkovne baze. Vsebuje metode za vstavljanje, spreminjanje, brisanje in vračanje vrednosti iz podatkovne baze in so uporabljene v vseh ostalih modulih. Ostali razredi iz `beanPackage` paketa podedujejo razred

`AbstractBean` in razširijo funkcionalnost potrebno za določen modul aplikacije. Paket `beanDto` vsebuje naslednje razrede: Elimination, Game, Player, Team, Tournament in TournamentType. Razredi paketa beanPackage so: `AbstractBean`, `EliminationBean`, `GameBean`, `PlayerBean`, `TeamBean`, `TournamentBean` in `TournamentTypeBean`.

Spletni uporabniški vmesnik je implementiran preko datotek s končnico `xhtml`. Te datoteka vsebujejo vse grafične elemente aplikacije. Za implementacijo komunikacije med strežnikom in odjemalcem je na zelo enostaven način poskrbljeno v samem ogrodju JSF 2. Ogrodje poskrbi za prikaz podatkovnih struktur, ki jih vrnejo razredi, povedati moramo samo, kje se bo določen podatek prikazal. To deluje tudi v obratno smer, kar pomeni da lahko določimo, v katerega od razredov se bo nek uporabnikov vnos shranil. Primer prikazuje vhodno polje, katerega vnos se shrani v atribut `name` objekta `team`, ki pripada razredu `TeamBean`.

```
<p:inputText id="name" value="#{teamBean.team.name}">
  <f:validateLength minimum="2" maximum="50" />
</p:inputText>
```

Ker želimo, da uporabnik vnaša pravilne vnose, smo dodali tudi preverjanje vnosnih polj v formah. Polja se preverja v skladu s tabelami v podatkovni bazi, lahko pa se vnos še dodatno omeji, če je to pomensko smiselno. Primer za to je vnosno polje za elektronsko pošto pri vnosu igralca v aplikacijo. V podatkovni bazi je predstavljeno kot niz znakov, vendar želimo, da vnos res predstavlja elektronski naslov. To rešimo z regularnim izrazom, ki je privzeto podprt v JSF 2.

```
<p:inputText id="email" value="#{playerBean.player.email}">
  <f:validateLength minimum="2" maximum="50" />
  <f:validateRegex pattern=".+\@.+\..+" />
</p:inputText>
```

4.5.1 Opozorila uporabniku

Za boljšo uporabniško izkušnjo smo implementirali tudi obvestila in opozorila, ki uporabniku sporočajo, katero akcijo je izvedel ter ga obvestijo o uspehu izvedbe. V primeru pravilne uporabe aplikacije služijo le kot pomoč, da uporabnik ve, katero akcijo je izvedel nazadnje. Opozorila pa so še bolj pomembna v primerih, ko uporabnik izvede akcijo, ki ni dovoljena, kar prikazuje slika 4.6. Opozorilo sporoči uporabniku, kaj je skušal storiti in zakaj akcija ni bila uspešno zaključena.

Slika 4.6: Opozorilo uporabniku ob napačni izpolnitvi forme

Poglavje 5

Testiranje aplikacije Tournament Organizer

5.1 Cilji testiranja

S testiranjem aplikacije želimo preveriti delovanje vseh modulov in povezav med njimi. Ker je pokritje čisto vseh možnih kombinacij uporabnikovih akcij nemogoče, smo si zadali cilj preveriti akcije vseh modulov in akcij, ki povezujejo module med seboj. Za celovito testiranje želimo razviti teste s pravilnimi vnosnimi podatki in teste, ki uporabljajo nepravilne vnose. Vsak modul mora imeti teste za vso funkcionalnost, ki jo podpira. Testi morajo biti zasnovani tako, da so ponovljivi. Po testiranju želimo, da se naredi poročilo, s katerega je razvidno kateri testi so se izvedli uspešno in kateri neuspešno.

5.2 Načrtovanje testiranja

Testiranje aplikacije Tournament Organizer smo pričeli z načrtovanjem testiranja. Načrtovati je bilo potrebno testne zgodbe, ki bi pokrile vso funkcionalnost aplikacije. Testne zgodbe so scenariji, ki poizkušajo zajeti zaporedje akcij, ki jih uporabniki aplikacije izvajajo. Testne zgodbe je potrebno načrtovati na dva različna načina, da zagotovimo kakovost. Prvi način je

načrtovanje testov, ki oponašajo pravilno uporabo programa. Pravilna uporaba programa pomeni, da uporabnik uporablja program tako, kot je bilo zamišljeno s strani razvijalcev in v vnosne forme vnaša pravilne vnose ter izvaja akcije v pravilnem zaporedju.

Drugi pogled na zasnovo testov je, kjer uporabnik zaradi kateregakoli razloga program uporablja na nepredviden način. Prav v takih situacijah največkrat pride do napačnega delovanja uporabe, zato je zelo pomembo, da načrtovalec testov zelo dobro pozna aplikacijo in vso njeno funkcionalnost.

Teste smo sprva načrtovali po modulih aplikacije, vendar zaradi prepletanja funkcionalnosti ta ločitev ni zelo stroga. Gre bolj za lažji nadzor in lažje vzdrževanje testov. Za testiranje osnovne in pravilne uporabe aplikacije so bili napisani testi, ki pokrijejo:

- dodajanje, igralcev, ekip in tekmovanj.
- urejanje iger, igralcev, ekip in tekmovanj.
- brisanje iger, igralcev, ekip in tekmovanj.
- povezovanje igralcev v ekipe.
- odjava igralcev iz ekip.
- prijava igralcev in ekip na tekmovanja
- pdjave igralcev in ekip od tekmovanj.
- prikaz vseh udeležencev določenega tekmovanja.
- prikaz vseh tekmovanj določenega igralca ali ekipe.
- prikaz vseh igralcev določene ekipe.

Drug sklop testov pokriva robne primere in izjemne scenarije. Tu se predvsem preverja, če aplikacija pravilno obravnava primere, kjer uporabnik želi storiti nekaj, kar bi lahko povzročilo nepravilno stanje v podatkovni bazi in tako aplikacija ne bi več delovala pravilno. Ti scenariji so:

- napačna izpolnitev vnosnih polj s prekratkimi ali predolgimi nizi znakov,
- napačnimi tipi vnosov (npr. črke namesto številk, ali črke v polje za datum),
- urejanje igralcev, ki so del ekipe,
- poizkus brisanja ekipe ali igralca, ki je prijavljen v tekmovanje,
- poizkus brisanja igre, ki je uporabljena v nekem tekmovanju,
- poizkus dodajanja iger, igralcev, ekip in tekmovanj, ki že obstajajo.

5.3 Selenium projekt

Testni Selenium projekt je strukturiran tako, da je pisanje in vzdrževanje testov čim bolj poenostavljeno. Projekt je razdeljen v šest paketov:

- `selenium.tournament`
- `selenium.pages.dto`
- `selenium.pages`
- `selenium.tournament.stories`
- `selenium.tournament.db`
- `selenium.tournament.db.jdbc`

5.3.1 Paket `selenium.tournament`

Paket vsebuje tri razrede:

- `LocatorType`
- `PageElement`

- `PageElementTemplate`

Te tri razrede smo razvili za pomoč pri iskanju grafičnih elementov po spletni aplikaciji, kot so gumbi in vnosna polja. `LocatorType` je samo ovojnica okoli Seleniumovega razreda `By`, saj ga priredi za lažjo uporabo v razredu `PageElement`. Selenium nam omogoča iskanje grafičnih elementov po spletni strani na podlagi različnih atributov in lastnosti. Elemente zna poiskati preko atributa `id`, atributa `name`, besedilu povezave ali pa s pomočjo jezika Xpath. Razred `LocatorType` vrača enum in vsebuje podporo za za vse tipe iskanja, ki jih vsebuje razred `By`. Element, ki ga želimo najti se vrača kot Seleniumov tip `WebElement`. V nadaljevanju je predstavljen el kode, ki nam omogoča iskanje po atributu `id`:

```
BY_ID {
  @Override
    public WebElement getElement(WebDriver driver, String locator) {
      WebElement we = (new WebDriverWait(driver,  7))
        .until(ExpectedConditions
          .presenceOfElementLocated(By.id(locator)));
          return driver.findElement(By.id(locator));
    }
}
```

`LocatorType` je uporabljen v ostalih dveh razredih tega paketa. Tako `PageElement`, kot `PageElementTemplate` sta namenjena predstavitvi grafičnega elementa spletne aplikacije. `PageElement` predstavi grafični element z dvema atributoma locator tipa `String` in `locatorType`, ki je tipa `LocatorType`. Atribut `locatorType` nam pove na kakšen način želimo poiskati element, atribut `locator` pa je dejanska vrednost, ki jo `locatorType` potrebuje. Na primer, če želimo poiskati vnosno polje za naslov igre, ki jo dodajamo v aplikacijo, bi jo poiskali preko atributa `id`, ki ima vrednost `addGameForm:add_game`.

```
PageElement FIELD_ADD_GAME =
new PageElement("addGameForm:add_game",LocatorType.BY_ID);
```

Razred `PageElementTemplate` je zelo podoben razredu `PageElement`. Omogoča nam parametrizacijo elementov, ki jih iščemo. Najlažje razložimo to s primerom. Če želimo izbrisati igro iz seznama iger, ki so na voljo v aplikaciji, moramo najprej najti vrstico, v kateri se nahaja naslov igre, ki jo iščemo. Ker želimo podatke ločiti od logike, nam to omogoči razred `PageElementTemplate`. S pomočjo jezika XPath poiščemo vrstico, ki vsebuje vrednost našega parametra. Vse kar nam preostane je to, da parameter vstavimo in preko metode `getPageElement` dobimo objekt, ki je tipa `PageElement` s katerim že znamo operirati.

```
PageElementTemplate EDIT_ITEM =
new PageElementTemplate("//tr[td[contains(.,''{0}'')]]/td",
LocatorType.BY_XPATH);

PageElement tempGame = EDIT_ITEM.getPageElement(game.getTitle());
```

5.3.2 Paket `selenium.pages.dto`

V tem paketu so zbrani razredi, ki predstavljajo entitete, ki jih v aplikaciji dodajamo. Vsaka entiteta je predstavljena s svojimi atributi, ki so enaki atributom v aplikaciji skupaj s svojimi metodami za nastavljanje in vračanje vrednosti atributov.

5.3.3 Paket `selenium.pages`

Vsaka stran aplikacije je za testiranje predstavljena kot samostojen razred v paketu `selenium.pages`. Vsak razred, ki predstavlja stran aplikacije vsebuje:

- vse grafične elemente, ki so na strani in s katerimi lahko uporabnik manipulira
- vse akcije, ki se na strani lahko izvajajo

Za boljšo organizacijo datotek in lažje vzdrževanje teh razredov vsi razredi dedujejo razred `AbstractPage`, ki vsebuje bolj splošne elemente in metode, ki so uporabljene povsod:

- pisanje v tekstovno polje,
- klik na element,
- klik na možnost iz spustnega seznama,
- označitev ali odznačitev preklopne izbire,
- desni klik na element,
- postavitev miške čez element,
- preverjanje, če obstaja določen tekst na strani.

Poleg akcij vsebuje tudi nekaj splošnih elementov, ki se pojavljajo na vseh straneh, kot na primer nekatere izbire iz spustnih seznamov ob desnem kliku na element. Primer za to je urejanje entitete (igre, igralca...), kjer je v vseh modulih ta izbira poimenovana enako `Edit`. Ostali razredi tega paketa podedujejo razred `AbstractPage` in ga razširijo z njihovimi grafičnimi elementi in metodami. Te razredi so: `GamePage`, `MenuPage`, `PlayerPage`, `TeamPage`, `TournamentPageSolo`, `TournamentPageTeam`.

Da se lahko v testih premikamo po straneh aplikacije uporabljamo Seleniumov razred `WebDriver`. Ta nam omogoča izvajanje akcij, kot bi jih lahko izvajal človeški uporabnik. Vsebuje tudi informacijo na kateri strani aplikacije se nahajamo. Če se na primer nahajamo na modulu `Igre`, ne moremo dodati igralca, saj nam ta stran tega ne omogoča, lahko pa dodamo igro. Ker smo vsako stran aplikacije postavili v svoj razred, moramo spremenljvko tipa `WebDriver` prenašati med njimi, da lahko izvedemo željene ukaze. Primer prikazuje odpiranje spletnega naslova `http://www.google.si`.

```
WebDriver driver = new FirefoxDriver();
driver.get("http://www.google.si");
```

5.3.4 Paket `selenium.tournament.stories`

V tem paketu so razredi, ki vsebujejo teste aplikacije Tournament Organizer. Tako kot drugje so tudi tu razredi razdeljeni v skladu z moduli aplikacije. Ker ima vsak test tudi začetno in končno fazo, smo razvili razred `AbstractStory`, ki ga ostali razredi tega paketa dedujejo. Glavni metodi, ki ju vsebuje, sta metoda `setUp`, ki se izvede v fazi `@Before` in metoda `tearDown`, ki se izvede po testu v fazi `@After`. Za naše potrebe ne potrebujemo faz `@BeforeClass` in `@AfterClass` zato ju nismo uporabili. V metodi `setUp` inicializiramo naš `WebDriver`, naložimo nastavitve iz konfiguracijske datoteke `settings.properties` ter registriramo gonilnik za podatkovno bazo MySql. Naša datoteka `settings.xml` izgleda takole:

```
webdriver.firefox.bin=/usr/bin/firefox
baseUrl=http\://localhost\:8082/jsf-blank/players.jsf
client.implicitWait.seconds=8
db.conn.url=jdbc\:mysql\://192.168.1.109/tekmovanja
db.conn.user=root
db.conn.pwd=12345
```

V metodi `tearDown` samo zapremo povezavo na naš `WebDriver` in pokličemo metodo, ki vzpostavi začetno stanje podatkovne baze. To je abstraktna metoda z imenom `databaseCleanup`, ki sprejme en parameter. Ta parameter je povezava do podatkovne baze. Abstraktna je zato, ker jo vsak podrazre implementira na svoj način. Metoda je namenjena za čiščenje podatkovne baze po izvedenih testih. Ker skoraj vsak test spreminja vsebino podatkovne baze, je zaradi ponovljivosti testov potrebno vzpostaviti začetno stanje baze. Primer za to je test, kjer dodamo igro v aplikacijo in jo na koncu testa tudi pobrišemo. V primeru, da smo se pri implementaciji metode za brisanje igre zmotili in ta ne deluje, nam ob naslenjem poganjanju tega testa ne bo uspelo igre niti dodati, saj že obstaja v podatkovni bazi. Metoda `databaseCleanup` nam torej skrbi za usklajenost podatkovne baze. Poleg vseh teh metod vsebuje razred `AbstractStory` tudi metodo `start`, ki

jo uporabljajo vsi testi in nam odpre spletno aplikacijo ter vrne stran tipa `MenuPage`, preko katere se lahko premikamo po celotni aplikaciji. Razredi, ki dedujejo razred `AbstractStory` so: `GameStory`, `PlayerStory`, `TeamStory`, `SoloTournamentStory`, `TeamTournamentStory`.

Ti razredi vsebujejo 36 testnih zgodb, ki testirajo pravilno delovanje aplikacije. Vsak test vsebuje nekaj akcij, ki bi jih lahko izvedel uporabnik. Pokrivajo scenarije, ki smo jih določili med načrtovanjem testiranja. Za zagotovitev pravilnega delovanja aplikacije ni dovolj le izvajanje enake akcije, kot bi jo uporabnik, temveč moramo po vsaki opravljeni akciji tudi preveriti stanje aplikacije, če je tako kot mora biti. To storimo s pomočjo razreda `Assert`, ki pripada testnemu ogrodju `junit`. Uporaba razreda `Assert` nam omogoča primerjanje trenutnega stanja s pričakovanim stanjem. Stanji se morata ujemati, sicer test ne bo uspešno zaključen, ampak bo prišlo do napake. Največkrat smo uporabili metodo `assertTrue` v kateri smo klicali metodo `isTextPresent` iz razreda `AbstractPage`. `AssertTrue` pričakuje, da bo izraz v argumentu vrnil vrednost `true`. Metoda `isTextPresent` pa vrne `true` v primeru, ko na strani aplikacije najdemo določen niz in `false`, če ga ne. Testne podatke, ki jih uporabljamo v testih, smo definirali kot konstante na začetku vsakega testnega razreda, saj so tako enostavno dostopni, razumljivi in enostavni za vzdrževanje in spreminanje. Vsi testi imajo enako osnovo, ki se v vsakem testu še malo razširi. Osnovni koraki za testiranje posameznega modula aplikacije so:

- definiraj objekt, ki predstavlja entiteto,
- odpri brskalnik in pojdi na domačo stran aplikacije,
- pojdi na določen modul aplikacije,
- dodaj entiteto,
- preveri, če je bila entiteta dodana,
- uredi entiteto,

- preveri, če je bila entiteta ustrezno spremenjena,
- zbriši entiteto,
- preveri, če je bila entiteta izbrisana.

To so le osnovni koraki, ki opisujejo izolirano testiranje posameznega modula. V primeru testiranj povezav med moduli so prisotni še koraki povezovanj modulov, kot na primer prijava ekipe na tekmovanje, in preverjanje, če je ekipa res prijavljena.

5.3.5 Paketa `selenium.tournament.db` in `selenium.tournament.db.jdbc`

Paket `selenium.tournament.db` vsebuje razred `DbService`, paket `selenium tournament db jdbc` pa razreda `DbJdbc` in `DatabaseHelper`. Razred `DbService` vsebuje statične metode, ki jih kličemo v testnih zgodbah znotraj metode `databaseCleanup`, da vzpostavimo začetno stanje podatkovne baze. Razred `DbService` povezuje testne zgodbe z razredom `DbJdbc`, ki prav tako vsebuje statične metode. Te metode pa predstavljajo najnižji nivo komuniciranja aplikacije z podatkovno bazo.

```
public static void deleteGame(Connection con, String gameTitle){
  DbJdbc.deleteGame(con, gameTitle);
}
```

```
private final static String SQL = "DELETE FROM game WHERE title = ?";
public static int deleteGame(Connection con, String gameTitle){
  return DatabaseHelper.executeUpdate(con, SQL, gameTitle);
}
```

Razred `DatabaseHelper` je pomožni razred, ki nam omogoča poizvedovanje po podatkovni bazi in izvajanje `UPDATE`, `INSERT` in `DELETE` sql povpraševanj.

Poglavje 6

Povezava projektov z orodjem Jenkins

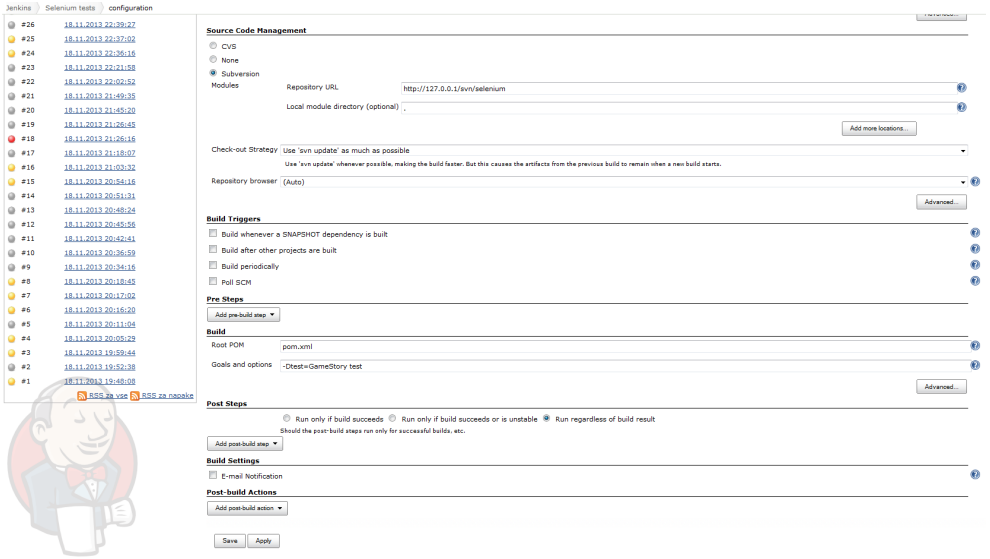
Z orodjem Jenkins smo avtomatizirali izgradnjo paketa naše aplikacije, postavljanje paketa na strežnik Jboss AS7 ter testiranje aplikacije. V orodju smo definirali dva koraka, ki nam to omogočita. Najprej se izvede korak `Tournament Organizer step`. V primeru, da se ta uspešno zaključi se izvede še drugi z imenom `Selenium tests`. Oba koraka smo nastavili preko spletnega vmesnika, ki je prikazan na sliki 6.1.

6.1 `Tournament Organizer step`

V tem koraku najprej preko vgrajene funkcije orodja Jenkins uporabimo zadnjo kodo s strežnika Subversion. Sledi izgradnja paketa in pošiljanje tega paketa na strežnik, da lahko aplikacijo začnemo uporabljati. To nam omogoča orodje Maven, da naredimo v eni vrstici, tako da združimo dva ukaza. Ukaz `package` nam zgradi paket, ki ga preko ukaza `jboss-as:deploy` pošljemo na naš aplikacijski strežnik.

```
mvn package jboss-as:deploy
```

Za pošiljanje paketa na strežnik preko orodja maven smo uporabili vtičnik za orodje maven `jboss-as-maven-plugin`. V konfiguraciji vtičnika določimo



Slika 6.1: Zaslon nastavitev v orodju Jenkins

strežnik za pošiljanje aplikacije ter uporabniško ime in geslo za avtorizacijo. Konfiguracija vtičnika je sestavljena iz več atributov. Atribut `groupId` je edinstveno ime, ki označuje ta vtičnik v mavenovem spletnem repozitoriju. `ArtifactId` je ime knjižnice, brez verzije, saj je ta določena z atributom `version`. Če imamo kdaj težave z verzijami knjižnic le spremenimo ta vnos in ob izgradnji novega paketa bo aplikacija uporabljala novo verzijo vtičnika. `Hostname` označuje spletni naslov našega aplikacijskega strežnika, `port` vrata, katera ima strežnik nastavljena za upravljanje s paketi, `username` in `password` pa sta namenjena avtoziraciji na strežnik.

```
<plugin>
  <groupId>org.jboss.as.plugins</groupId>
  <artifactId>jboss-as-maven-plugin</artifactId>
  <version>7.4.Final</version>
  <configuration>
    <hostname>localhost</hostname>
    <port>9999</port>
```

```
    <username>aaa</username>
    <password>bbb</password>
  </configuration>
</plugin>
```

6.2 Selenium tests

Najprej s strežnika Subversion pridobimo testni projekt, nato pa preko orodja maven poženemo teste. Da je poganjanje čimbolj enostavno smo v datoteki `pom.xml` vtičnik nastavili tako, da teste lahko poženemo z ukazom `test`.

```
mvn test
```

Potrebno je bilo nastaviti le pot do testov ter določiti začetek imen testnih razredov, ki se v našem projektu začnejo z besedo `Story`.

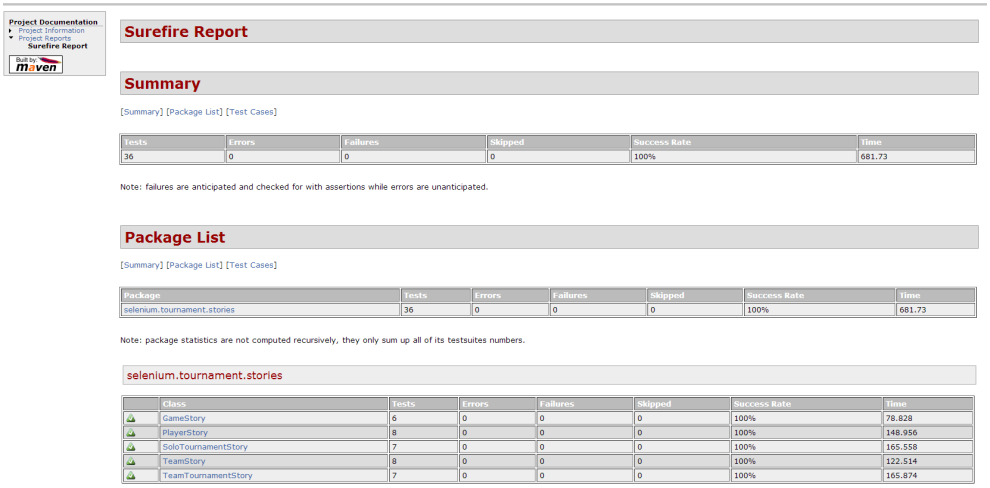
```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>2.14.1</version>
  <configuration>
    <testSourceDirectory>${basedir}/src/main/java/
    </testSourceDirectory>
    <testClassesDirectory>${project.build.directory}/classes/
    </testClassesDirectory>
    <includes>
      <include>**/*Story*.java</include>
    </includes>
  </configuration>
</plugin>
```

6.3 Analiza testiranja

S testnim projektom smo želeli izpolniti cilje testiranja, ki smo si jih zadali. Z implementacijo vseh testnih zgodb smo cilje dosegli v celoti. Pokrili smo vso funkcionalnost aplikacije. Med razvojem testov, smo odkrili nekaj napak v programu Tournament Organizer in jih sproti popravili. Ob vsakem popravku smo zagnali vse teste in tako preverili, da nismo vnesli kakšne regresije. S pomočjo avtomatskih testov smo aplikacijo pripeljali do stanja, kjer se vsi napisani testi izvedejo uspešno.

Za lažjo analizo testiranja smo uvedli tudi generiranje poročila, ki ga prikazuje slika 6.2. Poročilo nam prikaže število izvedenih testov, čas ki smo ga za izvedbo testov porabili, število uspešnih in število neuspešnih testov. Vsak test ima štiri možna stanja v katerih je po izvedbi. Lahko je bil zaključen uspešno, neuspešno, z napako ali pa je bil preskočen. Če je test preskočen, pomeni da je bil označen z `@Ignore`. Uspešno izveden test nam pove, da je delovanje aplikacije, ki jo ta test preverja, pravilno. Če je bil test zaključen neuspešno, nam to označuje napako v programu saj je neka funkcija razreda `Assert` našla drugačno stanje od pričakovanega. Zadnje stanje označeno z napako pa je nepravilno stanje in označuje da se je v programu sprožila izjema, ki je nismo obravnavali. Največkrat to pomeni, da test ni mogel nadaljevati izvajanja, ker ni našel spletnega elementa, ki bi ga potreboval za nadaljnje delovanje (npr. mankja gumb za dodajanje igralca).

Z avtomatskim testiranjem smo zadovoljni in mislimo, da nas je pripeljalo do kvalitetnejše kode, kot če bi aplikacijo testirali ročno. Dolgoročno smo privarčevali tudi nekaj časa, saj se vsi avtomatski testi izvajajo dobrih deset minut, za ročno testiranje pa bi potrebovali okoli pol ure do štirideset minut.



Project Documentation
Project Information
Project Reports
Surefire Report

Surefire Report

Summary

[Summary] [Package List] [Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
36	0	0	0	100%	681.73

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Package List

[Summary] [Package List] [Test Cases]

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
selenium.tournament.stories	36	0	0	0	100%	681.73

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

selenium.tournament.stories

Class	Tests	Errors	Failures	Skipped	Success Rate	Time
GameStory	6	0	0	0	100%	78.828
PlayerStory	8	0	0	0	100%	148.956
SoloTournamentStory	7	0	0	0	100%	165.558
TeamStory	8	0	0	0	100%	122.514
TeamTournamentStory	7	0	0	0	100%	165.874

Slika 6.2: Poročilo testiranja

Poglavje 7

Sklepne ugotovitve

V diplomskem delu smo najprej predstavili tehnologije in orodja, ki smo jih potrebovali za razvoj aplikacije Tournament Organizer in za razvoj testnega projekta. Nato smo aplikacijo načrtovali in razvili. Vzporedno z razvojem aplikacije smo razvijali tudi testni projekt, kjer smo uporabili ogrodje Selenium. Naše cilje naloge smo v celoti izpolnili, saj smo implementirali zastavljeno funkcionalnost aplikacije ter jo ustrezno pokrili z avtomatskimi testi. Največ težav smo pričakovali pri razvoju aplikacije, saj smo prvič uporabljali JavaServer faces 2.0 ter knjižnico Primefaces. Večjih težav nismo imeli in smo hitro osvojili potrebno znanje. Uspešno smo prenesli teste iz načrtovanih testnih scenarijev v testni projekt, da smo zagotovili pravilno delovanje aplikacije. Ker želimo naš čas čimbolje uporabiti, smo uporabili tudi orodje Jenkins, ki nam prevzame ponavljajoče naloge grajenja programskih paketov in postavljanje aplikacije na strežnik, kot tudi poganjanje naših testov. To se je izkazalo kot pravilna odločitev, ki nam je omogočala pot od izvorne kode do testiranja aplikacije z enim klikom. Obstajajo možnosti za izboljšave na vseh področjih našega dela. Aplikacijo bi lahko razširili z novimi funkcionalnostmi, poskrbeli za avtorizacijo uporabnika in povezali z drugimi storitvami. Testiranje bi prav tako lahko razširili tudi na druge nivoje razvoja in napisali teste enot.

Literatura

[1] (2013) "JSF 2 Tutorial Series". Dostopno na:
http://www.coreservlets.com/JSF-Tutorial/jsf2/

[2] (2013) Wikipedia, "MySQL". Dostopno na:
http://en.wikipedia.org/wiki/MySQL

[3] (2013) Wikipedia, "Apache Subversion". Dostopno na:
http://en.wikipedia.org/wiki/Apache_Subversion

[4] (2013) Wikipedia, "Java(programming language)". Dostopno na:
http://en.wikipedia.org/wiki/Java_(programming_language)

[5] (2013) Oracle, "Oracle and Sun". Dostopno na:
http://www.oracle.com/us/sun/index.html

[6] (2013) Wikipedia, "Java Platform, Standard Edition". Dostopno na:
http://en.wikipedia.org/wiki/Java_Platform,_Standard_Edition

[7] (2013) Wikipedia, "Java Platform, Enterprise Edition". Dostopno na:
http://en.wikipedia.org/wiki/Java_Platform,_Enterprise_Edition

[8] (2013) Wikipedia, "Java Platform, Micro Edition". Dostopno na:
http://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition

[9] (2011) Brian Stansberry, "Jboss AS 7 Operating modes". Dostopno na:
https://docs.jboss.org/author/display/AS7/Operating+modes

[10] (2012) Pravind Kumar, "Getting Started Guide". Dostopno na: https://docs.jboss.org/author/display/AS7/Getting+Started+Guide

[11] (2013) Wikipedia, "Apache Maven". Dostopno na: http://en.wikipedia.org/wiki/Apache_Maven

[12] (2013) Apache, "Introduction to the lifecycle". Dostopno na: http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

[13] (2013) Mozzila wiki, "Supported Platforms". Dostopno na: https://wiki.mozilla.org/Mobile/Platforms#Supported_Platforms

[14] (2013) Wikipedia, "JUnit". Dostopno na: http://en.wikipedia.org/wiki/JUnit

[15] (2013) Wikipedia, "XUnit". Dostopno na: http://en.wikipedia.org/wiki/XUnit

[16] (2013) Seleniumhq, "History of Selenium project". Dostopno na: http://docs.seleniumhq.org/docs/01_introducing_selenium.jsp#brief-history-of-the-selenium-project

[17] (2013) Seleniumhq, "Programming Languages". Dostopno na: http://docs.seleniumhq.org/about/platforms.jsp#programming-languages

[18] (2013) Seleniumhq, "What is Selenium". Dostopno na: http://docs.seleniumhq.org/

[19] (2013) Wikipedia, "Software testing". Dostopno na: http://en.wikipedia.org/wiki/Software_testing#The_box_approach

[20] (2013) TortoiseSVN, "About TortoiseSVN". Dostopno na: http://tortoisesvn.net/about.html

[21] (2013) Wikipedia, "TortoiseSVN". Dostopno na:
http://sl.wikipedia.org/wiki/TortoiseSVN

[22] (2013) Wikipedia, "Firefox". Dostopno na:
http://en.wikipedia.org/wiki/Firefox

[23] (2013) Wikipedia, "Firebug (software)". Dostopno na:
http://en.wikipedia.org/wiki/Firebug_(software)

[24] (2013) Jenkins, "Meet Jenkins". Dostopno na:
https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins

[25] (2013) Wikipedia, "Continuous integration". Dostopno na:
http://en.wikipedia.org/wiki/Continuous_integration