UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Nejc Ločniškar

Spletni zajem napovedne moči trgov

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: izr. prof. Marko Robnik Šikonja

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00115/2011
Datum: 05.04.2011

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

Kandidat: **NEJC LOČNIŠKAR**

Naslov: **SPLETNI ZAJEM NAPOVEDNE MOČI TRGOV**
WEB SITE FOR PREDICTION MARKETS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Za pridobivanja informacij o nekaterih vprašanjih se vse bolj uporabljajo napovedni trgi, ki temeljijo na načelu "modrosti množic". Ta princip predvideva, da je pri odgovorih na mnoga vprašanja, povprečenje množice posameznikov natančnejše kot posamezniki in celo boljše od strokovnjakov. Za veljavnost tega načela obstajajo številne potrditve v ekonomiji in psihologiji. Princip dobro deluje v primerih, ko so odgovori posameznikov v množici različni in neodvisni. Želimo raziskati veljavnost tega principa in v ta namen potrebujemo zanimiva in relevantna vprašanja in ustrezno spletno mesto, ki bo omogočalo zajem odgovorov in ustrezno motiviralo udeležence.
Zasnujte in implementirajte spletišče, ki bo simuliralo več tržnih mehanizmov in uporabnikom omogočalo postavljati vprašanja in trgovati z odgovori. Potrebna je ustrezna izvedba s podatkovno bazo in možnostjo analize odgovorov. Problem likvidnosti trga rešujte z metodo LMSR in njenimi izboljšavami.

Mentor:

prof. dr. Marko Robnik Šikonja

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Nejc Ločniškar,

z vpisno številko 63060199,

sem avtor diplomskega dela z naslovom:

Spletni zajem napovedne moči trgov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  izr. prof. Marka Robnika Šikonje,

- 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 04.07.2011 Podpis avtorja: _____

Zahvala

Hvala mentorju za pomoč,

Hvala moji družini za podporo in spodbudo,

Hvala moji Nini za ljubezen,

Hvala podjetju Halcom za potrpežljivost,

Hvala vsem, ki ste mi stali ob strani in verjeli vame.

Kazalo vsebine

Kazalo slik

Seznam uporabljenih kratic in simbolov

CSS – Cascading Style Sheet

DOM - Document Object Model

EJB - Enterprise Java Bean

HSW – Hollywood Stock Exchange

HTML - Hyper Text Markup Language

HTTP - HyperText Transfer Protocol

Java EE – Java Enterprise Edition

JSP – JavaServer Pages

POJO – Plain Old Java Object

SQL - Structured Query Language

POVZETEK

Cilj diplomske naloge je izdelava spletne aplikacije za napovedovanje verjetnosti uresničitve raznih dogodkov s pomočjo napovednih trgov. Pregledali smo področje napovednih trgov, njihov namen in delovanje, ter predstavili koncept modrosti množic ter njegovo umeščenost v našo aplikacijo.

Opišemo programska orodja, uporabljena pri razvoju ter arhitekturno zasnovo rešitve. Osrednji del diplomske naloge predstavlja opis implementacije napovednih trgov kot del spletne aplikacije. Predstavljeni so trije modeli napovednih trgov, to so klasična stavnica, trg ponudbe in povpraševanje ter likviden trg z uporabo algoritma LMSR. Predstavljen je življenjski cikel dogodka, od dodajanja, napovedovanja verjetnosti njegovih izidov in njegovega zaključka.

Napovedni trgi so sposobni natančnih napovedi, ki se lahko kosajo in v določenih primerih presežejo klasične anketne vprašalnike, zato je njihova uporaba v spletni aplikaciji zanimiva za raziskovalno in komercialno uporabo.

Ključne besede: napovedni trgi, modrost množic, razvoj spletnih aplikacij, stavnica, algoritem LMSR, trg ponudbe in povpraševanja

ABSTRACT

The goal of this work is to create a web based application for prediction of future events outcomes using prediction markets. We first analyze prediction markets, their purpose and their functioning. We present the wisdom of the crowds concept and its role in the application itself.

We describe the software tools used in the development process, the architectural design of the solution and the implementation of prediction markets as part of a web application. The focus is on three types of prediction markets: mutual betting market, supply and demand market and liquid market with the use of LMSR. The life cycle of an event is presented, from its creation, prediction of possible outcomes, and its termination.

Prediction markets are capable of producing accurate predictions, which can be as accurate and sometimes more accurate than classical surveys. Their use in a web application therefore encourages academical and commercial use.

Key words: prediction markets, wisdom of the crowds, web application development, mutual betting market, LMSR algorithm, supply and demand market

1. UVOD

Predmet diplomske naloge je spletna aplikacija za ugotavljanje napovedne moči napovednih trgov. Zanimalo nas bo, kakšen je potencial teh trgov za komercialno in raziskovalno uporabo, njihove možne oblike, natančnost napovedi dogodkov in njihova izvedba s programsko kodo.

Velika podjetja in izobraževalne ustanove napovedne trge večinoma s pridom uporabljajo za interne namene, predvsem kot sredstvo za podporo odločanju, mi pa smo želeli zaobjeti kar se da velik krog ljudi. S tem smo prišli do koncepta modrosti množic in sicer do hipoteze, ki pravi, da je v določenih okoliščinah odločitev skupine boljša od odločitve posameznika. Da bi lahko na kar se da enostaven način čim več ljudem dali možnost neodvisnega napovedovanja dogodkov, je najlažji način izvedbe spletna aplikacija.

Po analizi trenutnega stanja smo na naše začudenje prišli do spoznanja, da je takih spletišč izredno malo. Med največje in najstarejše spada Hollywood Stock Exchange [1] (HSW), katerega igralci so pravilno napovedali 32 od 36 zmagovalcev oskarjev leta 2006. Na HSW uporabniki na z virtualnim denarjem kupujejo in prodajajo delnice zvezdnikov in filmov ter s tem določajo njihovo okvirno privlačnost. Na podoben princip deluje tudi Intrade [2], kjer namesto sedme umetnosti uporabniki napovedujejo možnost uresničitve vsakovrstnih svetovnih dogodkov. Intrade se od HSW razlikuje tudi po načinu trgovanja; namesto borznega trga je namreč uporabljen princip ponudbe in povpraševanja (angl. supply and demand), napovedano verjetnost, s katero se bo dogodek zgodil, pa predstavlja razmerje med deležem pozitivnih napovedi in vsoto vseh napovedi. Obe spletišči nagrajujeta pravilno napovedovanje izidov in tako v okviru zabavnega in tekmovalnega okolja izrabljata koncepta modrosti množic in napovednih trgov za natančno napovedovanje dogodkov.

Za dobre napovedi je bistveno sodelovanje čim večjega števila med seboj neodvisnih posameznikov. Ker ljudje splet uporabljajo za razvedrilo, je za doseg velikega števila udeležencev oziroma igralcev pomembno, da je napovedni trg hitro priučljiv, na dolgi rok pa mora še vedno predstavljati izziv. Ne sme manjkati niti nagrajevanje pravilnih napovedi ter lestvica najboljših igralcev za spodbujanje tekmovalnosti.

Bistvo delovanja aplikacije bo možnost spletnega trgovanja z deleži dogodkov iz resničnega življenja in uporaba tega trgovanja za napovedovanje verjetnosti. Aplikacija bo zgrajena s pomočjo odprtokodnih programskih orodij, med katerimi bo temelj predstavljalo razvojno okolje Java EE in njene komponente. Uporabili bomo trinivojsko arhitekturo, pri kateri je

aplikacija razdeljena na podatkovno, poslovno in prikazno plast. Sam proces implementacije bo potekal po principih metodologije agilnega programiranja. Ker je za natančne napovedi na napovednih trgih pomembno veliko število udeležencev, kar je v tako kratkem času težko uresničljivo, bomo delovanje izdelane rešitve preverjali s pomočjo avtomatskih testov.

Naš namen je implementacija spletne aplikacije, ki s pomočjo napovednih trgov in modrosti množic pridobiva kar se da natančne ocene verjetnosti uresničitve raznovrstnih dogodkov. Zanimalo nas bo, kaj so napovedni trgi, kakšne vrste pridejo v poštev v naši aplikaciji, kateri so za uporabnike bolj zanimivi in lažje priučljivi, ter njihovo delovanje in obnašanje v praksi. Bolj podrobno bomo predstavili koncept modrosti množic, njegove prednosti in slabosti, ter njegovo natančnost in zanesljivost. Ustavili se bomo tudi pri tehnološkem vidiku našega dela, pri uporabljenih programskih orodjih in metodah, zanimalo nas bo, kako se koncepti delovanja napovednih trgov prevedejo v dejansko programsko kodo. Na koncu bomo naše delo kritično ovrednotili, poiskali bomo elemente, ki so bili dobri, prav tako pa ne bomo pozabili na možnosti izboljšanja.

2. NAPOVEDNI TRGI

Napovedni trgi (angl. prediction markets) so nastali kot posledica opazovanja dogajanja na povsem običajnih trgih. Cene, ki so se na njih oblikovale, so poskusili uporabiti za napovedovanje stanje trga v prihodnosti. Napovedi dogodkov tako niso osnovni namen trga, vendar nastanejo posredno in so stranski proizvod njegovega delovanja [3].

Prve prave napovedne trge, katerih glavni namen je priti do čim boljših napovedi prihodnjih dogodkov, so oblikovali na univerzi v Iowi, ZDA, in sicer kot laboratorijske poskuse z majhnim število udeležencev [3]. Izkazalo se je, da so bile napovedi, pridobljene na tak način, v večini primerov bolj natančne kot tiste, pridobljene prek anket. Uporabnost napovednih trgov se je razširila preko njihovih začetkov v akademske namene do komercialne in javnomnenjske uporabnosti.

Kot sredstvo za doseganje čim boljših napovedi se je izkazala motivacija igralcev preko konkurence in nagrajevanja uspešnega napovedovanja. Nemotivirani igralci namreč kvarno vplivajo na natančnost napovedi. Cilj je pridobiti dobro informirane posameznike, ki so za svoje natančne napovedi nagrajeni z denarnimi nagradami in povečanjem ugleda, sam napovedni trg pa njihova pričakovanja agregira in na ta način pridobi oceno verjetnosti izpolnitve določenega dogodka. Sodelujočim na tem trgu ni potrebno biti reprezentativen del populacije, pomembno je le, da je na trgu dovolj igralcev in prometa, da so napovedi kar se da natančne [3].

Za napovedni trg in kvaliteto napovedi, ki jih agregira, je pomembna tudi njegova oblika. HSX na primer uporablja borzni model, medtem ko Intrade temelji na ponudbi in povpraševanju. Da bi uporabniki bili kar se da motivirani, je pomembno, da je trg, na katerem trgujejo, lahko priučljiv, na dolgi rok pa še vedno zanimiv. Za potrebe naše spletne aplikacije smo se tako predvsem iz stališča enostavnosti za razumevanje odločili za naslednje tri modele napovednih trgov: klasična stavnica, ponudba in povpraševanje, ter model likvidnega trga.

2.1. Klasična stavnica

Stavnica je od treh izbranih modelov napovednih trgov zagotovo najbolj poznana in razširjena. Pri stavnici ima vsak dogodek dva ali več možnih izidov, verjetnost uresničitve vsakega izmed njih pa je izračunljiva po naslednji enačbi:

$$Verjetnost(izid\ dogodka) = \frac{\sum stave\ na\ izid\ dogodka}{\sum vse\ stave}$$

Enačba 1 : verjetnost izida dogodka pri stavnici

Ko se dogodek izteče, se igralcem, ki so stavili na pravilen izid, izplača glede na višino njihovih stav proporcionalen del celotnega zneska. Delovanje si oglejmo na primeru:

- dogodek ima tri možne izide, izid A, izid B in izid C
- uporabnik 1 na izid A stavi 20 enot, uporabnik 2 pa 10 enot,
- uporabnik 3 na izid B stavi 15 enot,
- uporabnik 3 na izid C stavi 30 enot,
- verjetnost izida A znaša $\frac{30}{75} = 0{,}4$; verjetnost izida B $\frac{15}{75} = 0{,}2$; verjetnost izida C $\frac{30}{75} = 0{,}4$,
- dogodek se zaključi z izidom A, tako da si uporabnik 1 in uporabnik 2 razdelita vsoto vseh stav na dogodku, to je 75 enot (v praksi je potrebno del te vsote nameniti za davke in provizijo stavnice),
- stava uporabnika 1 je znašala $\frac{2}{3}$, kar pomeni, da prejme $\frac{2}{3} * 75 = 50$ enot,
- uporabnik 2 prejme preostanek zneska, kar znaša 25 enot.

Stavnica je vsem razumljiva oblika napovednih trgov, vendar v primerjavi z ostalima modeloma zaostaja predvsem na področju dinamičnosti. Stave so fiksne, ko so enkrat vplačane, jih ni več možno spreminjati.

2.2. Ponudba in povpraševanje

Trg ponudbe in povpraševanja je trg dveh izidov – dogodek se bo zgodil ali dogodek se ne bo zgodil. V primeru, da uporabnik meni, da se bo dogodek zgodil, kupuje delnice dogodka, v nasprotnem primeru pa iste delnice prodaja. Uporabnik lahko delnice prodaja, tudi če jih v tistem trenutku nima v lasti (angl. short selling). V tem primeru si delnice »izposodi« z namenom, da jih bo odkupil kasneje ob zaključku dogodka [2].

Na ta način se s kupovanjem delnic ustvarja povpraševanje (angl. demand), s prodajanjem pa ponudba (angl. supply). Ko se dogodek izteče, vrednost delnice postane 10 enot v primeru, da se je dogodek zgodil, oz. 0 enot v primeru, da se dogodek ni zgodil. Ob zaprtju dogodka se delnice, katerih nakupna in prodajna cena se ujemata, združujejo v pare. Lastnik delnice para, ki je narobe napovedal izid dogodka, izplača dobitek lastniku druge delnice para.
To delovanje si oglejmo na primeru:

- Igralec 1 kupi eno delnico dogodka v vrednosti 6 enot, zanjo plača 6 enot. Igralec 2 zastavi eno delnico dogodka v vrednosti 6 enot, zanjo plača 4 enote. Dogodek se uresniči, delnici obeh igralcev postaneta del para, delnici se ovrednotita na 10 enot, igralec 2 igralcu 1 izplača dobitek v višini 4 enot.

Verjetnost, da se bo dogodek uresničil, ponazorimo z naslednjo enačbo:

$$Verjetnost(uresničitev\ dogodka) = \frac{\sum cena\ delnic\ dogodka\ v\ povpraševanju}{\sum cena\ vseh\ delnic\ dogodka}$$

Enačba 2: verjetnost izida dogodka pri ponudbi in povpraševanju

Pri tem tipu trga lahko uporabniki svoje delnice v kateremkoli trenutku prodajo po njihovi nakupni ceni.

2.3. Likviden napovedni trg

Trg tega tipa je tako kot trg ponudbe in povpraševanje trg dveh izidov, dogodek se bodisi zgodi bodisi ne. Njuna glavna razlika je v načinu določanja cen delnic na trgu. Pri prvem se za ceno, po kateri so pripravljeni kupovati ali prodajati, odločajo igralci sami, medtem ko se pri drugem cena delnice določa s pomočjo posebnega algoritma, imenovanega avtomatizirani vzdrževalec trga (angl. automated market maker) [4].

Avtomatizirani vzdrževalec trga je zadolžen za zagotavljanje likvidnosti trga, saj bi v nasprotnem primeru le-ta bil premalo aktiven in s tem nezmožen natančnih ocen verjetnosti. Med najbolj napredne algoritme te vrste sodi Hanson-ov logarithmic market scoring rule market maker (krajše LMSR) [5], katerega smo uporabili tudi v naši aplikaciji.

LMSR beleži, koliko delnic obeh možnih izidov dogodka so igralci do tega trenutka kupili. Naj q1 predstavlja število kupljenih delnic pozitivnega izida, q2 pa število kupljenih delnic negativnega izida. Funkcija cene C(q1,q2), s katero LMSR beleži celoten znesek, ki so ga igralci porabili, je odvisna samo od q1 in q2 [5]. Funkcija je naslednja:

$$C = b * \ln\left(e^{\frac{q1}{b}} + e^{\frac{q2}{b}}\right)$$

Enačba 3: funkcija cene LMSR

Parameter »b« predstavlja največji znesek, ki ga lahko vzdrževalec trga izgubi. Večji kot je, večje so lahko izgube, hkrati pa je trg bolj likviden in lažje prenaša večje število transakcij brez opaznega nihanja cen.

Ko želi igralec trgovati, izbere število delnic določenega izida, katere je pripravljeni kupiti ali prodati. Ceno nakupa 15 delnic pozitivnega izida LMSR določi s pomočjo funkcije cene, v našem primeru bi cena znašala C(q1+15,q2) – C(q1,q2). Na isti način izračunamo vrednost prodaje 20 delnic negativnega izida: C(q1,q2-20) – C(q1,q2).

Oglejmo si delovanje LMSR na dveh konkretnih primerih. Predpostavimo da b=100 in da do sedaj še nihče ni kupil nobenih delnic (q1=0, q2=0). Igralec bi za nakup 10 delnic pozitivnega plačal znesek, izračunan po naslednji enačbi:

$$C(10,0) - C(0,0) = 100 * \ln\left(e^{\frac{10}{100}} + e^{0}\right) - 100 * \ln(e^{0} + e^{0}) = 5{,}12$$

Predpostavimo, da so kasneje do nekega trenutka igralci kupili 50 delnic pozitivnega izida in 10 delnic negativnega izida. Isti igralec se odloči, da bo prodal svojih 10 delnic pozitivnega izida, cena teh delnic znaša:

$$C(40,10) - C(50,10) = 100 * \ln\left(e^{\frac{40}{100}} + e^{\frac{10}{100}}\right) - 100 * \ln\left(e^{\frac{50}{100}} + e^{\frac{10}{100}}\right) = -5{,}87$$

Enačba 5: izračun prodaje 10 delnic

Znesek je negativen, kar pomeni, da igralec na račun prejme izračunano vrednost in s tem ustvari dobiček v višini 0,75 enote.

Verjetnost uresničitve dogodka na likvidnem trgu predstavlja razlika pri nakupu ene delnice pozitivnega izida dogodka:

$$Verjetnost(uresničitev\ dogodka) = C(q1 + 1, q2) - C(q1, q2)$$

Enačba 6: verjetnost uresničitev dogodka pri likvidnem trgu

3. MODROST MNOŽIC

Modrost množic (angl. wisdom of the crowds) je proces, pri katerem odgovor na vprašanje dobimo s pomočjo skupinskega mnenja med sabo neodvisnih posameznikov. S pomočjo tega procesa, ki ga ameriški novinar James Surowiecki podrobno opiše v svoji knjigi [6], lahko ob pravih pogojih množice in skupine odločajo bolj modro od posameznih strokovnjakov.

Vsaka skupina ni primerna za tak način odločanja, za čim boljše rezultate je potrebno, da je skupina:

- raznolika – različna izobrazba, različni pogledi in nazori in različen dostop do informacij pri posameznikih v skupini,
- neodvisna – na mnenje posameznikov ne smejo vplivati drugi člani skupine,
- decentralizirana – hierarhija posameznikov ne sme vplivati na končno odločitev, mnenje vsakega člana skupine je pomembno.

Pomemben je tudi mehanizem združevanja mnenj v skupinsko odločitev. Pod ustreznimi pogoji je odločitev skupine v večini primerov pravilna. Zanimivo je, da so po avtorjevih besedah najboljše odločitev tiste, ki so plod nestrinjanja in tekmovalnosti v skupini. V določenih primerih je lahko odločitev skupine izredno slaba, v tem primeru gre krivca skoraj vedno iskati v prekomernem posnemanju med člani skupine. Na skupinsko inteligenco kvarno vpliva tudi veliko komunikacija med člani skupine.

Proces modrosti množic gre z roko v roki s cilji naše aplikacije. Namen aplikacije je pridobivanje odločitev oziroma napovedi od množice posameznikov, zato bo za natančne napovedi morala izpoljevati iste zapovedi kot proces modrosti množic. Dostopna bo na spletu, prijave bodo odprte vsem, igralci drug od drugega ne bodo odvisni, z raznolikimi napovednimi trgi in kvantifikacijo nihove uspešnosti pa bo zagotavljala prisotnost raznolikih uporabnikov različnih znanj in ozadij.

4. OPIS REŠITVE

V tem poglavju si bomo podrobneje ogledali implementacijo spletišča za napovedovanje dogodkov s pomočjo napovednih trgov. Zanimala nas bodo konkretna programska orodja, arhitekturna zasnova rešitve, implementacija algoritmov vseh treh tipov napovednih trgov in implementacija spletnega dela rešitve.

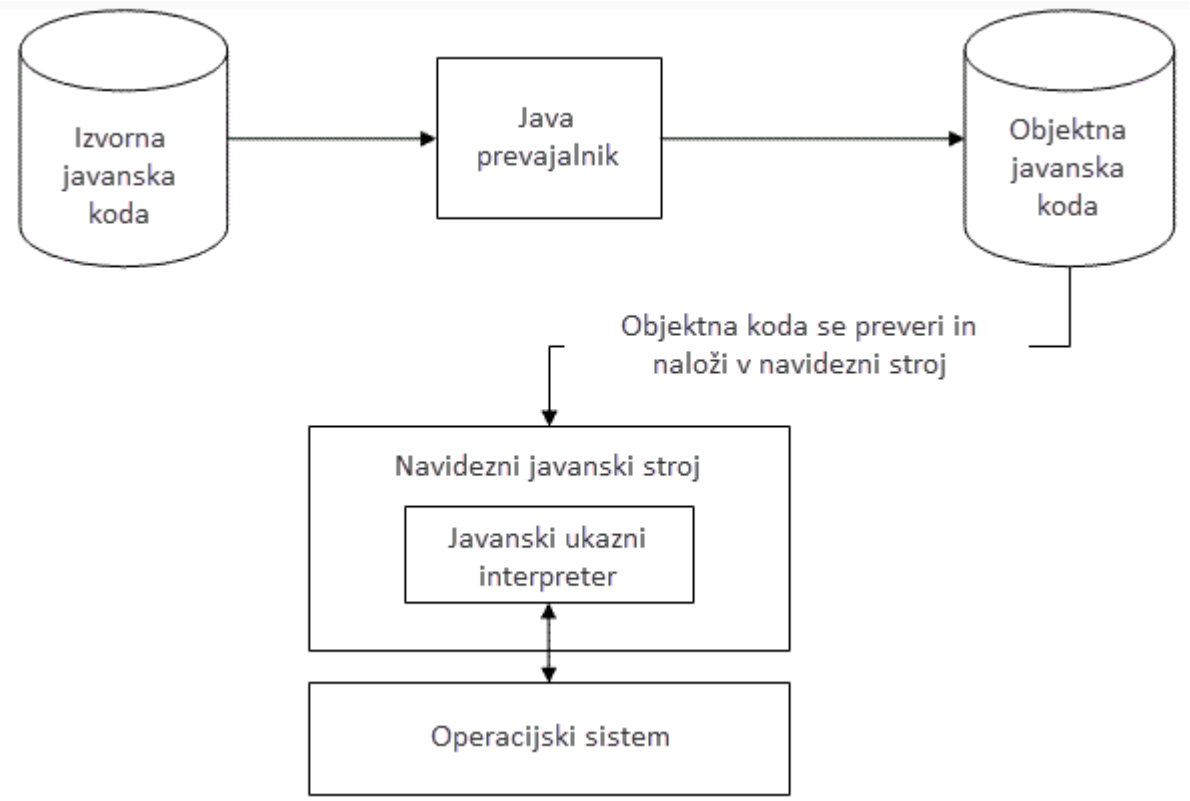
4.1 Programska orodja

Ker se bomo za potrebe naše naloge osredotočili na spletno aplikacijo, se ta usmeritev odraža na izbiri programskih orodij. Želeli smo si, da so orodja odprtokodna, prosto dostopna in da bo končna rešitev platformno neodvisna, tj. neodvisna od operacijskega sistema.

V nadaljevanju si bomo pobliže ogledali vsakega izmed njih in izbiro utemeljili s funkcijskimi zahtevami.

4.1.1 Java EE

Java je trenutno eden izmed najbolj razširjenih objektno usmerjenih programskih jezikov. Njena glavna prednost je neodvisnost od operacijskega sistema, kar pomeni, da javanska aplikacija zgrajena v Windows okolju brez kakršnihkoli prilagajanj deluje tudi v Unix okolju. To omogoča JVM (angl. Java Virtual Machine), oziroma navidezni javanski stroj, čigar naloga je prevajanje vmesne kode v platformi prilagojene ukaze (diagram delovanja na sliki 4.1) in upravljanje s spominom med izvajanjem, kar zmanjša obseg dela programerja.



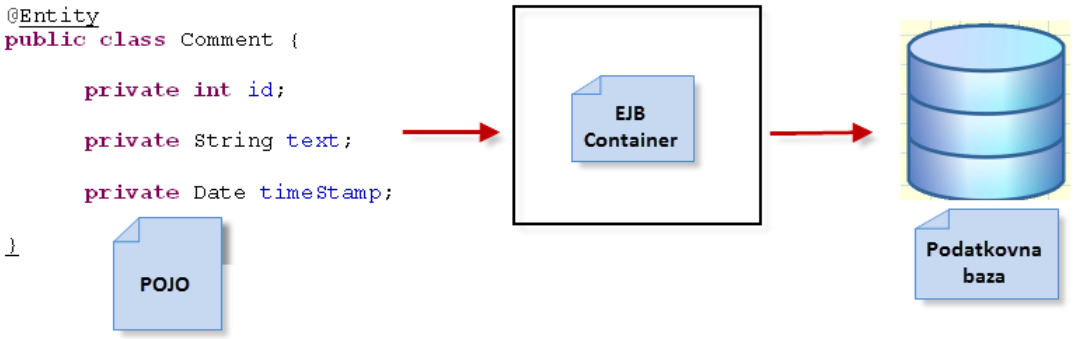
Slika 4.1: Diagram delovanja JVM.

Java EE (angl. Java Enterprise Edition) [7] je javansko razvojno okolje, ki se od običajne oblike Jave (angl. Java Standard Edition) razlikuje v tem, da vsebuje knjižnjice, ki omogočajo razvoj modularnih, porazdeljenih spletnih aplikacij, ki tečejo na aplikacijskem strežniku. Aplikacijski strežnik upravlja s transakcijami, skrbi za varnost, nadgradljivost in upravlja z vsemi komponentami, ki so na njem nameščene, in tako omogoča razvijalcem, da se osredotočijo na poslovno logiko sistema in ne na njegovo infrastrukturo.

Oglejmo si tehnologije Java EE okolja, ki smo jih uporabili za potrebe naše spletne aplikacije.

4.1.1.1 EJB in JPA

EJB (angl. Enterprise Java Bean) je strežniški model za razvoj porazdeljenih spletnih aplikacij. Za našo aplikacijo je bistvena njegova zmožnost dela z entitetnimi zrni. Entitetna zrna (angl. Entity Beans) so povsem običajni javanski razredi (POJO – Plain Old Java Object), opremljeni z ustreznimi anotacijami, ki določajo način preslikave v podatkovno bazo. Sama preslikava je prepuščena vsebniku (angl. EJB container), ki skrbi tudi za veliko ostalih, programerju manj ljubih opravil (shema delovanje arhitekture na sliki 4.2). Vsebnik tako skrbi za podrobnosti podatkovne baze, zmožen je kreiranja podatkovne baze iz entitetnih zrn, vzdržuje podatkovno shemo, upravlja s povezavami na bazo, omogoča sočasno dostopanje do istih virov, skrbi za transakcije, itd.



Slika 4.2: Shema delovanja arhitekture EJB.

Aplikacije, zgrajene z modelom EJB so transakcijske, nadgradljive in varne, prav tako pa niso odvisne od aplikacijskega strežnika, za tega je važno le to, da podpira specifikacijo EJB. V naši aplikaciji je EJB funkcionalnost vzporedna podatkovnem nivoju trinivojske arhitekture.

Za uspešen prenos v podatkovno bazo skrbi knjižnjica JPA (angl. Java Persistence API). Javanski razred označen z JPA anotacijami predstavlja tabelo v podatkovni bazi, instanca istega razreda pa en vnos v tabeli (primer preslikave iz javanskega objekta v zapis v bazi na sliki 4.3). S pomočjo JPA lahko razvijalec brez uporabe jezika SQL vpisuje, briše in spreminja podatke v bazi. Jedro JPA predstavlja objekt EntityManager, ki sinhronizira stanje v podatkovni bazi, skrbi za istočasno izvajanje transakcij in za odpiranje in zapiranje povezav na bazo.

```
Person person = new Person();
person.setUsername(username);
person.setScreenName(screenName);
person.setPassword(password);
person.setType(personType.text);
person.setActive(true);

em.getTransaction().begin();
em.persist(person);
em.getTransaction().commit();
```

Slika 4.3: Primer vnosa v podatkovno bazo z JPA.

Obstaja večje število odprtokodnih ogrodij za delo z JPA, med najbolj razširjenimi najdemo Hibernate, Spring in OpenJPA. Za potrebe naše aplikacije smo uporabili slednjega, predvsem zaradi enostavnosti, saj je ogrodje že priloženo aplikacijskemu strežniku Apache Geronimo, ki je opisan v nadaljevanju.

4.1.1.3 Servlet

Servlet je javanski razred, ki je sposoben sprejemati HTTP zahteve in se na njih po potrebi odzivati. V okviru Java EE so največkrat uporabljeni za generiranje dinamičnih spletnih strani kot odgovor na zahtevo. Njihova funkcionalnost je v mnogočem identična stranem JSP, ki se na aplikacijskem strežniku obdelajo enako kot servleti. Glavna razlika in prednost servletov je v tem, da kot javanski razredi poenostavljajo objektno programiranje.

Kot dobra praksa se smatra uporabo servletov za izvajanje poslovne logike in delegiranje obdelanih podatkov stranem JSP, ki poskrbijo za prikazno logiko. Na ta način se loči poslovno plast od predstavitvene plasti v tristopenjski arhitekturi. Ta princip smo uporabili tudi v naši aplikaciji.

4.1.1.4 JSP

JSP (angl. JavaServer Pages) je javanska tehnologija za izdelavo spletnih strani z dinamično vsebino. To so običajne spletne strani HTML, XML ali XHTML, ki so razširljive z javansko kodo. Tako napisano stran aplikacijski strežnik ob vsakem zahtevku obdela in dinamično zgenerira stran HTML.

V naši aplikaciji smo strani JSP uporabili predvsem za prikazovanje vsebine, opravljajo torej naloge predstavitvene plasti.

4.1.1.4.1 jQuery

jQuery [8] je odprtokodna JavaScript knjižnjica, namenjena poenostavljanju procesa pisanja odjemalskih skript HTML. Spletnemu razvijalcu omogoča:

- olajšano izbiranje in manipuliranje z objekti DOM,
- olajšano delo z dogodki (ob kliku z miško, ob pritisku tipke na tipkovnici, itd.),
- enostavno spreminjanje stilskih predlog CSS,
- dodajanje animacij in efektov,
- olajšano upravljanje z Ajax metodami,
- nadgradljivost z vtičniki,
- preverjeno delovanje v večini spletnih brskalnikov.

4.1.1.2 Aplikacijski strežnik

Aplikacijski strežnik (angl. Application Server) predstavlja okolje, na katerem tečejo Java EE komponente. V našem primeru to pomeni, da na njem teče storitev EJB, servleti, zadolžen pa je tudi za dinamično generiranje strani JSP. Strežnik skrbi za učinkovito izvajanje procedur, med njegove naloge štejemo tudi učinkovito dodajanje, brisanje in spreminjanje komponent, ki se na njem izvajajo.

V Java EE okolju aplikacijski strežnik za komponente, ki na njem tečejo, predstavlja podaljšan JVM, ki transparentno skrbi za povezave do baze na eni strani in povezave do spletnega klienta na drugi. Moderni javanski aplikacijski strežniki imajo v večini primerov že priložene tudi spletne strežnike, ki skrbijo za povezave HTTP, SSL seje in zahtevke HTTP.

Odločili smo se za aplikacijski strežnik Apache Geronimo, ki v polnosti podpira Java EE verzije 6. Vanj je vključeno veliko število javanskih komponent, za nas pa so zanimive predvsem naslednje:

- Apache Tomcat (strežnik HTTP, vsebnik za izvajanje servletov, dinamično generiranje strani JSP),

- Apache OpenEJB (vsebnik za EJB),
- Apache OpenJPA (ogrodje JPA).

Strežnik tako za našo aplikacijo predstavlja enovito rešitev za izvajanje Java EE aplikacij, brez nameščanja dodatnih komponent.

Za uporabo Java EE smo se odločili zato, ker je prosto dostopna, vsebuje komponente, ki znatno olajšajo izdelavo enostavno nadgradljivih spletnih aplikacij, poenostavijo delo s podatkovno bazo, poleg tega pa v njej izdelane aplikacije niso odvisne od operacijskega sistema. Z uporabo strani JSP, servletov in EJB enostavno dosežemo trinivojsko arhitekturo. Zanjo obstaja močno razvojno okolje (Eclipse IDE), zaradi njene popularnosti pa je moč hitro priti do pomoči ostalih razvijalcev in raznovrstnih učnih pripomočkov.

4.1.2 Eclipse

Eclipse je poleg Netbeans najbolj razširjeno prostodostopno razvojno okolje za razvoj običajnih javanskih in Java EE aplikacij. Omogoča enostavno nameščanje vsakovrstnih razširitev, vdelan ima močan razhroščevalnik (angl. debugger), s katerim lahko enostavno sledimo toku izvajanja (tudi spletnih aplikacij), v njem pa je moč uporabiti tudi aplikacijski strežnik, s katerim lahko lokalno testiramo delovanje spletne aplikacije brez nameščanja.

Za Eclipse smo se odločili zato, ker je brezplačen, podpira širok nabor razširitev, ki olajšajo programiranje, vanj je enostavno vključiti katerikoli aplikacijski strežnik, med drugim tudi Apache Geronimo. Z razvojnim okoljem Eclipse je lokalno razvojno okolje postavljeno v zelo kratkem času.

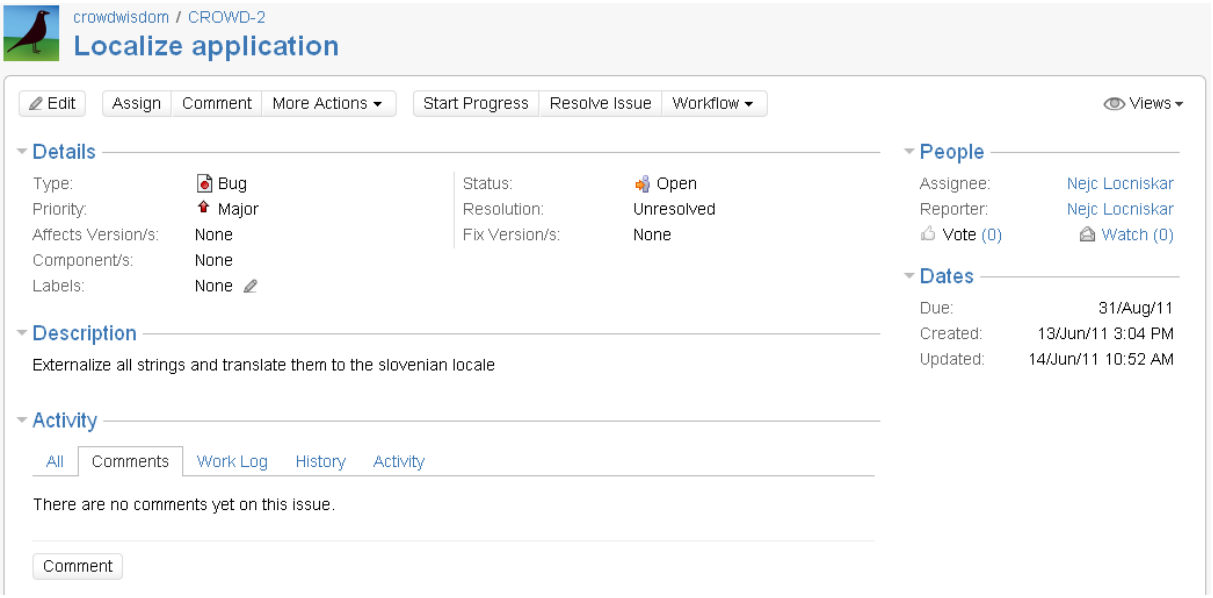
4.1.3 PostgreSQL

PostgreSQL je odprtokoden sistem za upravljanje objektno-relacijskih podatkovnih baz (angl. Object-relational Database Management System), ki ga v primerjavi z ostalimi prostodostopnimi rešitvami odlikuje robustnost, velika stopnja integritete podatkov, dobra podpora transakcijskemu načinu dela, v zadnjem času pa tudi precejšnja hitrost.

Čeprav bi za naše potrebe zadoščal tudi kakšen manj napreden sistem, smo se PostgreSQL, zaradi zgoraj naštetih razlogov vseeno odločili.

4.1.4 JIRA

JIRA je orodje za učinkovito sledenje problemom in napakam na poslovnih projektih. Podpira vse aspekte agilnih metodologij programiranja in je z beleženjem porabljenega časa na posameznem problemu in možnostjo dodeljevanja problemov posameznim programerjem tudi učinkovito orodje za projektno vodenje. Poleg tega je orodje JIRA prek vtičnika možno integrirati v razvojno okolje Eclipse, kar programerju omogoča enostavno sledenje in popisovanju njemu dodeljenih problemov in napak.



Slika 4.4: Primer problema v JIRI.

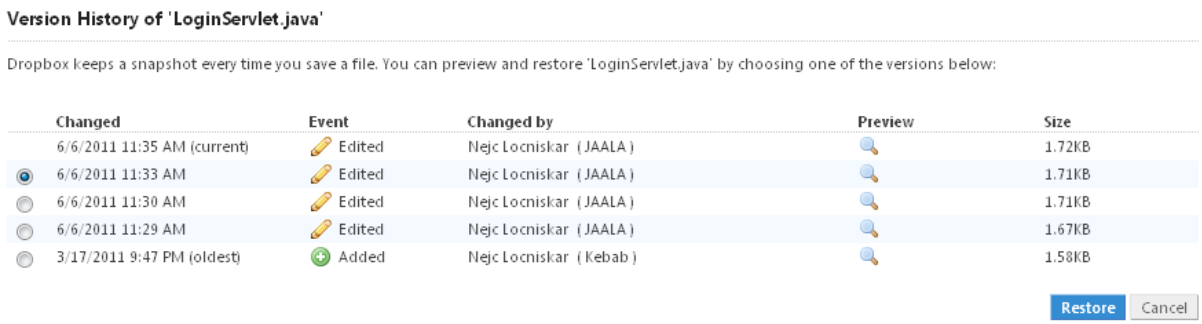
Za potrebe naše aplikacije ima JIRA občutno preobširen nabor funkcionalnosti, zato smo orodje v okrnjeni obliki uporabljali predvsem za sledenje in prijavljanje delov funkcionalnosti, ki jih je bilo potrebno implementirati, poleg tega pa smo s sledenjem porabljenega časa na problemih dobili jasno sliko o tem, katere naloge so bile najbolj časovno potratne.

Za JIRO smo se odločili zato, ker smo želeli implementacijo izvajati strukturirano in usmerjeno, prav tako pa smo si želeli imeti jasno sliko o stanju projekta. Tak način ciljno usmerjenega dela je s stališča motivacije veliko boljši od dela, pri katerem edini cilj predstavlja zaključek implementacije. JIRA je brezplačna za vse odprtokodne projekte.

4.1.5 Dropbox

Večina modernih razvojnih projektov uporablja orodja za kontrolo verzij datotek izvorne kode (angl. revision control system), ker pa je bil naš projekt manjšega obsega in ker je na njem delal samo en razvijalec, smo si želeli enostavenega načina nadzora nad datotekami.

Dropbox je storitev v oblaku, ki uporabnikom omogoča hrambo in delitev datotek in map preko spleta s pomočjo sinhronizacije. Storitev je brezplačna, omogoča pa tudi hrambo in pregledovanje večih različic iste datoteke (primer sledenja na sliki 4.5), kar je osnovna zahteva za kontrolo verzij.

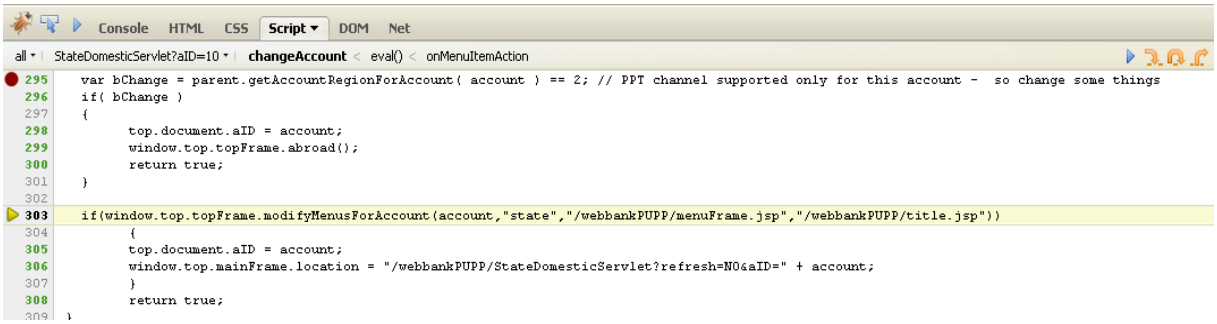


Slika 4.5: Primer sledenja verzijam datoteke.

Za Dropbox smo se odločili predvsem zaradi enostavnosti uporabe, vse kar je bilo potrebno za kontrolo verzij datotek izvorne kode je bila postavitev delovnega okolja v mapo, ki je pod Dropbox nadzorom. Ker se lahko na isti Dropbox račun prijavljamo iz kateregakoli računalnika in celo mobilnika, je na ta način mogoče delo z istim delovnim okoljem s poljubne lokacije.

4.1.6 Firebug

Firebug je vtičnik za Mozilla Firefox brskalnik, s katerim lahko pregledujemo posamezne gradnike spletnih strani, omogoča pregled DOM strukture, ima Javascript konzolo, s katero lahko prožimo Javascript procedure, omogoča pa tudi sledenje izvajanju Javascript kode korak za korakom, kar je izredno uporabno za preverjanje pravilnosti kode.

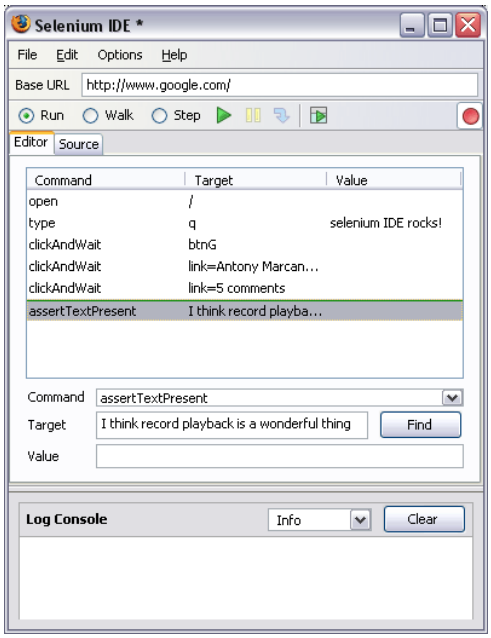


Slika 4.6: Sledenje Javascript ukazom v orodju Firebug.

Firebug je močan, preprost in prosto dostopen dodatek, zaradi katerega se večino analiz in testiranja spletnih strani izvaja ravno v brskalniku Firefox, saj Firebug za Internet Explorer ni dostopen v polni obliki. Za Firebug smo se odločili, ker na svojem področju nima prave konkurence.

4.1.7 Selenium

Selenium IDE je tako kot Firebug vtičnik za brskalnik Firefox, ki omogoča snemanje premikov in klikov miške, posnete akcije pa je mogoče kasneje ponoviti. S tem nam Selenium omogoča izvedbo testov na spletni strani, kar je že od nekdaj rak rana razvoja spletnih aplikacij. Tako posnete teste je možno prevesti v javansko kodo in jih kot običajne teste poganjati v okviru testnega ogrodja, kot je na primer JUnit.

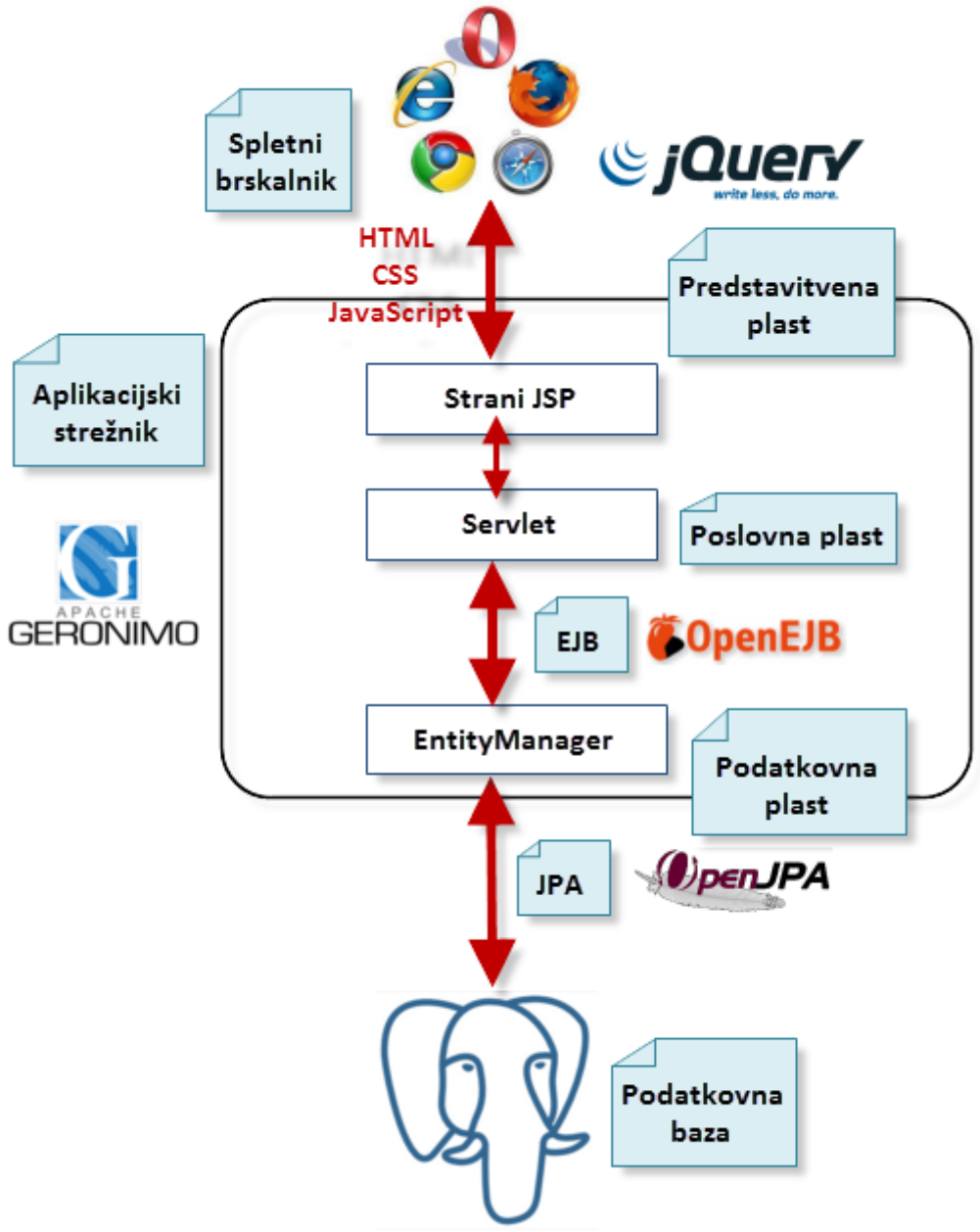


Slika 4.7: Primer testa spletne strani v Selenium IDE.

V naši aplikaciji smo Selenium teste izvajali predvsem po vsakem dodajanju nove funkcionalnosti na obstoječe spletne strani; pomembno je bilo, da se kljub dodajanju novih funkcionalnosti stare še vedno ohranijo. S takimi testi se zagotovi konsistentnost dela in zmanjša obseg potrebnih popravkov v prihodnosti. Selenium smo uporabili tudi za simuliranje aktivnosti uporabnikov.

Za Selenium IDE smo se odločili, ker za testiranje funkcionalnosti samih spletnih strani ne obstaja alternativne rešitve in ker je integracija Selenium testov v javansko kodo enostavna.

4.2 Arhitekturna zasnova



Slika 4.8: Arhitekturna zasnova aplikacije.

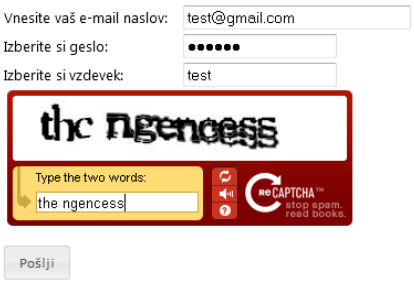
Spletna aplikacija uporablja tristopenjsko arhitekturo in sicer:

- **podatkovna plast:** nalogo podatkovne plasti opravlja knjižnjica JPA, ki skrbi za preslikavo entitetnih zrn v podatkovno bazo,
- **poslovna plast:** logiko poslovne plasti predstavlja storitev EJB v navezi z servleti. EJB s pomočjo entitetnega upravljalca servletom dostavlja entitetna zrna, servleti pa izvajajo obdelavo in izpis posredujejo stranem JSP,

- **predstavitvena plast:** predstavitveno plast predstavljajo strani JSP, ki skrbijo za prikaz podatkov, ki jim jih posredujejo servleti.

Delovanje arhitekture si oglejmo na preprostem primeru registracije novega uporabnika.

1. uporabnik, ki se želi registrirati, izpolne vnosno formo in jo pošlje na registracijski servlet **(JSP -> servlet)**



Slika 4.9: Prvi korak registracije uporabnika.

2. servlet preveri vnešene podatke in s pomočjo entitetnega upravljalca preveri, če je uporabniško ime že zasedeno **(servlet -> EJB)**

```
Person p = broker.getPerson(email);
if(p != null)
    //a user with that email already exists
    error = Labels.getLabel("REGISTRATION_EMAIL_TAKEN", req);
```

Slika 4.10: Drugi korak registracije uporabnika.

3. v primeru, da je izbrano uporabniško ime na voljo, servlet entitetnemu upravljalcu ukaže, naj v bazo doda novega uporabnika z vnešenimi podatki **(EJB -> JPA)**

```
Person person = new Person();
person.setUsername(username);
person.setScreenName(screenName);
person.setPassword(password);
person.setType(personType.text);
person.setActive(true);

em.getTransaction().begin();
em.persist(person);
em.getTransaction().commit();
```

4	4	TRUE	MTIzNDU2	test	user	test@gmail.com	104

Slika 4.11: Tretji korak registracije uporabnika.

4. po končani transakciji entitetni upravljalec servlet obvesti o uspešnosti, servlet pa odgovor posreduje strani JSP **(JPA -> EJB -> servlet)**
5. aplikacijski strežnik JSP dinamično generira in uporabniku prikaže spletno stran z obvestilom o uspešni registraciji. **(servlet -> JSP)**

Na podoben način se izvaja celotna funkcionalnost aplikacije.

4.3 Spletišče

Preden si podrobno ogledamo implementacijo napovednih trgov, je za lažje razumevanje potrebno razložiti nekaj osnovnih pojmov.

- **dogodek** je temeljni pojem celotnega sistema, napovedovanje njegovega izida pa glavni cilj aplikacije. Vsak dogodek je umeščen na enega izmed treh napovednih trgov, to pomeni, da poznamo tri tipe dogodkov: dogodek na klasični stavnici, dogodek na trgu ponudbe in povpraševanja ter dogodek na likvidnem trgu. Dogodki na klasični stavnici imajo lahko med dva in desetih možnih izidov, medtem ko imajo dogodki na ostalih dveh trgih le dva: dogodek se bo zgodil ali dogodek se ne bo zgodil,
- **izid** predstavlja enega izmed možnih zaključkov dogodka, le-ta se zaključi takrat, ko se uresniči eden izmed njegovih možnih izidov. Uporabniki z opredeljevanjem za določen izid dogodka ustvarjajo delnice tega dogodka,
- **delnica** predstavlja napoved izida dogodka, višja kot je njena cena in večje kot je število delnic istega izida, večja je napovedana verjetnost njegove uresničitve. Cena delnice na klasični stavnici in na trgu ponudbe in povpraševanje je določena s strani uporabnikov, medtem ko ceno na likvidnem trgu določa avtomatizirani vzdrževalec likvidnosti trga. Delnica je v obtoku, dokler se dogodek, na katerega se navezuje, še ni zaključil. Takoj, ko se dogodek zaključi, se iz obtoka umakne vse delnice njegovih izidov. Delnice, ki so v uporabnikovi lasti, se nahajajo v njegovi denarnici,
- **denarnica** hrani vse uporabnikove delnice, ki so še v obtoku (investicije) in sredstva, s katerimi lahko uporabnik razpolaga na trgih.

V nadaljevanju je opisan življenjski cikel dogodka v aplikaciji. Uporabnik, pooblaščen za dodajanje dogodkov, doda dogodek z določenim datumom preteka. V časovnem obodbju, ko je dogodek aktiven, lahko vsi registrirani uporabniki z vplačevanjem stav in kupovanjem delnic dogodka izvajajo napovedi. Ko dogodku preteče doba veljavnosti, uporabnik, ki je dogodek dodal, le-tega zaključi z izbiro izida, ki se je zgodil. Aplikacija uporabnikom, ki so pravilno napovedali izid dogodka, izplača dobitke in prilagodi lestvico najbolj uspešnih napovedovalcev.

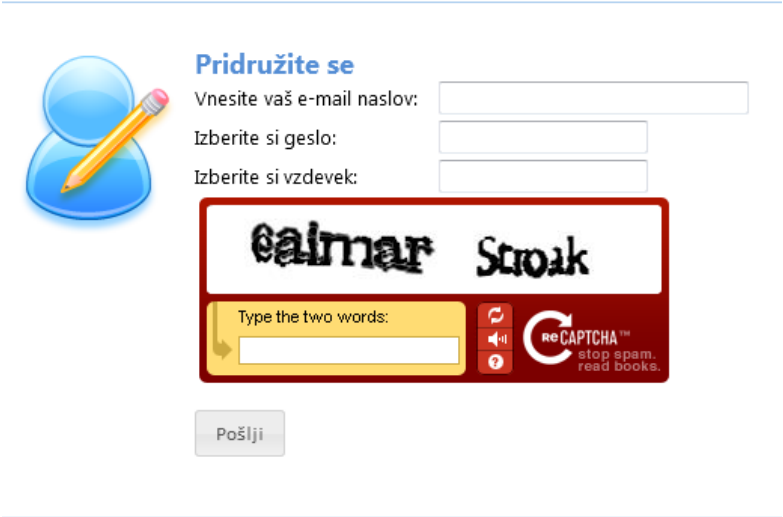
Spletna aplikacija omogoča ustvarjanje dogodkov, vplačevanje stav oziroma kupovanja in prodajo delnic, spremljanje napovedi dogodkov, ogled statistike zaključenih dogodkov, ogled lestvice najbolj priljubljenih dogodkov in najuspešnejših napovedovalcev ter pregledovanje uporabniškega profila. Aplikacija je logično razdeljena na 6 modulov:

- jedro sistema,

- prva stran,
- ustvarjanje dogodkov,
- trgovanje,
- uporabniški profil,
- statistike.

4.3.1 Jedro sistema

Modul jedro sistema opravlja temeljne storitve aplikacije. Med temi so dostop do baze, periodično preverjanje pretečenosti dogodkov, zbiranje podatkov za statistično analizo, del modula pa sta tudi prijava v aplikacijo in registracija novih uporabnikov (posnetek zaslona na sliki 4.12).

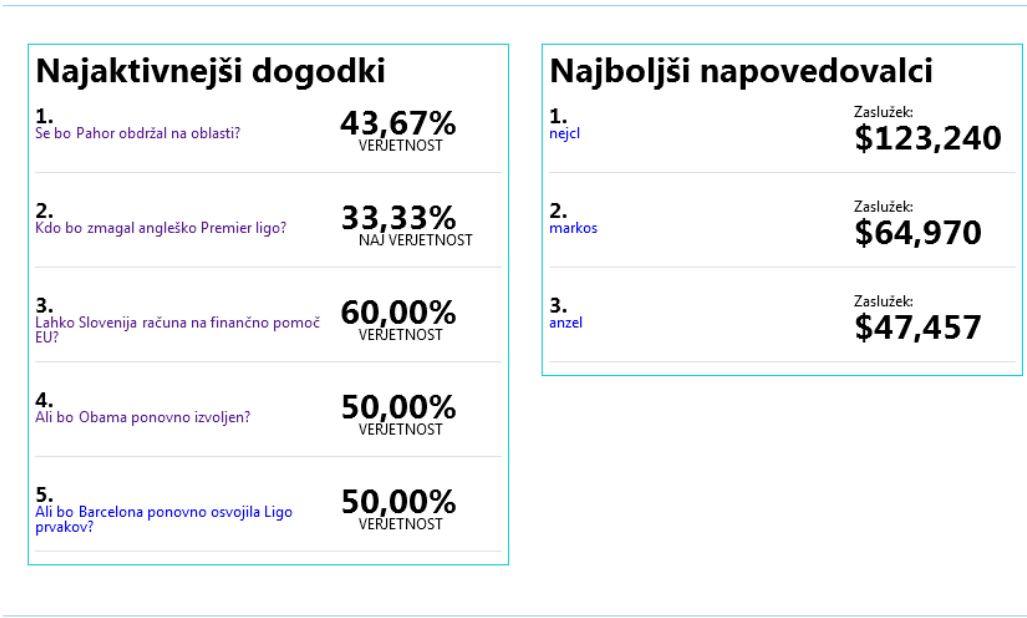


Slika 4.12: Posnetek zaslona za registracijo.

Pri registraciji je uporabljena tudi storitev reCAPTCHA, ki zahteva prepoznavanje besed na slikah, s čimer preverja, da je uporabnik, ki se želi registrirati, res človek in ne avtomatiziran program. To storitev bomo podrobneje predstavili v naslednjem poglavju.

Modul vsako minuto periodično preverja dogodke in v primeru, da je datum preteka starejši od trenutnega datuma, dogodek označi za pretečenega in s tem onemogoči trgovanje z delnicami njegovih izidov.

4.3.2 Prva stran

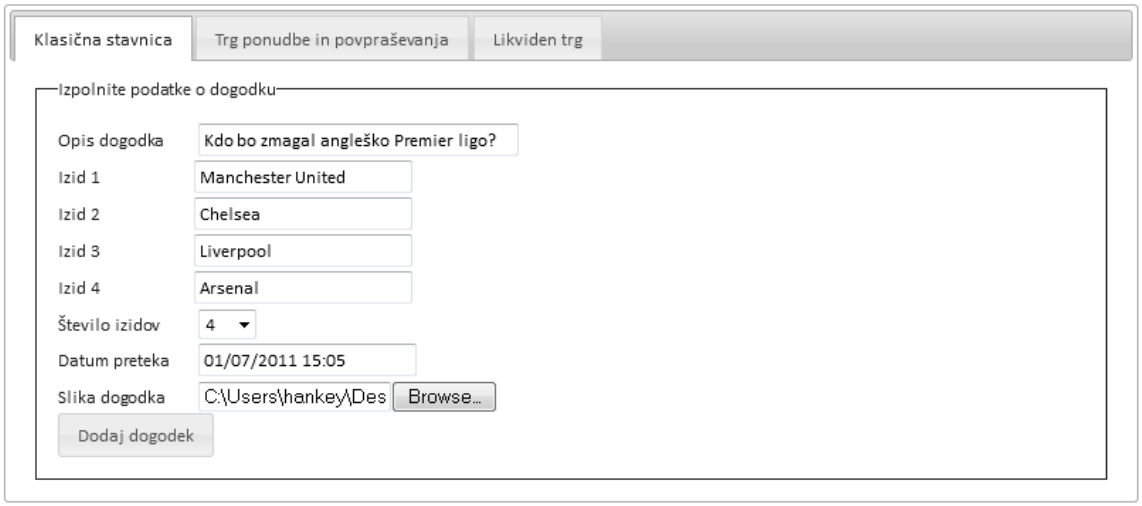


Slika 4.13: Posnetek prve strani.

Na prvi strani sta prikazani lestvici najaktivnejših dogodkov in najuspešnih napovedovalcev (posnetek strani na sliki 4.13). Najaktivnejši dogodki so tisti, pri katerih so za napovedovanje izidov uporabniki porabili največ denarja, najboljši igralci pa tisti, ki so z napovedmi zaslužili največ denarja.

4.3.3 Dodajanje dogodkov

Modul omogoča dodajanje novih dogodkov, pri čemer je treba izbrati tip in opis dogodka, v primeru, da gre za klasično stavnico, pa je potrebno izbrati še število možnih izidov in njihov opis. Določiti je potrebno še datum preteka, ki določa obdobje veljavnosti dogodka, opcijsko pa je možno dogodku priložiti tudi sliko (posnetek zaslona na sliki 4.14).



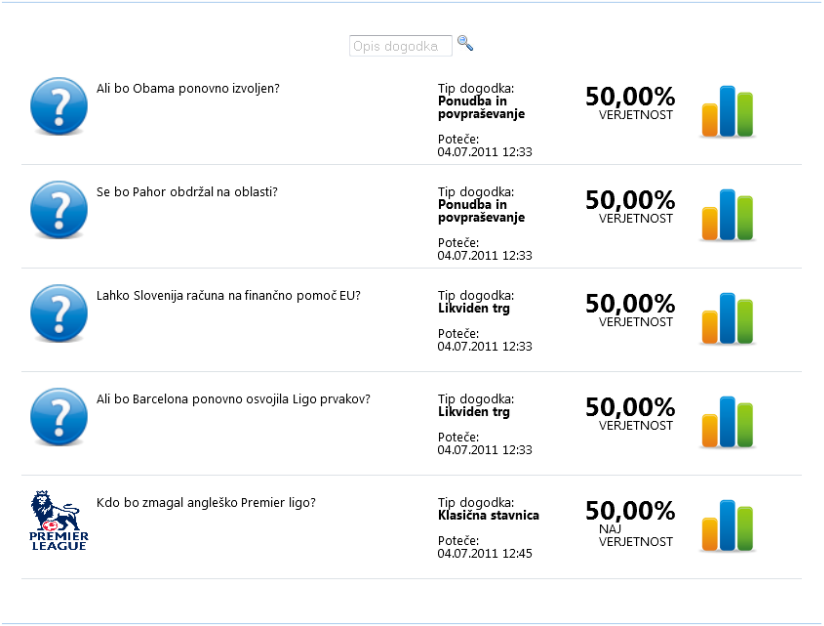
Slika 4.14: Posnetek zaslona za ustvarjanje dogodkov.

Tako dodan dogodek je nato viden na listi aktivnih dogodkov vse do preteka njegove veljavnosti.

Zaradi nevarnosti zlorab in poplave dogodkov brez vrednost (angl. spam) dodajanje dogodkov ni dovoljeno vsem uporabnikom, ampak samo tistim, ki jih odobri eden od administratorjev.

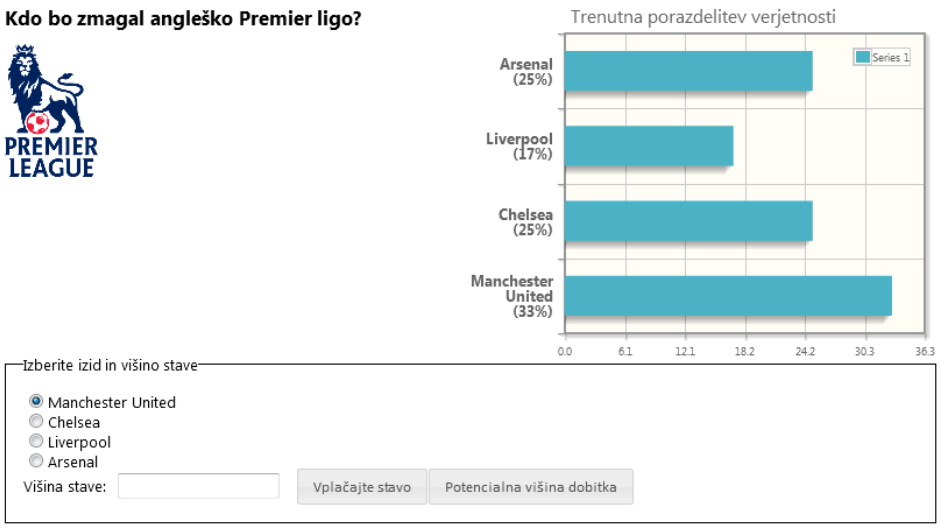
4.3.4 Trgovanje

Modul omogoča iskanje po seznamu aktivnih dogodkov, vplačevanje stav in kupovanje ter odprodajo delnic dogodka, s čimer uporabniki napovedujejo izid.



Slika 4.15: Posnetek zaslona s seznamom aktivnih dogodkov.

Uporabnik, ki si želi sodelovati pri napovedovanju, si iz liste aktivnih dogodkov (posnetek zaslona na sliki 4.15) izbere enega, in se odloči za njegov izid. V primeru klasične stavnice je vidna trenutna porazdelitev verjetnosti po posameznih izidih (posnetek zaslona na sliki 4.16), pri dogodku na trgu ponudbe in povpraševanja pa so vidne cene, po katerih so ljudje pripravljeni delnice dogodka kupovati in prodajati. Pri vplačevanju stav lahko uporabnik dobi tudi informacijo o potencialni višini dobitka glede na višino vplačane stave.



Slika 4.16: Posnetek zaslona vplačevanja stav.

Pri kupovanju delnic na trgu ponudbe in povpraševanja, aplikacija uporabniku samodejno predlaga trenutno najbolj ugodno ceno, po kateri ljudi prodajajo ali kupujejo. Pod zavihkom »Nepokrite delnice« lahko uporabniki dobijo informacijo o delnicah, ki še nimajo svojega para in so v obtoku.

Slika 4.17: Posnetek zaslona kupovanja delnic na trgu ponudbe in povpraševanja.

Pri trgovanju na likvidnem trgu je zaradi hitrega spreminjanja cen delnic pomembno, da se uporabnika pred nakupom obvesti o celotnem znesku transakcije (posnetek zaslona na sliki 4.18).

Slika 4.18: Posnetek zaslona kupovanja delnic na likvidnem trgu.

4.3.5 Uporabniški profil

Modul uporabnikom omogoča pregled njihovih financ in investicij ter vseh dogodkov, ki so jih ustvarili. Uporabniki lahko v razdelku z investicijami prodajajo delnice ponudbe in povpraševanja in delnice likvidnega trga, prav tako pa lahko dobijo informacijo o potencialni višini dobitka za vsako investicijo. V razdelku dogodkov lahko uporabniki pregledujejo trenutno stanje njihovih dogodkov in pretečene dogodke tudi zaključujejo (posnetek zaslona na sliki 4.19).

Slika 4.19: Posnetek zaslona uporabniškega profila, razdelek dogodki.

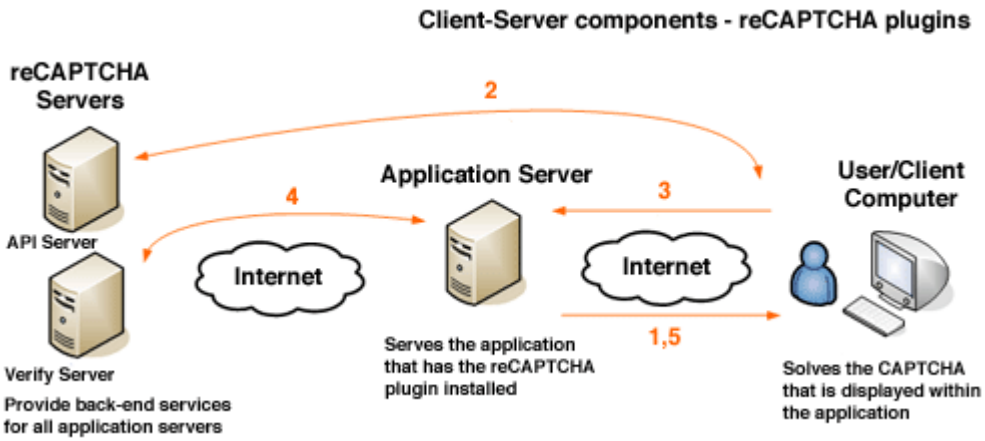
4.3.6 Statistike

Modul omogoča pregledovanje natančnosti napovedi uporabnikov za izide že pretečenih dogodkov (posnetek zaslona na sliki 4.20).

Slika 4.20: Posnetek zaslona s statistiko

Prikazan je izid, ki se je uresničil, verjetnost, s katero so uporabniki napovedali uresničen dogodek in vsote investicij.
Uporabniki lahko statistiko razvrščajo po vseh stolpcih, možno pa je tudi iskanje po opisu dogodka.

4.4 Uporaba reCAPTCHA prepoznavanja



Slika 4.21: Diagram delovanja storitve reCAPTCHA.

Namen storitve reCAPTCHA je ločevanje dejanskih uporabnikov od avtomatiziranih računalniških programov, s čimer preprečuje množične vnose podatkov brez vrednosti. Od uporabnika se zahteva prepoznavanje popačenih besed na sliki, česar računalniki niso zmožni.

Življenski cikel reCAPTCHA prepoznavanja (diagram delovanja na sliki 4.21):

1. uporabik naloži stran JSP z registracijo, ki vsebuje JavaScript skripto za prikaz reCAPTCHA,
2. brskalnik s skripto zahteva sliko z besedama, reCAPTCHA strežnik odgovori s sliko in ključem, ki identificira sliko,
3. uporabnik prepiše besedi s slike in ju pošlje na servlet,
4. servlet pošlje besedi in ključ na strežnik, ki vrne rezultat,
5. v primeru, da je uporabnik pravilno prepisal besedi s slike, ga sistem registrira.

Na strani JSP potrebno inicializacijo opravi naslednja koda:

```
ReCaptcha c = ReCaptchaFactory.newReCaptcha(privateKey, publicKey, false);
out.print(c.createRecaptchaHtml(null, null));
```

Privatni in javni ključ sta vezana na domeno, na kateri je aplikacija nameščena, za vsako domeno je potreben nov par ključev.

Na strežniški (servlet) strani preverjanje pravilnosti vnešenih besed dosežemo na podoben način:

```
ReCaptchaImpl reCaptcha = new ReCaptchaImpl();
reCaptcha.setPrivateKey(privateKey);
ReCaptchaResponse reCaptchaResponse =
reCaptcha.checkAnswer(naslov, iziv, uporabnikovOdziv);
            if(reCaptchaResponse.isValid())
                  //pravilen vnos, nadaljuj z registracijo
```

4.5 Testiranje aplikacije

Pri testiranju sistema nas zanima nihanje cen delnic izidov dogodkov na napovednih trgih pri večji količini stav. S tem bomo ugotovili, kateri izmed trgov je bolj robusten in manj dovzeten za nenadne spremembe. Zanima nas tudi korelacija med vrednostmi izidov posameznega dogodka.

4.5.1 Testni scenarij

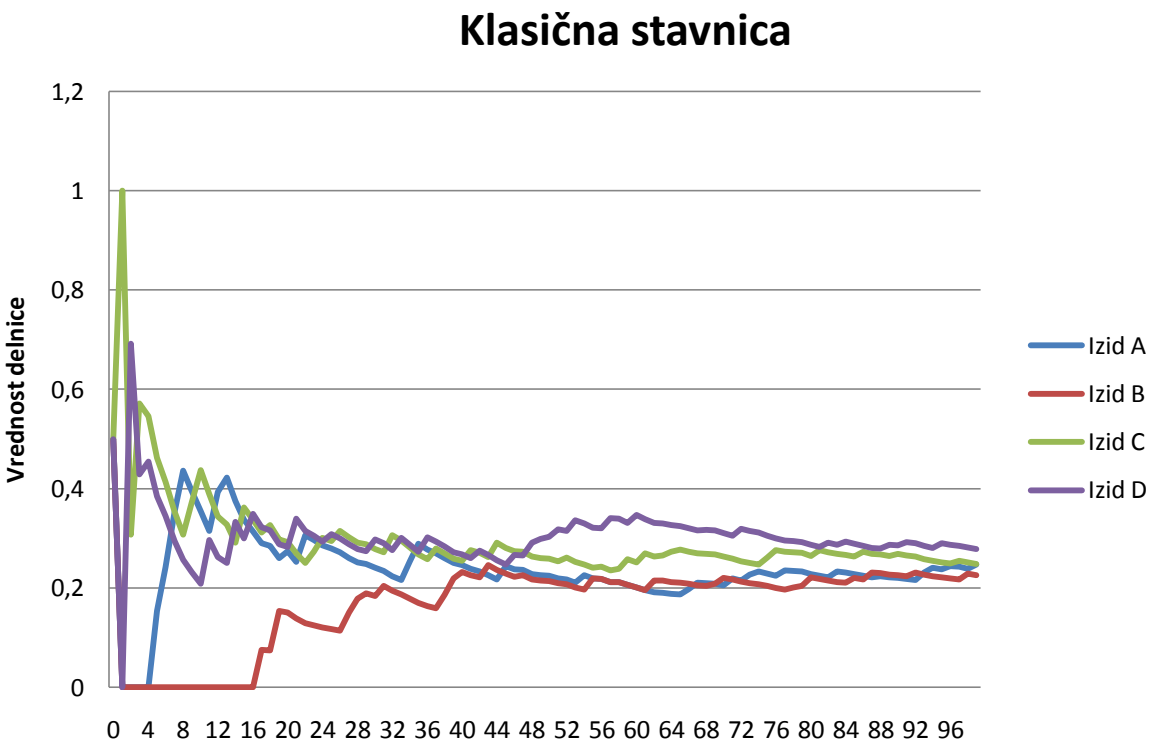
Testni scenarij predstavlja zaporedje 100 nakupov delnic oziroma vplačanih stav. Njihova vrednost je naključno določena, ravno tako je naključno določen izid dogodka, na katero se delnica oz. stava navezuje. Vse tri napovedne trge smo testirali z istim naborom testnih podatkov, ki so bili generirani naključno z enakomerno porazdelitvijo. Format datoteke s testnimi podatki je bil naslednji:

zaporedna številka delnice/stave	številka izida	višina stave/delnice
0	2	8
1	1	7
2	2	2
.		
.		

Za testiranje smo uporabili orodje Selenium, ki omogoča izvajanje testnih scenarijev na spletnih aplikacijah. Za vsak trg smo pripravili svoj testni scenarij, saj se vhodni parametri za vsakega izmed njih malenkostno razlikujejo.

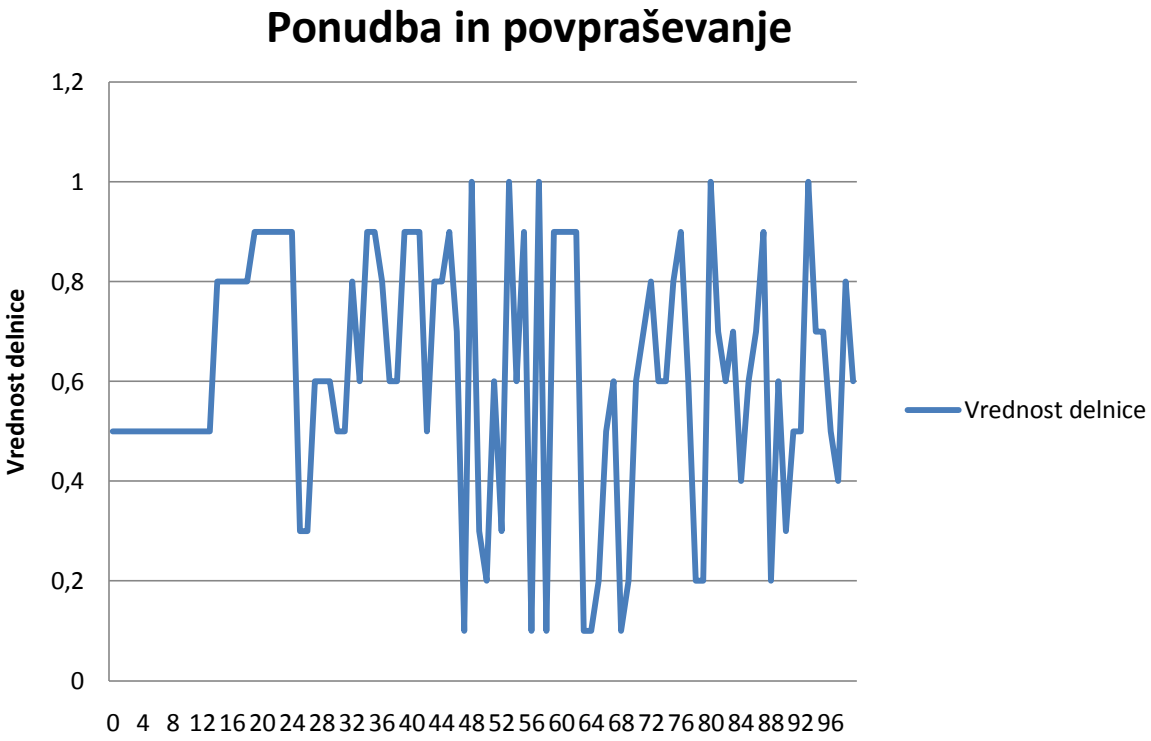
Pri testiranju klasične stavnice smo uporabili dogodek s 4 možnimi izidi, tako da je polje »številka izida« vsebovalo vrednosti med 1 in 4. Ker na trgu ponudbe in povpraševanja cene delnic nihajo med 1 in 10, je bil isti nabor vrednosti prisoten tudi v polju »višina stave/delnice«.

4.5.2 Rezultati testiranja



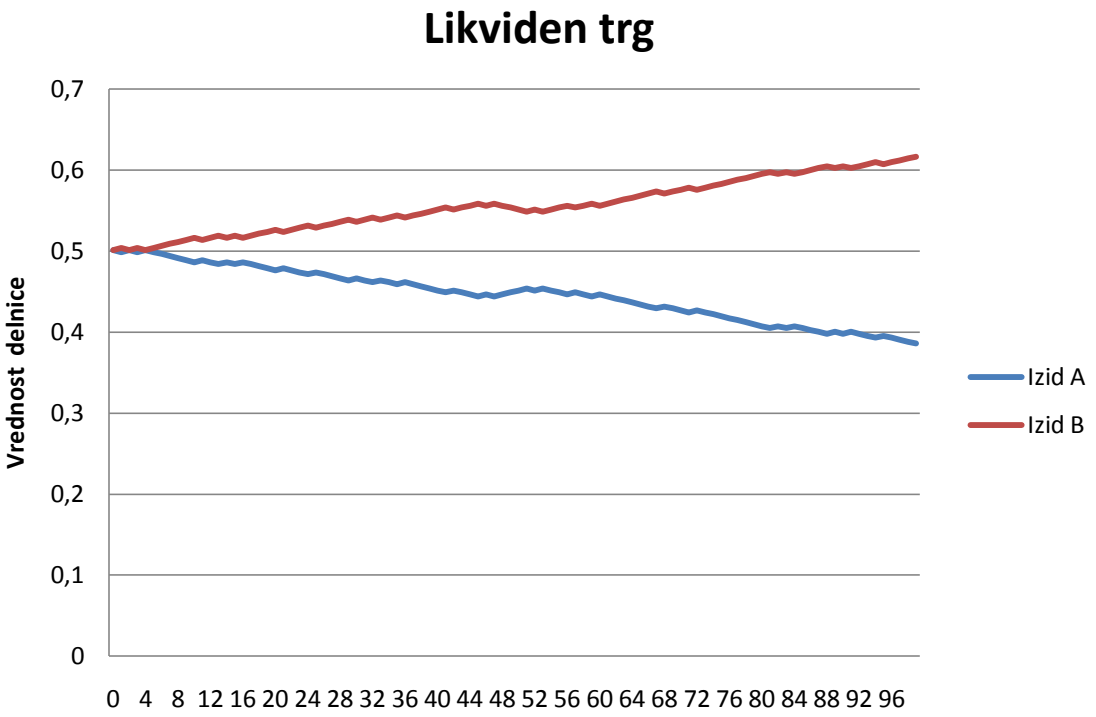
Slika 4.22: Graf vrednosti delnice na trgu klasične stavnice.

Na trgu klasične stavnice so na grafu (slika 4.22) opazna velika nihanja predvsem pri majhnem številu vplačanih stav, kar je logično, saj vrednost posamezne delnice pri stavnici predstavlja razmerje med vsoto vseh stav določenega izida in vsoto vseh stav dogodka. Proti koncu se nihanja bistveno zmanjšajo, kar je posledica boljše porazdeljenosti stav med izidi.



Slika 4.23: Graf vrednosti delnice na trgu ponudbe in povpraševanja.

Na grafu ponudbe in povpraševanja (slika 4.23) so opazna velika nihanja, kar je posledica dejstva, da je vrednost delnice na trgu ponudbe in povpraševanja enaka ceni zadnjega ujemajočega se para delnic obeh izidov. Velikim nihanjem botruje tudi nabor testnih podatkov, saj zaradi naključnosti ne odraža realnega stanja na trgu. Na trgu z resničnimi uporabniki je tako moč pričakovati manj drastične skoke vrednosti delnic.



Slika 4.24: Graf vrednosti delnice na likvidnem trgu.

Z grafa vrednosti delnice likvidnega trga (slika 4.24) je moč opaziti bistveno manj nihanj kot pri ostalih dveh. Ker vrednost delnice določa avtomatski vzdrževalec trga, lahko uporabniki kupujejo večje število delnic z isto ali podobno ceno, brez povzročanja večjih sprememb. Opazna je tudi korelacija med obema izidoma, saj je vrednost njunih delnic obratno sorazmerna.

Na klasični stavnici so nihanja pristona na začetku, pri majhnem število vplačanih stav, na trgu ponudbe in povpraševanj so večja nihanja možna v vseh fazah testiranja, medtem ko se je likviden trg izkazal za najbolj stabilnega, z največjo mero likvidnosti.

5. SKLEPNE UGOTOVITVE

V diplomski nalogi smo izdelali spletno aplikacijo, ki s pomočjo napovednih trgov in modrosti množic pridobiva ocene verjetnosti uresničitve raznovrstnih dogodkov. Aplikacija omogoča izvedbo celotnega življenjskega cikla dogodka, od dodajanja, izvajanja napovedi in trgovanja, do zaključka in pregledovanja natančnosti napovedi končanih dogodkov.

Pregledali smo teorijo napovednih trgov in modrosti množic, našli skupne značilnosti obeh konceptov in ju povezane uporabili v praksi. Na koncu smo implementirane napovedne trge testirali in njihove rezultate primerjali. Predstavili smo uporabljena programska orodja in utemeljili njihovo izbiro.

Najvažnejši prispevek je zagotovo implementacija likvidnega napovednega trga z uporabo avtomatskega vzdrževalca trga LMSR, ki je trenutno ena od najnaprednejših oblik napovednih trgov. Aplikacija za pridobivanje napovedi uporablja tri različne modele napovednih trgov, kar je precejšnja redkost, saj sorodne rešitve v večini primerov uporabljajo le enega samega.

Kljub uspešni implementaciji pa dejanske natančnosti napovedi, ki jih lahko z aplikacijo dobimo, ni bilo moč ustrezno preveriti. Ker je za reprezentativnost napovedi na napovednih trgih potrebno veliko število sodelujočih in daljši čas izvajanja, trenutno zbrani rezultati še nimajo prave teže.

Teorija modrosti množic napoveduje, da so napovedi dogodkov, pridobljene s spodbujenim trgovanjem, tako ali še bolj natančne od napovedi pridobljene z anketami. Rezultati, pridobljeni s pomočjo izdelane aplikacije imajo tako potencialno raziskovalno in tržno vrednost.

Priložnost za izboljšavo vidimo v bolj dejavnem motiviranju in nagrajevanju uporabnikov, izboljšanju sledenja vrednosti delnic, kategorizaciji dogodkov in uporabo borze kot modela napovednega trga. Borzni trg je večini ljudi poznan in bi lahko pritegnil širši krog uporabnikov. Ena od možnih izboljšav bi bila tudi integracija z socialnimi omrežji. Z njihovo uporabo bi rezultati napovedi dobili večji odziv, aplikacija pa bi pridobila informacije o profilu uporabnika in mu skozi čas predlagala dogodke s področij, ki so mu blizu.

LITERATURA IN VIRI

[1] (2011) Spletno mesto Hollywood Stock Exchange. Dostopno na http://www.hsx.com/.

[2] (2011) Spletno mesto Intrade. Dostopno na http://www.intrade.com/v4/home/.

[3] J. Šušteršič, S. Šušteršič, »Trgi kot orodje za napovedovanje: Primer slovenske volilne borze«, *IB Revija*, februar 2009, str. 51-53.

[4] (2011) Oddhead Blog, »Why automated market makers«. Dostopno na http://blog.oddhead.com/2010/07/08/why-automated-market-makers/.

[5] (2011) Oddhead Blog, »Implementing Hanson's Market Maker«. Dostopno na http://blog.oddhead.com/2006/10/30/implementing-hansons-market-maker/.

[6] J. Surowiecki, *The Wisdom of Crowds*, ZDA: Doubleday; Anchor, 2004.

[7] (2011) Spletno mesto Java EE. Dostopno na http://download.oracle.com/javaee/.

[8] (2011) Spletno mesto jQuery. Dostopno na http://jquery.com/.

[9] R. Hanson, »Logarithmic Market Scoring Rules for Modular Combinatorial Information Aggregation«, v zborniku *Journal of Prediction Markets*, Velika Britanija, februar 2007, str. 3-15.

[10] R. Hanson, »Combinatorial Information Market Design«, v zborniku *Information Systems Frontiers 5*, Nizozemska, 2003, str. 107-119.

[11] A. Othman, T. Sandholm, D. M. Pennock, D. M. Reeves, »A Practical Liquidity-Sensitive Automated Market Maker«, v zborniku *Proceedings of the 11th ACM conference on Electronic commerce*, ZDA, 2010, str. 377-386.

[12] A. Othman, T. Sandholm, »Automated Market-Making in the Large: The Gates Hillman Prediction Market«, v zborniku *Proceedings of the 11th ACM conference on Electronic commerce*, ZDA, 2010, str. 367-376.