UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Peter Karlovšek

KONCEPT PROSTO DOSTOPNIH PROGRAMSKIH ORODIJ ZA RAZVOJ APLIKACIJ

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Mira Trebar

Ljubljana, 2010

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: 00475/2009
Datum: 15.10.2009

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

Kandidat: **PETER KARLOVŠEK**

Naslov: **KONCEPT PROSTODOSTOPNIH PROGRAMSKIH ORODIJ ZA RAZVOJ APLIKACIJ**

CONCEPT OF FREE SOFTWARE TOOLS IN APPLICATIONS DESIGN

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Kandidat naj pregleda in analizira področje programskih tehnologij, orodij in programskih jezikov, ki omogočajo enostaven in zanesljiv razvoj aplikacij. V diplomski nalogi naj za izbrani koncept prostodostopnih orodij zasnuje in izdela aplikacijo za izračun potresne odpornosti zidanih zgradb. Na podlagi preizkusa naj oceni zahtevnost razvoja aplikacije in ovrednoti uporabnost predlagane rešitve.

Mentor:
doc. dr. Mira Trebar

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Franc Solina

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

diplomskega dela

Spodaj podpisani Peter Karlovšek,

z vpisno številko 63050398,

sem avtor diplomskega dela z naslovom:

KONCEPT PROSTO DOSTOPNIH PROGRAMSKIH ORODIJ ZA RAZVOJ APLIKACIJ

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mire Trebar
- 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 _____ Podpis avtorja: _____

Zahvala

Zahvaljujem se mentorici, doc. dr. Miri Trebar za nasvete, vodenje, pripombe in pregled diplomskega dela in Gregorju Gregorcu univ.dipl.inž.grad. iz podjetja Karlovšek d.o.o. za strokovno pomoč iz področja gradbene statike.

Hvala staršem za podporo in spodbudo v času študija.

Prijateljici pa se zahvaljujem za pregled in lektoriranje diplomske naloge.

Kazalo

Kratice

ESP	Encapsulated PostScript
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
JDK	Java Development Kit
JPEG	Joint Photographic Experts Group
OS	Operating system
PDF	Portable Document Format
PHP	PHP: Hypertext Preprocessor
PNG	Portable Network Graphics
SVG	Scalable Vector Graphics
SVN	Subversion
TCP	Transmission Control Protocol
URL	Uniform Resource Locator
WebDAV	Web-based Distributed Authoring and Versioning
XML	Extensible Markup Language

Povzetek

Diplomsko delo obravnava nabor prosto dostopnih programskih tehnologij, orodij in programskih jezikov za izdelavo aplikacij ter njihovo uporabo na primeru izdelave aplikacije ZIDEC.

Danes si življenja brez računalnikov ne predstavljamo več in čeprav si ob besedi računalnik ponavadi predstavljamo "škatlo", monitor, tipkovnico in računalniško miško, dejansko pa so računalniške aplikacije in programi tisti, ki nam olajšajo oz. omogočajo reševanje zapletenih problemov. Skozi razvoj programske opreme sta se v grobem oblikovala dva tabora in sicer komercialne lastniške in prosto dostopne programske rešitve. V času gospodarske recesije je za posameznike kot tudi za podjetja pomembno gospodarno ravnanje z denarnimi sredstvi. Eden izmed možnih načinov je tudi zamenjava komercialnih programskih rešitev s prosto dostopnimi.

V diplomskem delu so med množico prosto dostopnih rešitev izbrane tiste, ki nam v največji meri pomagajo pri izdelavi in vzdrževanju aplikacij kot so sistemi za vodenje projektov, revizij izvorne kode, virtualizacija razvojnega okolja in izbira programskega jezika. Poleg samega izbora programske oprema je opisan koncept, kako razvojno okolje postaviti v virtualnem stroju. Eden glavnih razlogov za uporabo izključno prosto dostopnih orodij je v čim večji meri znižati stroške izdelave in vzdrževanja aplikacij.

Namen aplikacije ZIDEC je poenostaviti in pospešiti dolgotrajen in zahteven postopek za izračun potresne odpornosti zidanih zgradb. Izdelana aplikacija se uporablja v podjetju Karlovšek d.o.o.

Ključne besede: prosto dostopna programska oprema**,** virtualizacija razvojnega okolja, revizija izvorne kode, spletni strežnik, sistem za vodenje projektov

Abstract

This thesis deals with a set of free software technologies, tools and programming languages that are used for application development and the utilization of these tools on the development of a specific application named ZIDEC.

Nowadays, we can hardly imagine life without computers, and even though the word *computer* more or less reminds us of the hardware box, screen, keyboard and computer mouse, the true value of problem solving is brought to us by computer applications and software. During the evolution of software, two main streams appeared: commercial ownership solutions and free ware. In these rough economic times many individuals and companies realize the importance of economical behaviour. One of the possible ways of economizing is changing commercial solutions with free ware.

In this diploma paper I have tried to choose from the many free ware solutions and find those that help in developing and up-keeping applications such as project management, code revisions, virtualization of development environment and the choice of programming language. Along with the choice of software, this paper describes the concept of setting a development environment in a virtual machine. One of the main reasons to use free ware is of course economizing, saving on the costs of development and up-keeping of the applications.

The main goal of the developed application ZIDEC is to simplify and speed up the long term process of calculating the seismic resistance of masonry building. The application is fully used in the Karlovšek d.o.o. company.

Key words: free software tools, virtualization of development environment, version-control system, web server, system for software development projects

1. Uvod

Že od pradavnine pa vse do danes si je človek pomagal z izdelovanjem orodij, ki so mu pomagali pri opravljanju vsakodnevnih opravil. Naj si gre za lok in puščico za lažje lovljenje, ali za škripec za lažje dvigovanje bremena ali vžigalnik za lažje netenje ognja.

V dobi informacijske tehnologije pa se poslužujemo predvsem računalniških programov, aplikacij in sistemov, ki nam prihranijo čas in energijo pri opravljanju časovno kompleksnih in ponavljajočih se problemov. V največji meri to zavzema obdelavo tekstovnih, numeričnih, statističnih, vizualnih, zvočnih in drugih podatkov. V zadnjih letih lahko spremljamo pravi razcvet novih programskih jezikov, spletnih, namiznih in strežniških tehnologij in pristopov za izdelovanje orodij. Pri tem pa moramo biti pazljivi, saj lahko na koncu za izdelavo orodja, ki bo reševal enak problem, z različnimi pristopi in tehnologijami nastanejo velike razlike v stroških izdelave in vzdrževanja.

Med študijem sem se pri delu na projektih srečeval z veliko različnimi računalniškimi aplikacijami in tehnologijami. Mnogo od njih je bilo komercialnih, kar pomeni, da je za njihovo uporabo potrebno plačati licenčnino. Veliko pa sem uporabljal tudi prosto dostopne aplikacije (Tabela 1). V veliki večini so po zmogljivosti enako dobre kot komercialni lastniški izdelki [20], v nekaterih primerih so celo bolj prilagodljive, vsekakor pa zadovoljijo potrebo po večini zahtev uporabnikov in so primerne za širšo uporabo, kar nakazuje tudi vztrajen pohod v slovensko [22] in evropsko [21] javno upravo.

	Lastniško	**Prosto dostopno**
Platforma	Windows, Mac OS	Ubuntu, Fedora
Orodja	Visual Studio	Eclipse
	VMware Workstation	VirtualBox
	Internet Information Service	Apache HTTP Server
	ClearCase, Team Fundation Server	Subeversion, Git
	Photoshop	Gimp
	Microsoft Office	OpenOffice
Programski jeziki	C#, ASP	Java, PHP

Tabela 1. Primerjalna tabela možnih zamenjav lastniških aplikacij s prosto dostopnimi in obratno.

V podjetju Karlovšek d.o.o., v katerem se že več kot 25 let ukvarjajo s projektiranjem gradbenih objektov, so izrazili željo po izdelavi namenske aplikacije za analizo potresne odpornosti zidanih zgradb. Poleg glavnih vodil pri izdelavi aplikacije so predlagali, da bi bila narejena z odprtokodno tehnologijo, za katero ne bi bilo treba plačevati licenčnin, kar bi pocenilo samo izdelavo in vzdrževanje aplikacije. Poleg premišljenega izbora samih programskih jezikov in programov, je zelo pomembna tudi postavitev celovitega razvojnega okolja.

Pri projektiranju nosilnih gradbenih konstrukcij je potrebno med drugim upoštevati tudi določbe Pravilnika o mehanski odpornosti in stabilnosti objektov [4], oziroma morajo biti nosilne gradbene konstrukcije izračunane in dimenzionirane skladno z načeli in pravili tudi pri nas sprejetih evropskih standardov (Evrokodi). Med temi standardi je tudi standard SIST EN 1998-1, imenovan tudi Evrokod 8, ki obravnava projektiranje potresno odpornih konstrukcij. Skladno z zahtevami tega standarda je potrebno vse zidane stavbe, ki ne sodijo v kategorijo »enostavnih zidanih zgradb«, analizirati na vplive potresne obtežbe, oziroma je potrebno izračunati njihovo potresno odpornost in preveriti varnost proti porušitvi. Zahteve za metodo takšnega izračuna so določene v standardu. Gre za razmeroma zahteven in zamuden iteracijski postopek, pri katerem t. i. »peš račun na roke« projektantu ni prijazen in ekonomsko upravičen. Ker na trgu trenutno ne obstaja komercialna verzija programske opreme za takšen izračun, so se v podjetju Karlovšek d.o.o. odločili za izdelavo računalniške aplikacije, ki pri izračunu upošteva vse zahteve standarda.

V Sloveniji obstaja ena aplikacija, ki je namenjena reševanju problema potresnih izračunov. Gre za aplikacijo **SREMB**, ki jo je razvilo podjetje Gradbeni inštitut ZRMK d.o.o. in je napisana v programskem jeziku Matlab. V primerjavi s predlagano rešitvijo je potrebno za zagon aplikacije SREMB kupiti licenco za program Matlab. Vnos podatkov je zapleten in poteka v več korakih. Na začetku se podatke vpiše v Microsoft Excel preglednico in se jih potem izvozi v formatu *vrednosti ločene z vejico* (ang. comma-separated values). Datoteka z izvoženimi podatki se vključi kot argument aplikaciji SREMB, ki se jo zažene preko ukazne vrstice v programu Matlab. Aplikacija je namenjen samo interni uporabi, tako da je ostalim projektantskim birojem nedostopna.

Na evropskem tržišču obstajata še dva programa, ki v določeni meri pokrivata reševanje podobnega problema (FEDRA [13] in POR 2000 [9]). Čeprav imajo države, ki so članice EU, enake zakone in predpise, pa je Slovenija s svojo geografsko lego dokaj specifičen primer, kar bi pomenilo potrebo po dodatni prilagoditvi programov.

Programi iz drugih držav po svetu niso primerni, ker imajo države različne predpise in zakone.

Namen diplomskega dela je prikaz koncepta izdelave programske opreme s pomočjo prosto dostopnih programskih tehnologij, orodij in programskih jezikov na primeru izdelave aplikacije za izračun potresne analize zidanih zgradb ZIDEC.

2. Postavitev strežnika

Med izdelavo aplikacije se je pokazala potreba po orodjih, ki bi olajšala nekatera opravila, naredila potek izdelave aplikacije bolj pregledno oz. skrbela za boljšo urejenost podatkov.

Trije glavni sklopi, ki sem jih potreboval za razvoj in izdelavo aplikacije, so:

- Spletna stran, kjer bo dostopna dokumentacija programa, podrobnejša navodila za uporabo ter primeri uporabe.
- Sistem za revizijo programske kode ter ustvarjanje varnostnih kopij.
- Sistem za vodenje projekta, z možnostjo dodeljevanja opravil, časovnim pregledom nad opravljenim ter nedokončanim delom in prikazom napredka.

Za vse tri sklope sem uporabil aplikacije, ki so se izvajale v strežniškem načinu, kar mi je omogočilo oddaljen dostop do njih neodvisno od računalnika, njegove strojne arhitekture, operacijskega sistema in lokacije.

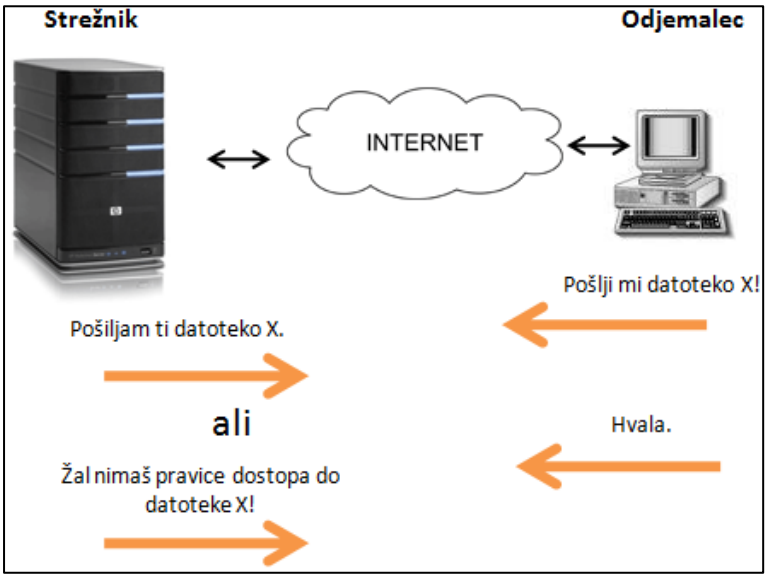
Imel sem dve možnosti in sicer: ali bi za storitve nadzora različic programske kode, sledenje programskih hroščev in wiki[1] sistem najel spletne servise, ali pa bi si potrebno infrastrukturo postavil sam. Na začetku sem mislil, da bi za vse to poskrbel preko spletnih ponudnikov. Po raziskovanju in testiranju pa sem ugotovil sledeče: v evropskem okolju je zelo težko najti ponudnika, ki bi vse to združeval na enem mestu za neko relativno nizko ceno. Pri ponudnikih iz drugih držav, predvsem iz ZDA, se je izkazalo, da so cene nižje ali celo brezplačne (v zameno nam prikazujejo oglase). Glavna slabost neevropskih ponudnikov je oddaljenost njihovih strežnikov, kar je posledično močno upočasnilo samo prikazovanje spletnih strani in delo s programi za revizijo kode.

Odločil sem se, da bom samostojen sistem postavil na starejšem, manj zmogljivem računalniku (AMD 1000 MHz, 265MB delovnega pomnilnika - RAM, 80 GB disk).

2.1. Odjemalec/strežnik

Najbolj pogost model v porazdeljenem sistemu je model odjemalec/strežnik (Slika 1), ki predstavlja tudi temelj interneta [1]. V porazdeljenem sistemu imamo enega ali več strežnikov, ki zagotavljajo storitve drugemu delu sistema, ki jim pravimo odjemalci. Ko se strežnik zažene, najprej odpre vrata (ang. port), preko katerih je dostopen. Po tem čaka v ozadju, da se nanj poveže odjemalec.

[1] wiki je koncept spletne strani, ki jo je mogoče enostavno urejati preko spletnega brskalnika

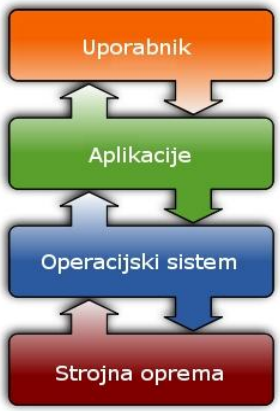


Slika 1. Potek komunikacije odjemalec/strežnik.

Da se odjemalec poveže na strežnik, mora vedeti njegov IP naslov oz. ime gostitelja (ang. hostname) ter številko vrat. Odjemalec mora strežniku pravilno oblikovati zahtevke in poskrbeti za komunikacijo z uporabnikom računalnika. Programi, ki zagotavljajo povezavo med odjemalcem in strežnikom, se sporazumevajo z enakim protokolom in so lahko tudi prisotni na istem računalniku. Program odjemalca uporabniku zagotavlja vmesnik za delo s strežnikovim programom. Ta sprejme ukaze, ki jih izvajamo na našem računalniku in jih pošlje na program strežnika. Na primer: uporabnik interneta lahko za priključitev in delo na HTTP strežniku uporablja program odjemalca Firefox, Opera, Internet Explorer in idr. Ko je vzpostavljena povezava po protokolu TCP/IP, odjemalec pošlje strežniku HTTP zahtevo za dostop do dokumenta. Ko strežnik prejme zahtevo, jo obdela in odjemalcu vrne dokument.

2.2. Operacijski sistem

Operacijski sistem služi kot vmesnik med strojno opremo računalnika in aplikacijami, ki jih poganja uporabnik (Slika 2).



Slika 2. Umestitev operacijskega sistema med strojno opremo in uporabnikom.

Glavne funkcije operacijskega sistema so:

- Upravljanje s sredstvi računalnika kot so: procesor, pomnilnik, periferne enote.
- Postavitev uporabniškega vmesnika.
- Izvajanje in podpora storitev za uporabniško programsko opremo.

Najbolj razširjeni operacijski sistemi so:

- Microsoft Windows
- Mac OS X
- GNU/Linux in ostali Unixu podobni operacijski sistemi

Operacijski sistem GNU/Linux (v nadaljevanju Linux) se je zdela najbolj primerna odločitev. Ponuja relativno visoko varnost, zanesljivost in prilagodljivost, zanj pa ni potrebno plačevati licenc ali vzdrževalnih pristojbin. Med številnimi namenskimi distribucijami sem izbral *Ubuntu Server Edition*.

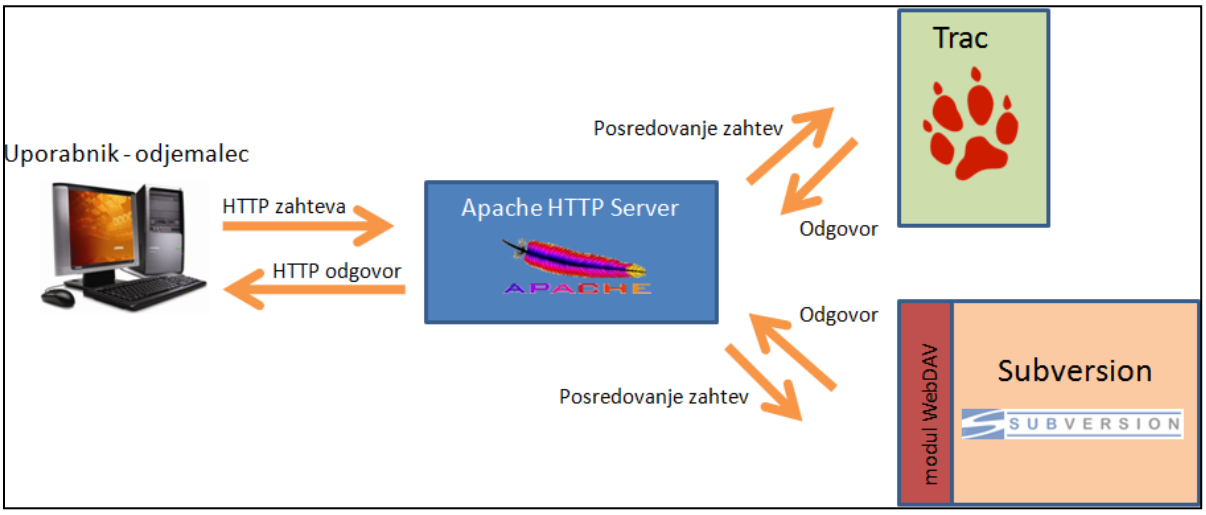
Za namestitev dodatnih programskih paketov sem uporabljal aplikacijo *apt-get*. Zagon aplikacije poteka iz ukazne vrstice. Pred nameščanjem sem osvežil sezname, iz katerih aplikacija prenese nove programske pakete: `sudo`[2] `apt-get update`.

2.3. Strežniške aplikacije

Povezava strežniških aplikacij (Slika 3) prikazuje, kako uporabnik s HTTP (ang. Hypertext Transfer Protocol) zahtevami dostopa:

- S spletnim brskalnikom preko strežnika Apache do sistema Trac.
- S SVN odjemalcem preko Apache strežnika z omogočenim modulom WebDAV[3] (ang. Web-based Distributed Authoring and Versioning) do SVN strežnika.

[2] *sudo* program nam omogoča, da ukaze in programe, ki mu jih podamo kot argument po vpisu administratorskega gesla, izvrši s privilegiji administratorja.
[3] WebDAV je skupek dodatkov k protokolu HTTP, ki uporabniku omogočajo skupno urejanje datotek na oddaljenem strežniku dostopnem preko svetovnega spleta.



Slika 3. Shema povezanosti strežnikov z zaporedjem dostopov.

Zaradi zahtevnih konfiguracij strežniških aplikacij ter njihove povezljivosti, so v nadaljevanju poleg osnovnih namembnosti in lastnosti opisani tudi postopki namestitve in konfiguracije.

2.4. Revizija izvorne kode

Upravljanje z izvorno kodo je že dolgo časa kritično početje za programerje. Posamezni razvijalci si s sistemi za revizijo izvorne kode pomagajo npr., kadar v izvorni kodi napravijo spremembe, ki jih nato želijo razveljaviti. V razvijalskih skupinah je običajno, da več razvijalcev hkrati popravlja določeno datoteko, ob koncu dneva pa je potrebno ugotoviti čigave spremembe so najboljše – sistemi za revizijo kode lahko pomagajo tudi pri tem.

Sam sem izbral odprtokodno rešitev **SVN** (Subversion) [2, 7], ki je trenutno najbolj razširjen sistem revizije. Da bi delo na različnih računalnikih potekalo kar najbolj optimalno, je pametno postaviti SVN strežnik, na katerega je mogoče dostopati preko interneta, saj v tem primeru delo lahko poteka tudi od doma ali s kake druge oddaljene lokacije. Strežnik pa ni nujno potreben za uporabo sistema Subversion, saj le-ta podpira sintakso URL za dostop do skladišča (http://, https://, svn://, svn+ssh://, file:/// itd.) in s tem omogoča, da je centralno skladišče izvorne kode na praktično kateremkoli računalniku, povezanem v lokalno omrežje. V tem primeru je do skladišča potrebno dostopati z URL predpono „file:///“.

SVN je sistem, ki deluje v ukazni vrstici, kar pogosto ni najbolj praktično. Obstaja veliko grafičnih vmesnikov, ki v ozadju izvajajo ukaze za SVN. Eden izmed njih je odprtokodni Subclipse [8] katerega dobra lastnost je, da se integrira v razvojno orodje Eclipse.

Glavne značilnosti sistema SVN:

- Pregled nad trenutnimi in prejšnjimi verzijami tekstovne datoteke.
- Ustvarjanje varnostne kopije izvorne kode in časovni pregled nad njimi.
- Večje število programerjev lahko istočasno ureja vsebino ene tekstovne datoteke.
- Podpira protokol WebDAV.
- Široka podpora v razvojnih orodjih.
- Pogosta uporaba v poslovnih okoljih.

Sistem Subversion namestimo z ukazom:

```
sudo apt-get install subversion
```

Z vpisom spodnjih ukazov ustvarimo skladišče (ang. repository) v mapi `/var/svn`:

```
cd /var
sudo mkdir svn
sudo svnadmin create /var/svn/repos
```

Za tem moramo določiti, kdo bo imel dostop do skladišča. Dodamo uporabnika *svn*, ki bo tudi lastnik datotek v tem skladišču. Z dodajanjem uporabnika dodamo tudi skupino z istim imenom:

```
sudo adduser --system --no-create-home --group svn
```

Sedaj moramo spremeniti pravice dostopa do datotek v skladišču:

```
sudo chown -R svn.svn svn
```

Če hočemo, da do datotek v skladišču dostopajo še drugi uporabniki, jih moramo dodati v skupino svn. Moje uporabniško ime je *karlos*. V skupino se dodam z ukazom `sudo vigr`.

Pri koncu datoteke poiščem vrstico, ki se začne s *svn* in dodam svoje uporabniško ime:

```
admin:x:110:karlos
svn:x:1001:karlos
```

2.5. Spletni strežnik

Spletni strežnik je programska aplikacija, ki odgovarja na zahteve po podatkih, prejete iz spletnih brskalnikov s pomočjo HTTP protokola.

Za obdelavo HTTP zahtev, dostopa do SVN strežnika, Trac sistema ter ostalih statičnih in dinamičnih spletnih strani sem izbral **Apache HTTP Server** [16]. Je najbolj popularen odprtokodni spletni strežnik [20] in edini s podporo sistemu SVN.

Glavne značilnosti Apache strežnika:

- Ponuja prenosljivost: lahko se namesti in deluje na več različnih operacijskih sistemih.
- Ima veliko dodatnih modulov, ki izboljšajo osnovne funkcionalnosti npr. mod_rewrite, mod _proxy, mod_auth, mod_gzip, idr.
- Nudi podporo splošno razširjenim jezikom kot so Perl, Python, PHP (ang. PHP: Hypertext Preprocessor) in Tcl.
- Z eno namestitvijo Apache strežnika lahko gostimo več različnih spletnih strani.

Namestitev strežnika Apache s podporo za SVN se izvede s potrditvijo ukaza:

```
sudo apt-get install apache2 libapache2-svn
```

Sledi urejanje konfiguracijske datoteke z ukazom:

```
sudo vi[4] /etc/apache2/sites-enabled/000-default,
```

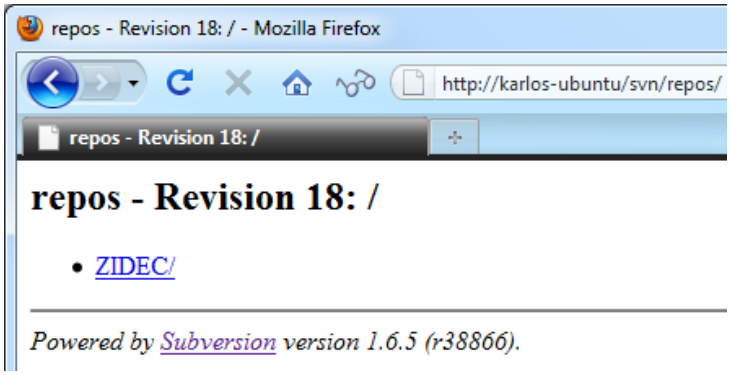
v katero dodamo kodo, za določanje poti do SVN skladišča z:

```
<Location /svn/repos>
   DAV svn
   SVNPath /var/svn/repos
</Location>
```

Shranimo spremembe in za njihovo uveljavitev ponovno zaženemo spletni strežnik z:

```
sudo /etc/init.d/apache2 restart
```

Po uspešni izvedbi bi morali v brskalniku imeti dostop do skladišča na URL (ang. Uniform Resource Locator) naslovu *http://imeracunalnika/svn/repos* (Slika 4).



Slika 4. Dostop do skladišča preko spletnega brskalnika.

[4] vi - konzolni urejevalnik tekstovnih datotek

Zavarovanje spletnega dostopa

Če hočemo omejiti pravice dostopa do skladišča, dodamo v datoteko */etc/apache2/sites-enabled/000-default* naslednjo kodo:

```
AuthType Basic
AuthName "Subversion Repository"
AuthUserFile /etc/apache2/passwords
Require valid-user
```

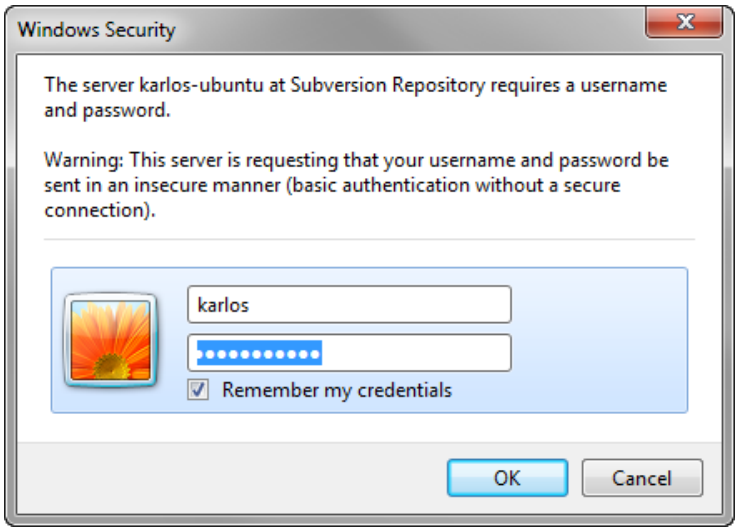
in v datoteko z gesli še uporabnika ter njegovo geslo:

```
sudo htpasswd -cbm5 /etc/apache2/passwords karlos geslo123
```

Za uveljavitev sprememb je potrebno izvesti ponovni zagon spletnega strežnika:

```
sudo /etc/init.d/apache2 restart
```

Če sedaj odpremo skladišče na prej omenjenem URL naslovu, se moramo avtorizirati s svojim uporabniškim imenom in geslom (Slika 5).



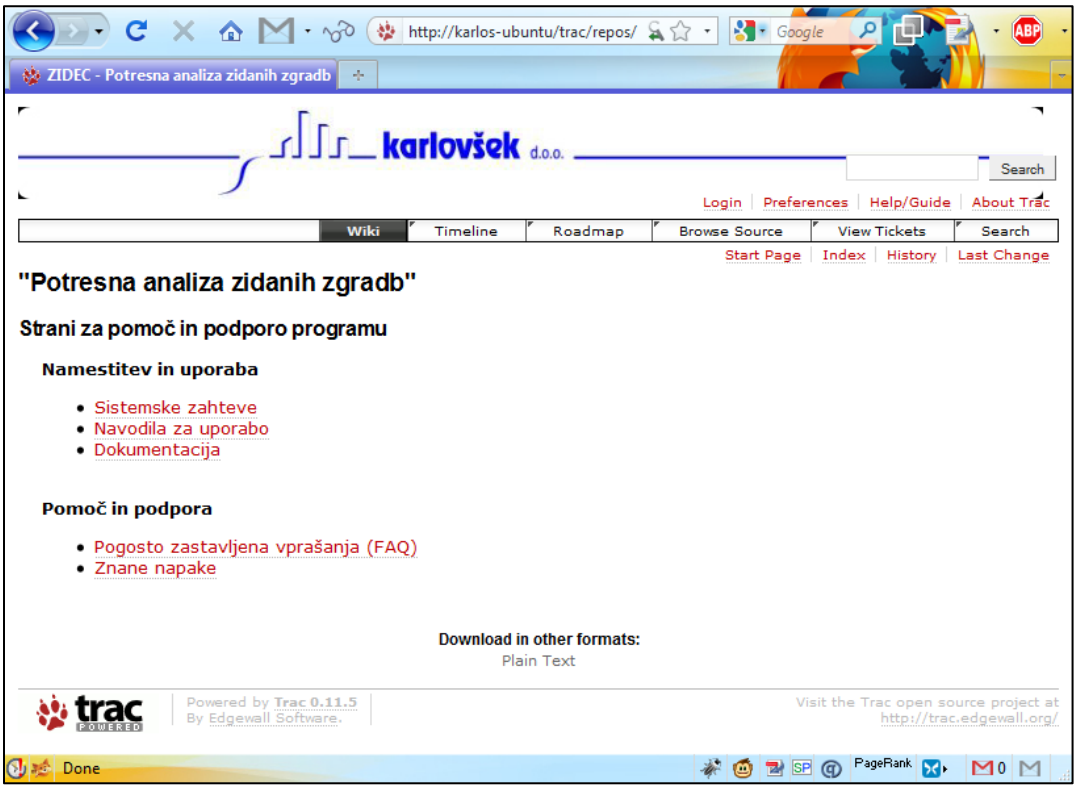
Slika 5. Avtorizacija z uporabniškim imenom in geslom.

2.6. Sistem za vodenje projektov

Za delo na projektu in sledenje programskim hroščem (Slika 6) sem uporabljal odprtokodno orodje **Trac** [6]. Na računalniku teče kot strežniška aplikacija, ki je dostopna preko spletnega brskalnika. Napisan je v programskem jeziku Python in ponuja integracijo s sistemom Subversion.

[5] Razlaga stikala "*-cbm*":

- -c: ustvari datoteko z gesli. Če datoteka že obstaja, jo prepiše.
- -b: namesto, da vpraša za geslo, geslo prebere iz ukazne vrstice podanega kot argument.
- -m: za gesla uporabi šifriranje MD5.



Slika 6. Osnovna stran sistema Trac.

Orodje Trac je razdeljeno na šest glavnih modulov:

1. **Wiki**

   Modul Wiki služi kot spletna stran za dokumentacijo razvijalcem in uporabnikom. Temelji na sistemu wiki, kar pomeni, da omogoča enostavno ustvarjanje in urejanje spletnih strani preko spletnega brskalnika. Omogoča nastavljanje pravic za dostop ali urejanje vsebine.

2. **Timeline**

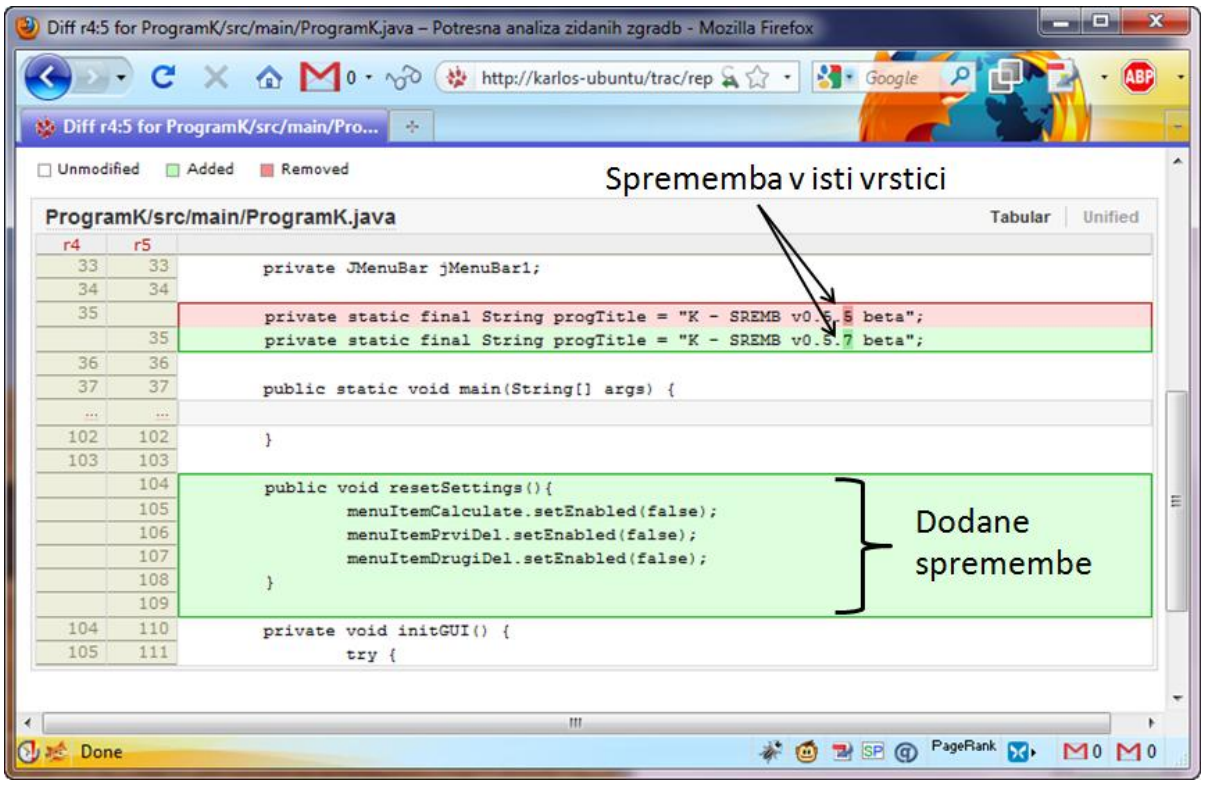
   Modul Timeline omogoča kronološki pregled dogodkov in sprememb, ki so se ustvarjale na celotnem sistemu Trac in Subversion.

3. **Roadmap**

   Modul Roadmap omogoča grafični pregled nad številom dodeljenih in opravljenih opravil.

4. **Brouse Source**

   V tem modulu lahko dostopamo do izvorne kode projektov. Omogoča tudi prikaz med različnimi revizijami posamezne tekstovne datoteke. Na sliki 7 je prikaz razlik med revizijama 4 in 5.



Slika 7. Prikaz sprememb v reviziji 4 in 5.

5. **View Tickets**

   Kot osrednji element sistema Trac, ta modul skrbi za dodajanje in spremljanje zahtev pri projektnih nalogah, programske izboljšave, poročila o napakah in vprašanjih za podporo programske opreme.

6. **Search**

   Modul Search omogoča iskanje vsebine po modulu Wiki in modulu Tickets ter po komentarjih (ang. Changesets) in mejnikih (ang. Milestones).

Sistem Trac za svoje delovanje potrebuje spletni strežnik, v tem primeru je to Apache, izvajalno okolje Python in podporo spletnega strežnika za Python. Vse to namestimo z vpisom spodnjega ukaza v terminal:

```
sudo apt-get install trac python-setuptools libapache2-mod-python enscript
```

Po uspešni namestitvi moramo ustvariti bazo Trac:

```
sudo mkdir /var/www/trac
sudo trac-admin /var/www/trac/repos initenv
```

Med konfiguracijo je potrebno odgovoriti na naslednja vprašanja (v oglatih oklepajih so podane privzete vrednosti):

- Ime projekta [My Project].
- Tip in pot podatkovne baze [sqlite:db/trac.db].
- Tip sistema za nadzor revizij [svn].
- Absolutna pot do skladišča SVN [/path/to/repos].

Večinoma lahko odgovorimo s privzetimi vrednostmi, z izjemo, ko nas vpraša za pot do skladišča SVN, kjer napišemo absolutno pot do njega, v mojem primeru `/var/svn/repos`.

Nastavitev spletnega strežnika Apache za zagon sistema Trac.

Za dostop do sistema Trac preko spletnega strežnika Apache je potrebno spremeniti pravice datotek v mapi /var/www/trac. S spodnjim ukazom rekurzivno spremenimo lastništvo datotek v mapi trac uporabniku www-data, ki pripada strežniku Apache:

```
sudo chown -R www-data.svn /var/www/trac
```

Ponovno je potrebno urediti datoteko `/etc/apache2/sites-enabled/000-default`, v katero dodamo spodnjo kodo:

```
<Location ~ "/trac/\w+/login">
  AuthType Basic
  AuthName "Subversion Repository"
  AuthUserFile /etc/apache2/passwords
  Require valid-user
</Location>

<Location /trac>
  SetHandler mod_python
  PythonInterpreter main_interpreter
  PythonHandler trac.web.modpython_frontend
  PythonOption TracEnvParentDir /var/www/trac
  PythonOption TracUriRoot /trac
</Location>
```

S spodnjim ukazom uveljavimo nove spremembe nastavitev za spletni strežnik Apache:

```
sudo /etc/init.d/apache2 force-reload
```

Sedaj lahko preko spletnega brskalnika dostopamo do sistema Trac na URL naslovu *http://imeracunalnika/trac* (Slika 8).

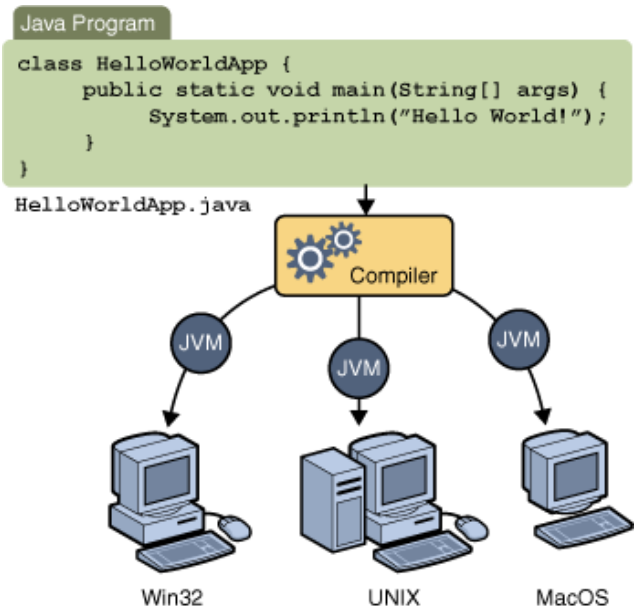
Slika 8. Seznam dostopnih projektov v sistemu Trac.

3. Razvojne tehnologije in programi

V tem poglavju so predstavljena vsa programska orodja in tehnologije, ki so pripomogla k izdelavi aplikacije. V vsakem opisu so napisane glavne značilnosti oz. kako so bila orodja in tehnologije vključena v proces izdelave aplikacije oz. njihova vključitev v samo aplikacijo.

3.1. Programski jezik Java

Java [17] je objektno usmerjen programski jezik, ki omogoča razvoj in distribuiranje strojno neodvisne programske opreme. Zasnova je zamišljena tako, da prevajalnik prevede izvorno kodo v vmesno kodo (ang. bytecode), ki je sestavljena iz množice platformno neodvisnih ukazov. Ti tečejo na t.i. virtualnem stroju Java, oz. jih interpretira Java Runtime Environment. Tak način delovanja omogoča razvijalcu programske opreme, da aplikacijo napiše enkrat, le-ta pa se lahko izvaja na več različnih platformah (Slika 9).



Slika 9. Ista aplikacija se lahko preko Javinega virtualnega stroja poganja na različnih platformah.

Glavne značilnosti Jave so:

- enostavnost,
- zanesljivost,
- neodvisnost od spodaj ležeče platforme,
- varnost,
- dinamičnost,
- veliko že napisanih knjižnic in

- za razvijanje in poganjanje programov napisanih v programskem jeziku Java ni potrebno plačevati licenc.

Razvijalcem so na voljo 3 različne izdaje, glede na to, kakšen produkt razvijajo [18]:

- Java tehnologija za majhne in mobilne naprave (Java Micro Edition),
- Java tehnologija za osebne računalnike (Java Standard Edition) in
- Java tehnologija za poslovna okolja (Java Enterprise Edition).

Izdaja Java Micro Edition je namenjena razvoju in poganjanju aplikacij za naprave z zelo omejenimi viri, kot so npr. mobilni telefoni, tiskalniki in vgrajeni sistemi (ang. embedded systems).

Izdaja Java Standard Edition je namenjena razvoju varnih, prenosnih in visoko zmogljivih aplikacij za najširši krog namiznih računalnikov, neodvisnih od operacijskih sistemov.

Končnim uporabnikom je namenjen paket JRE (Java Runtime Environment), ki služi izvajanju programov in aplikacij.

3.2. Razvojno orodje Eclipse

Razvoj programov in aplikacij je mnogo lažji, če uporabljamo razvojna orodja. Razvoj bi bil mogoč tudi z enostavnimi aplikacijami kot so **Notepad** v Windows OS, **vim** v Unix OS in **TextEdit** v Mac OS, vendar nam razvojna orodja nudijo veliko prednosti, med drugim tudi:

- preverjanje sintaktične pravilnosti kode med pisanjem,
- razhroščevanje kode,
- prikaz pomoči in definicije metod in razredov,
- prevajanje kode,
- integriranje s skladišči za izvorno kodo (ang. source code repositories) in
- označevanje določenih segmentov kode za lažjo orientacijo in navigacijo po kodi.

Eclipse [19] je integrirano razvojno orodje (ang. Integrated Development Environment - IDE), ki se uporablja za razvoj konzolnih aplikacij, aplikacij z grafičnimi vmesniki, spletnih aplikacij, strežniških servisov, mobilnih aplikacij in podobno. Prvotno je bil izdelan za razvoj aplikacij napisanih v programskem jeziku Java. Z razvojem dodatnih vtičnikov sedaj podpira velik nabor priljubljenih programskih jezikov kot so C, C++, Python, PHP, HTML (ang. HyperText Markup Language), Javascript in druge. Eclipse odlikujejo tudi številni brezplačni vtičniki, ki še izboljšajo njegovo zmogljivost ter velika skupnost razvijalcev, ki popravljajo programske hrošče in dodajajo nove funkcionalnosti. Pri razvoju programa sem uporabljal vtičnika *Subclipse* za podporo razvoja različice kode SVN in *Jigloo SWT/Swing GUI Builder* za pomoč pri načrtovanju grafičnega vmesnika.

3.3. Java logging framework

Java logging framework je paket za beleženje dogodkov v okolju Java. Pri razvoju večjega projekta je priporočljivo postaviti sistem za beleženje pomembnejših dogodkov. V primeru, da se aplikacija ne odziva po pričakovanjih, je s pomočjo dnevnikov lažje odkrivati in popravljati napake.

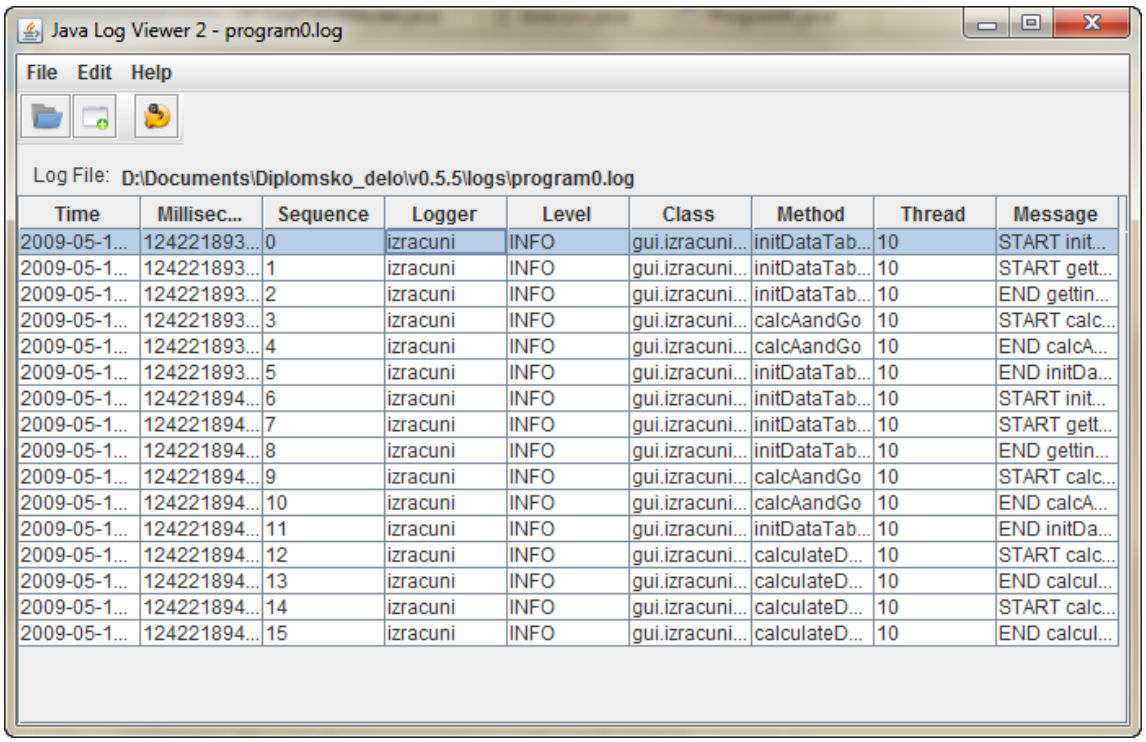
Glavne značilnosti:

- Vgrajen v izvajalno okolje Jave.
- Uporablja se ga preko Javanskega vmesnika za namensko programiranje beleženja (ang. Java Logging Application Programming Interface).
- Podpira 7 nivojev beleženja dogodkov, ki so razvrščeni po pomembnosti (našteti od najbolj do najmanj kritičnega):

SEVERE	Resna napaka, ki povzroči predčasno prekinitev delovanja. Napaka je takoj vidna tudi v statusni konzoli.
WARNING	Nakazuje slabo uporabo vmesnikov API, kar lahko privede do kritičnih napak. Napaka je takoj vidna tudi v statusni konzoli.
INFO	Dogodki namenjeni prikazovanju poteka izvajanja programa.
CONFIG	Dogodki, ki se izvajajo med inicializacijo programa.
FINE, FINER, FINEST	Prikaz vseh možnih dogodkov, namenjeno lažjemu razhroščevanju programske kode.

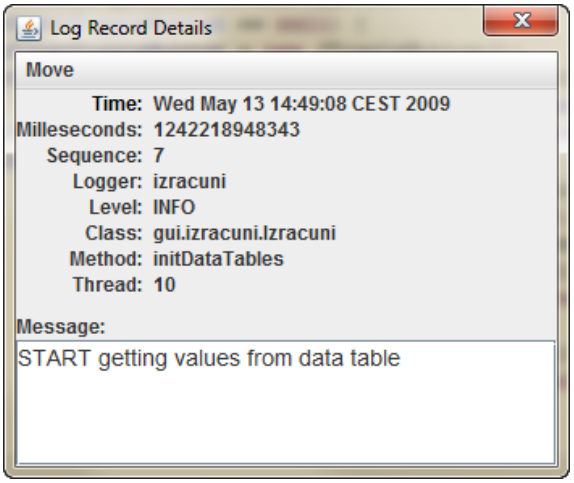
Java Log Viewer

Java Log Viewer [12] je brezplačni in odprtokodni program za pregledovanje dnevniških datotek ustvarjenih s pomočjo Java Logging API (Slika 10). Namenjen je prvi in osnovni diagnostiki v primeru napak izračunov ali pri delovanju samega programa. Program je zaščiten pod licenco LGPL, ki omogoča spreminjanje kode. Za lažje delo s programom sem mu dodal dve funkcionalnosti in sicer zmožnost premikanja po podrobnih dnevniških zapisih (Slika 11) s pomočjo smernih puščic (gor, dol), ter zmožnost filtriranja zapisov po prioriteti opozoril (severe, warning, info, config, fine, finer in finest).



Time	Millisec...	Sequence	Logger	Level	Class	Method	Thread	Message
2009-05-1...	124221893...	0	izracuni	INFO	gui.izracuni...	initDataTab...	10	START init...
2009-05-1...	124221893...	1	izracuni	INFO	gui.izracuni...	initDataTab...	10	START gett...
2009-05-1...	124221893...	2	izracuni	INFO	gui.izracuni...	initDataTab...	10	END gettin...
2009-05-1...	124221893...	3	izracuni	INFO	gui.izracuni...	calcAandGo	10	START calc...
2009-05-1...	124221893...	4	izracuni	INFO	gui.izracuni...	calcAandGo	10	END calcA...
2009-05-1...	124221893...	5	izracuni	INFO	gui.izracuni...	initDataTab...	10	END initDa...
2009-05-1...	124221894...	6	izracuni	INFO	gui.izracuni...	initDataTab...	10	START init...
2009-05-1...	124221894...	7	izracuni	INFO	gui.izracuni...	initDataTab...	10	START gett...
2009-05-1...	124221894...	8	izracuni	INFO	gui.izracuni...	initDataTab...	10	END gettin...
2009-05-1...	124221894...	9	izracuni	INFO	gui.izracuni...	calcAandGo	10	START calc...
2009-05-1...	124221894...	10	izracuni	INFO	gui.izracuni...	calcAandGo	10	END calcA...
2009-05-1...	124221894...	11	izracuni	INFO	gui.izracuni...	initDataTab...	10	END initDa...
2009-05-1...	124221894...	12	izracuni	INFO	gui.izracuni...	calculateD...	10	START calc...
2009-05-1...	124221894...	13	izracuni	INFO	gui.izracuni...	calculateD...	10	END calcul...
2009-05-1...	124221894...	14	izracuni	INFO	gui.izracuni...	calculateD...	10	START calc...
2009-05-1...	124221894...	15	izracuni	INFO	gui.izracuni...	calculateD...	10	END calcul...

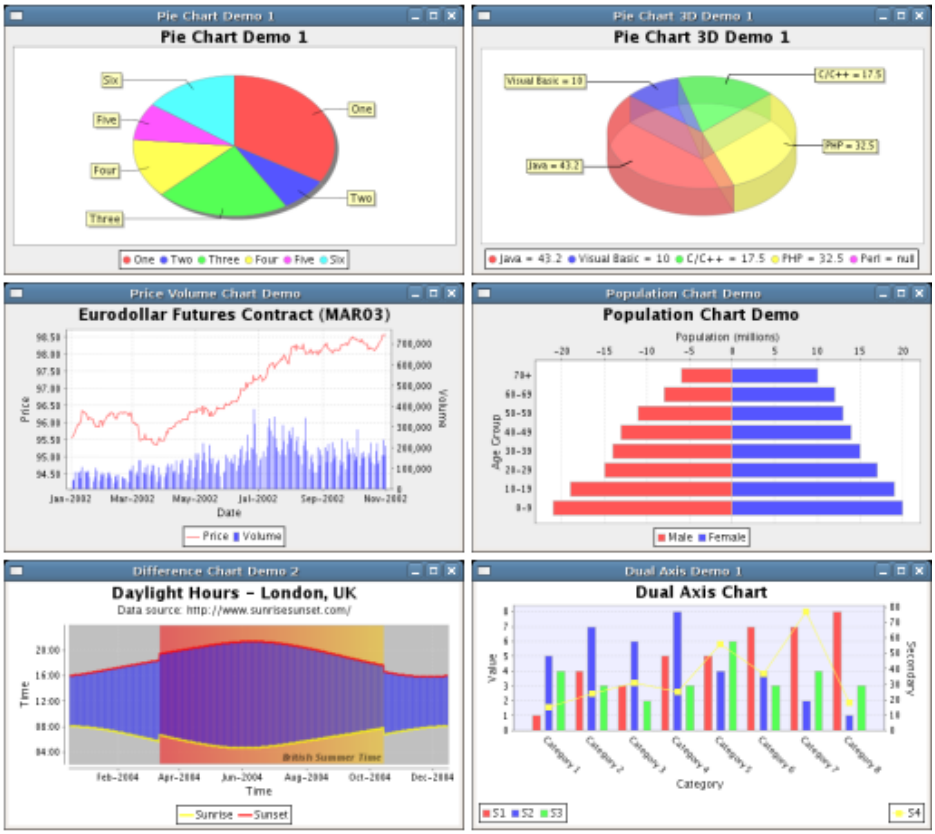
Slika 10. Pregled zapisov dogodkov v dnevniku po izračunu potresne odpornosti.



Slika 11. Podrobni izpis enega zapisa v dnevniku.

3.4. JFreeChart

JFreeChart [11] je programska knjižnica za prikazovanje podatkov v obliki grafov (Slika 12). Napisana je v programskem jeziku Java. Omogoča izvoz podatkov kot slike PNG (ang. Portable Network Graphics) in JPEG (ang. Joint Photographic Experts Group) ali kot vektorsko grafiko v formatih PDF (ang. Portable Document Format), EPS (ang. Encapsulated PostScript) in SVG (ang Scalable Vector Graphics). V programu ZIDEC je uporabljena za prikaz končnih rezultatov (Slika 26, stran 35).



Slika 12. Primeri grafov izrisanih s pomočjo JFreeCharts knjižnice.

3.5. Shranjevanje in urejanje podatkov

Podatki in nastavitve aplikacije ZIDEC se shranjujejo v datoteke, ki so strukturirane v obliki XML (ang. Extensible Markup Language) [23]. Ta način omogoča enostaven prenos podatkov in nastavitev med računalniki, ustvarjanje varnostnih kopij, sam zapis XML pa tudi omogoča dokaj enostavno luščenje potrebnih podatkov iz datotek ter njihovo zapisovanje. Razdeljen je na 3 dele:

- Podatkovni - vanj shranjujemo podatke v obliki, ki jo sami strukturiramo z želenimi značkami (ang. tag).

- Deklarativni - skrbi za to, da lahko pri dodajanju novih podatkov vidimo, kaj predstavlja posamezna značka.

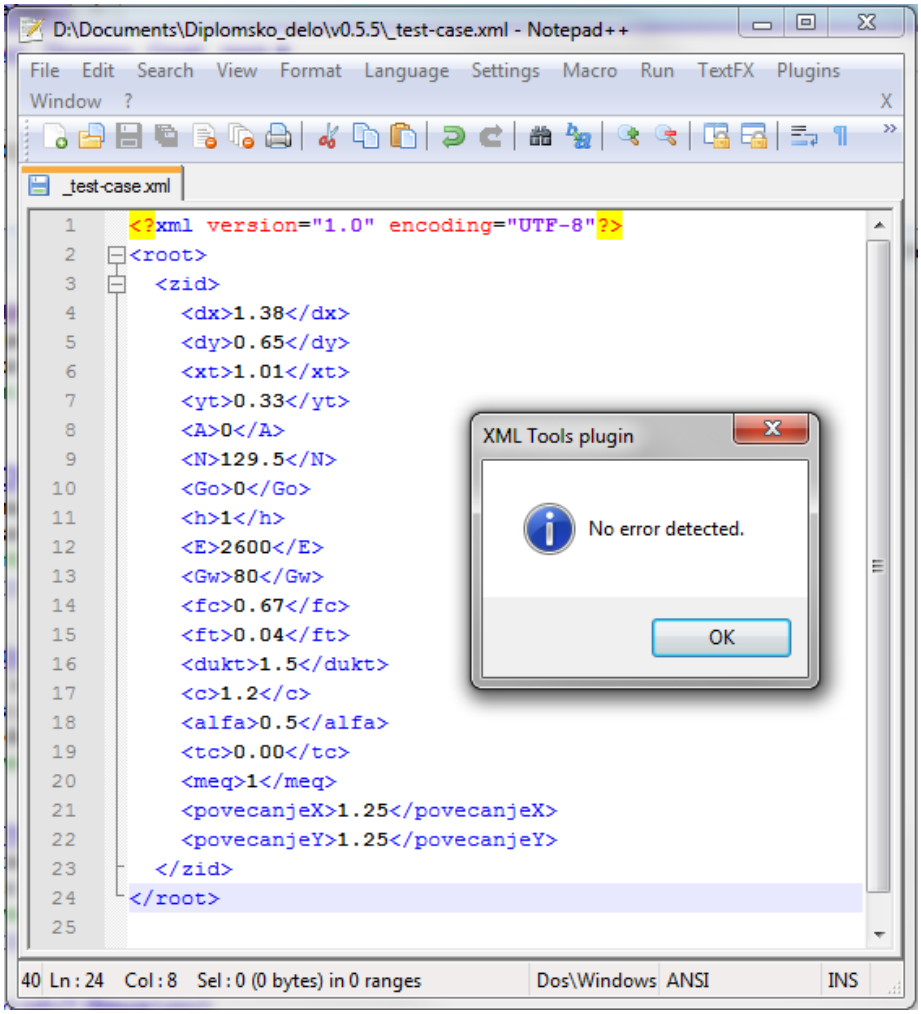
- Predstavitveni - z njim oblikujemo izpis podatkov.

XML je pravilen, če je ustrezno oblikovan (izpolnjuje vsa sintaktična pravila) in veljaven. Je generično ogrodje za hrambo poljubne dolžine teksta ali podatkov, katerih struktura je prikazana v obliki drevesa. Sintaktična pravila določajo, da:

- morajo vsi XML elementi imeti končno značko,
- so značke občutljive na male/velike črke,

- morajo biti elementi XML pravilno gnezdeni in
- XML dokument mora imeti en element, ki bo korenski (ang. root element).

Sintaktičnih pravil je še veliko več, vendar za predstavitev XML jezika bralcu ni potrebno poznati vseh. Slika 13 prikazuje primer pravilno oblikovanega XML dokumenta, ki izpolnjuje vsa zgoraj našteta pravila. Sintaktično pravilnost dokumentov se preverja z *XML Tools* dodatkom v programu Notepad++[6], ki je zmogljiv in hiter urejevalnik tekstovnih datotek [10].



Slika 13. Primer XML zapisa podatkov za en zid in preverjanje sintaktične pravilnosti zapisa XML v aplikaciji Notepad++.

3.6. Virtualizacija razvojnega okolja

Med opravljanjem obvezne študijske prakse se je kot zelo praktična metoda izkazala postavitev razvojnega okolja z vsemi potrebnimi programi na operacijskem sistemu, ki ga poganjamo v virtualnem okolju. Tak način ponuja številne prednosti:

[6] Notepad++ je napisan za operacijski sistem Micorsoft Windows, vendar se ga da namestiti in poganjati tudi na Linuxu preko emulatorja Wine.

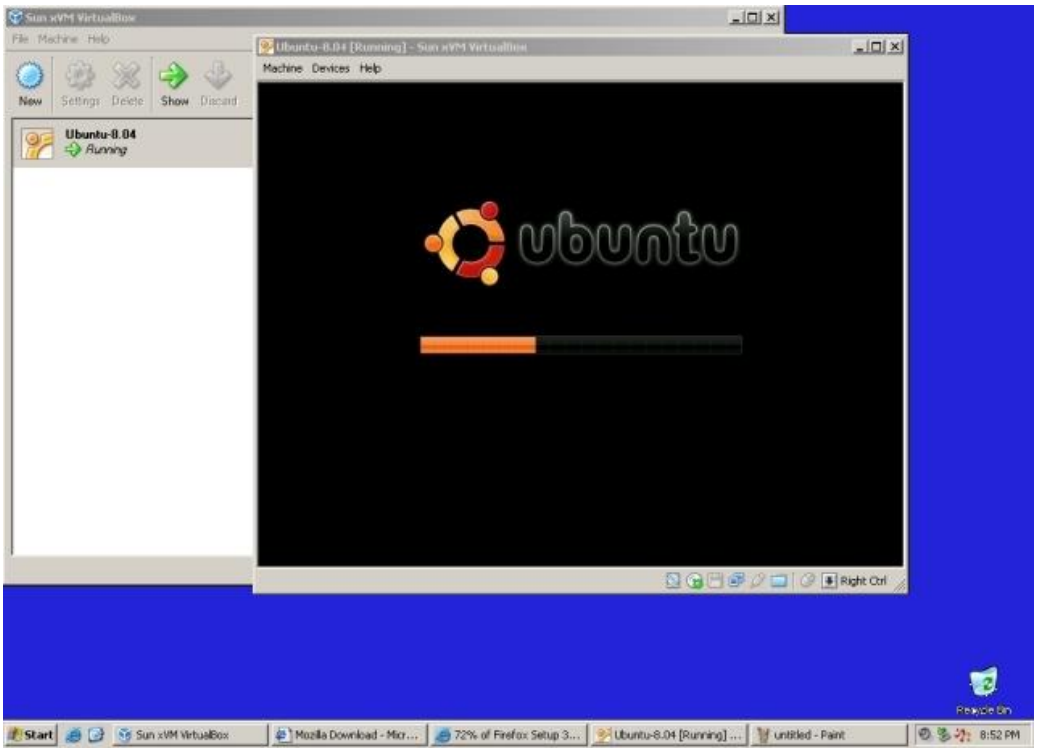
- Enostavno in hitro shranimo pravilno nastavljen sistem za razvoj aplikacij.

- Ohranimo verzije aplikacij, s katerim smo razvijali na projektu, četudi bi za drugi projekt potrebovali starejše ali novejše verzije.

- V primeru razširitve števila sodelujočih na istem projektu se lahko delovno okolje zelo hitro in enostavno kopira na nov računalnik.

Za sočasno delovanje dveh operacijskih sistemov (gostiteljski in virtualizirani) potrebujemo zmogljiv računalnik. Predvsem je potrebna večja količina pomnilniških modulov (RAM). V zadnjih letih smo priča velikem napredku na področju strojnih zmogljivosti osebnih računalnikov, zatorej to ne predstavlja večjega problema. Virtualno razvojno okolje je zadovoljivo hitro in odzivno delovalo na računalniku s karakteristikami: procesor AMD 3000, RAM 1˙5 GB, disk 150GB z 7200 obr./min.

Virtualizacijo operacijskih sistemov omogoča več programskih produktov. Med bolj znanimi in uporabljanimi so WMware, VirtualBox in VirtualPC. Izbral sem VirtualBox, ki ponuja širok spekter možnosti poganjanja in gostovanja operacijskih sistemov, ter izdelovanja novih virtualnih strojev (Slika 14).



Slika 14. Zagon Ubuntu distribicije Linux-a znotraj Windows XP operacijskega sistema.

4. Izvedba aplikacije ZIDEC

Aplikacija ZIDEC je programska izvedba priporočila Državne tehnične pisarne o tem, kaj mora vsebovati račun konstrukcij kot del projektne dokumentacije za rekonstrukcijo objektov v okviru popotresne obnove Posočja. Zasnovana je na podlagi dokumenta *OPERATIVNO NAVODILO DRŽAVNE TEHNIČNE PISARNE za izdelavo seizmične analize zidanih konstrukcij v okviru popotresne obnove objektov v Posočju* [3].

4.1. Zahteve za izdelavo aplikacije

Glavna vodila pri izdelavi in načrtovanju aplikacije so bila:

- Izdelava aplikacije na primeru *Operativnega navodila Državne tehnične pisarne za izdelavo seizmične analize zidanih konstrukcij v okviru popotresne obnove objektov v Posočju.*
- Enostaven in intuitiven vnos podatkov.
- Shranjevanje vnesenih podatkov.
- Pregledna oblika končnih rezultatov.
- Enostaven izvoz grafičnega prikaza končnih rezultatov v urejevalnike besedila.

4.2. Postavitev razvojnega okolja

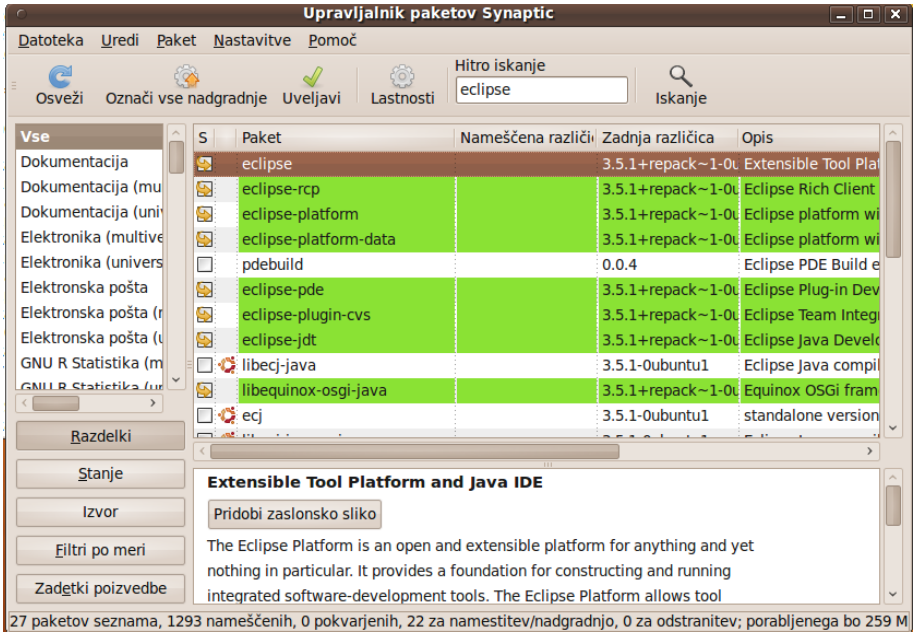
Razvojno okolje sem postavil in prilagodil v virtualnem stroju Linux operacijskega sistema, z uporabo programa VirtualBox [15]. Po prenosu programa VirtualBox iz njihove uradne spletne strani in namestitvi na gostujoči operacijski sistem (Windows XP), sem naredil novo navidezno napravo.

Preko čarovnika za ustvarjanje novih navideznih naprav sem moral določiti naslednje parametre (v oklepaju so podane vrednosti, ki sem jih izbral):

- Ime, pod katerim bo shranjena navidezna naprava (Razvoj-ZIDEC)
- Operacijski sistem (Linux)
- Različica (Ubuntu)
- Velikost RAM (702 MB)
- Ustvarjanje navideznega diska
    1. Dinamično večaj velikost diska (izbrano)
    2. Velikost (4,12 GB)

Med mnogimi distribucijami Linux-a sem izbral Ubuntu [5], trenutno najbolj popularno distribucijo z veliko skupnostjo in dobro pomočjo in podporo [14]. Namesto CD-ROM-a sem v VirtualBox-u nastavil pot do zadnje različice namestitvene slike (ang. iso image) Ubuntu distribucije verzije *9.10*.

Po zagonu navidezne naprave se je sistem naložil z vsebino zgoraj omenjene namestitvene slike ter mi ponudil namestitveni čarovnik. Po izbiri jezika, kode tipkovnice, uporabniškega imena in gesla se je začela namestitev Ubuntu Linux distribucije. Po končani namestitvi operacijskega sistema sem začel z nameščanjem razvojne različice programskega jezika Java (Java JDK (ang. Java Development Kit) 1.6) in integriranega razvojnega orodja Eclipse (Slika 15).

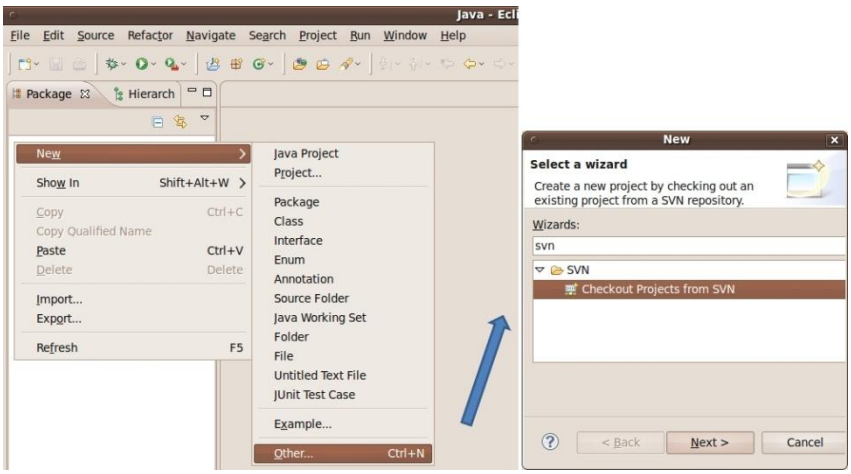


Slika 15. Nameščanje Eclipsa preko aplikacije Synaptic.

4.3. Eclipse: ustvarjanje projekta in povezava s SVN strežnikom

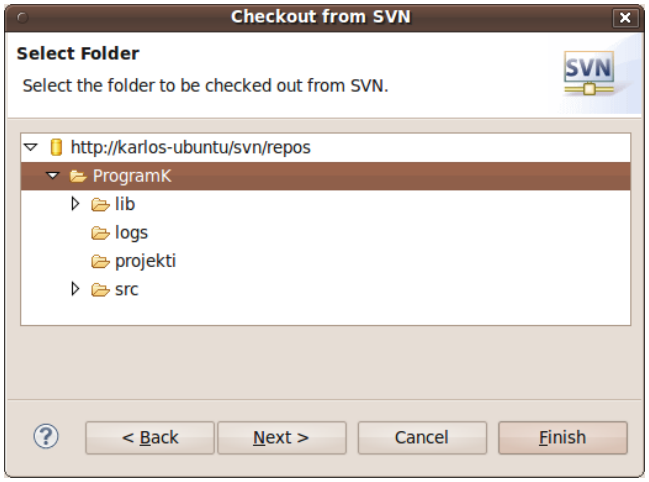
Za delo in povezavo s prej postavljenim SVN strežnikom sem moral v aplikaciji Eclipse namestiti še vtičnik Subclipse. Sama namestitev vtičnika v razvojno orodje je na enostaven in pregleden način opisana na njihovi spletni strani [8].

Naredil sem nov projekt SVN (Slika 16), kateremu sem za URL lokacijo podal pot do SVN skladišča. Ker sem dostop do skladišča zaščitili, sem moral vpisati še uporabniško ime in geslo.



Slika 16. Ustvarjanje novega projekta v Eclipsu s podporo za SVN.

Potem, ko sem izbral korensko mapo skladišča (Slika 17), ki se poveže z mapo v računalniku, je sledil običajni postopek ustvarjanja novega projekta v aplikaciji Eclipse.



Slika 17. Izbor korenske mape iz skladišča SVN.

Projekt sem razdelil na več logično povezanih sklopov, ki sem jih v mapah razdelil po drevesni strukturi (Slika 18). Vso izvorno kodo sem pisal v projektni mapi *src*. Od tu naprej sem projekt razdelil na 6 paketov (ang. package):

1. **Main** skrbi za osnovni izris aplikacije na zaslonu, kot tudi za prikaz osnovnih menijev (Datoteka, Izračuni, Akcije in Pomoč) ter prestrezanje na njihove dogodke (ang. action listener).

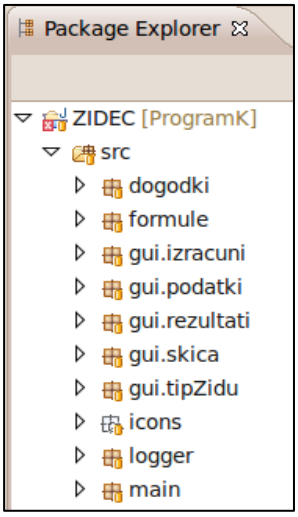
2. Paket **gui** sem še bolj podrobno razdelil na 5 glavnih grafičnih gradnikov aplikacije (izracuni, podatki, rezultati, skica in tipZidu). Vsak grafični gradnik poskrbi za svoj izris na glavni delovni plošči aplikacije (ang. Panel) ter za prestrezanje uporabnikovih interakcij preko tipkovnice in računalniške miške.

3. V paketu **dogodki** so napisane vse funkcije, ki obravnavajo uporabnikovo interakcijo zaznano preko grafičnih gradnikov.

4. Paket **formule** vsebuje vse potrebne enačbe za matematične izračune, ki jih rešuje aplikacija. Glavne enačbe so podane v dodatku B.

5. Paket **logger** skrbi za beleženje dogodkov med izvajanjem aplikacije.

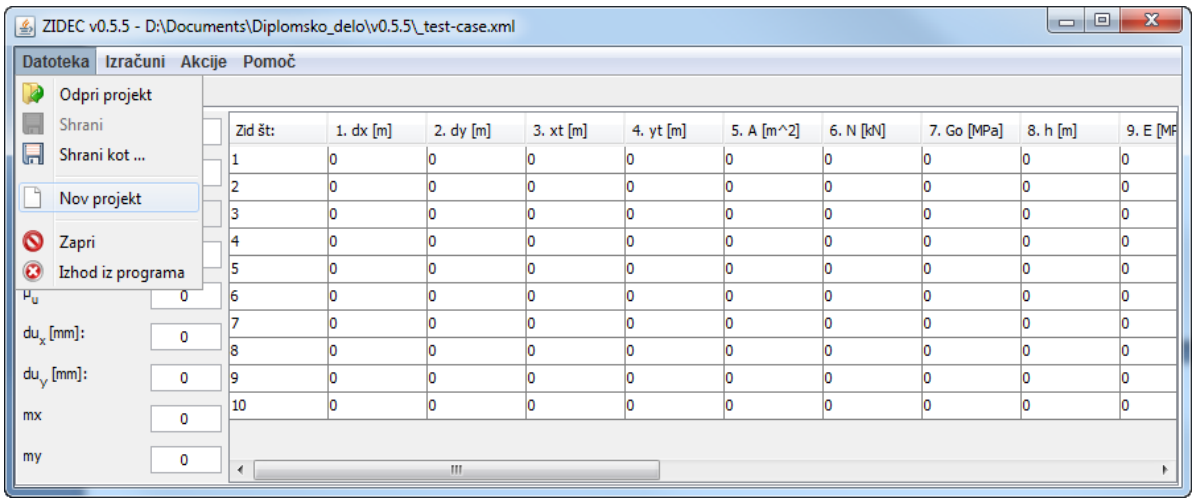
6. V paketu **icons** so shranjene slike, ki so uporabljene kot ikone v menijih aplikacije.



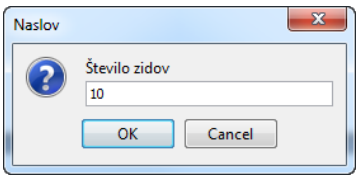
Slika 18. Drevesna struktura razporeditve programske kode.

4.4. Uporaba aplikacije ZIDEC

Ko zaženemo aplikacijo ZIDEC, ustvarimo nov projekt (Slika 19 in Slika 20) in začnemo vpisovati podatke za izračun. V stolpec na levi strani je potrebno vpisati konstante, ki veljajo za celo etažo. V tabelo na desni strani pa je potrebno vpisati karakteristike za vsak posamezen zid v etaži (dolžina, širina, višina, teža, kakovost materiala, itd.). Število zidov je omejeno na 1000. Ko vpišemo vse potrebne podatke za izračun in preko menija *Izračuni* izberemo *Izračunaj*, se pojavijo štirje glavni zavihki (Slika 21): Podatki, Skica, Izračuni in Rezultati.



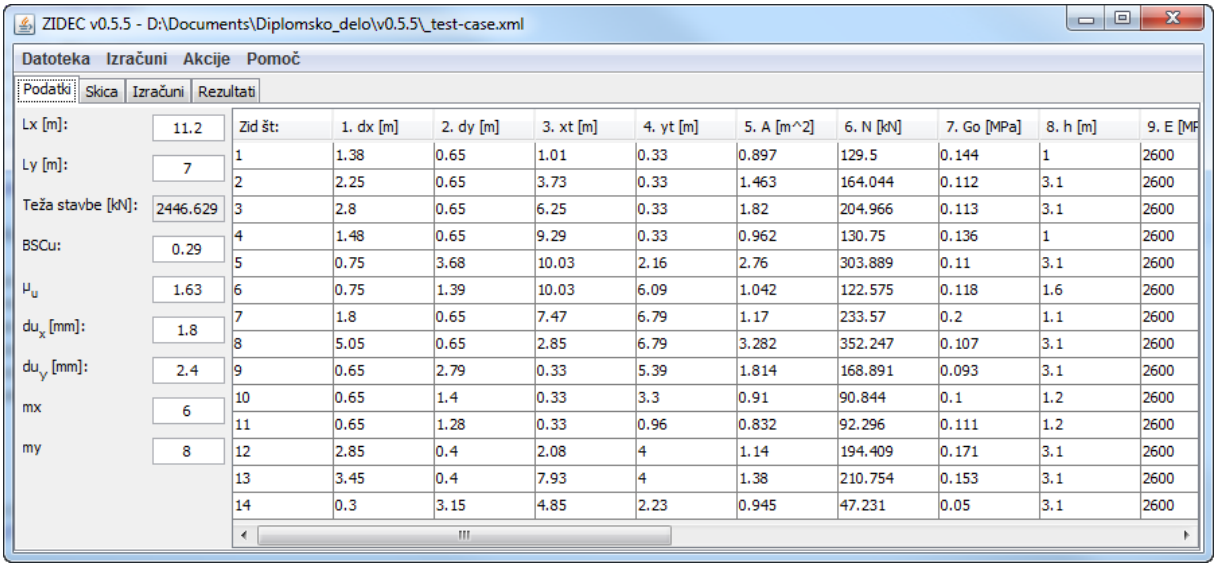
Slika 19. Ustvarjanje novega projekta.



Slika 20. Vnosni obrazec za število zidov.

4.4.1. Podatki

Pod zavihkom *Podatki* se nahajajo vsa vnosna polja, ki od uporabnika zahtevajo vnos podatkov. Na levi strani je stolpec s konstantami, na desni pa tabela zidov (Slika 21). Vsaka vrstica predstavlja en zid, skupaj pa tvorijo eno etažo.

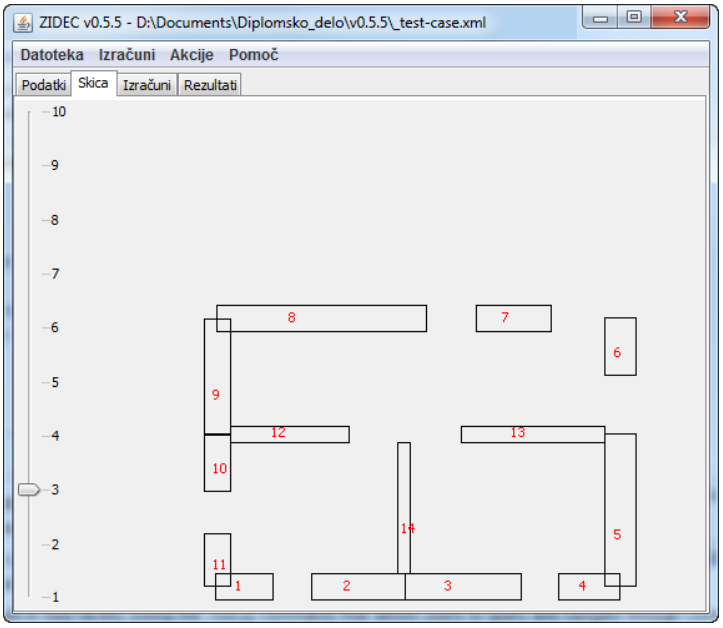


Zid št:	1. dx [m]	2. dy [m]	3. xt [m]	4. yt [m]	5. A [m^2]	6. N [kN]	7. Go [MPa]	8. h [m]	9. E [MP
1	1.38	0.65	1.01	0.33	0.897	129.5	0.144	1	2600
2	2.25	0.65	3.73	0.33	1.463	164.044	0.112	3.1	2600
3	2.8	0.65	6.25	0.33	1.82	204.966	0.113	3.1	2600
4	1.48	0.65	9.29	0.33	0.962	130.75	0.136	1	2600
5	0.75	3.68	10.03	2.16	2.76	303.889	0.11	3.1	2600
6	0.75	1.39	10.03	6.09	1.042	122.575	0.118	1.6	2600
7	1.8	0.65	7.47	6.79	1.17	233.57	0.2	1.1	2600
8	5.05	0.65	2.85	6.79	3.282	352.247	0.107	3.1	2600
9	0.65	2.79	0.33	5.39	1.814	168.891	0.093	3.1	2600
10	0.65	1.4	0.33	3.3	0.91	90.844	0.1	1.2	2600
11	0.65	1.28	0.33	0.96	0.832	92.296	0.111	1.2	2600
12	2.85	0.4	2.08	4	1.14	194.409	0.171	3.1	2600
13	3.45	0.4	7.93	4	1.38	210.754	0.153	3.1	2600
14	0.3	3.15	4.85	2.23	0.945	47.231	0.05	3.1	2600

Slika 21. Izpolnjeni podatki za etažo s štirinajstimi zidovi.

4.4.2. Skica

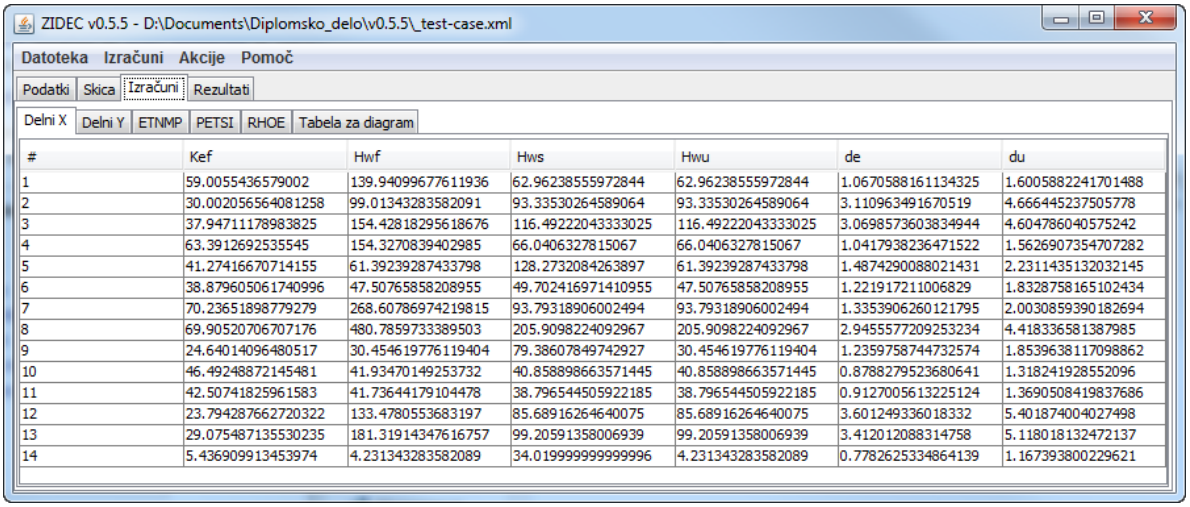
V zavihku *Skica* se uporabniku iz vnesenih podatkov izriše skica tlorisa etaže (Slika 22). Služi predvsem za preverjanje pravilnosti vnosa lege zidov. Na levi strani je drsnik, ki omogoča povečevanje in pomanjševanje izrisa.



Slika 22. Prikaz tlorisa etaže.

4.4.3. Izračuni

Zavihek *Izračuni* vsebuje pet podzavihkov (Slika 23). Vsak podzavihek vsebuje tabelo vmesnih izračunov, ki aplikaciji služijo za začasno shranitev vmesnih izračunov in kasnejšemu izrisu grafa, uporabnik pa lahko preveri pravilen potek računanja. Povzetek glavnih enačb in formul, po katerih se računa, je prikazan v dodatku B.

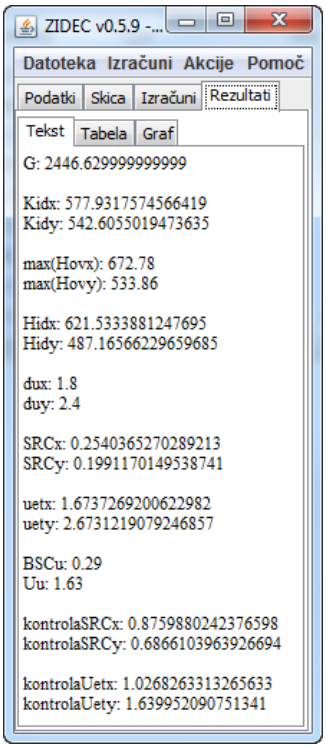


#	Kef	Hwf	Hws	Hwu	de	du
1	59.0055436579002	139.94099677611936	62.96238555972844	62.96238555972844	1.0670588161134325	1.6005882241701488
2	30.002056564081258	99.01343283582091	93.33530264589064	93.33530264589064	3.110963491670519	4.666445237505778
3	37.94711178983825	154.42818295618676	116.49222043333025	116.49222043333025	3.0698573603834944	4.604786040575242
4	63.3912692535545	154.3270839402985	66.0406327815067	66.0406327815067	1.0417938236471522	1.5626907354707282
5	41.27416670714155	61.39239287433798	128.2732084263897	61.39239287433798	1.4874290088021431	2.2311435132032145
6	38.879605061740996	47.50765858208955	49.702416971410955	47.50765858208955	1.221917211006829	1.8328758165102434
7	70.23651898779279	268.60786974219815	93.79318906002494	93.79318906002494	1.3353906260121795	2.0030859390182694
8	69.90520706707176	480.7859733389503	205.9098224092967	205.9098224092967	2.9455577209253234	4.418336581387985
9	24.64014096480517	30.454619776119404	79.38607849742927	30.454619776119404	1.2359758744732574	1.8539638117098862
10	46.49248872145481	41.93470149253732	40.858898663571445	40.858898663571445	0.8788279523680641	1.318241928552096
11	42.50741825961583	41.73644179104478	38.796544505922185	38.796544505922185	0.9127005613225124	1.3690508419837686
12	23.794287662720322	133.4780553683197	85.68916264640075	85.68916264640075	3.601249336018332	5.401874004027498
13	29.075487135530235	181.31914347616757	99.20591358006939	99.20591358006939	3.412012088314758	5.118018132472137
14	5.436909913453974	4.231343283582089	34.019999999999996	4.231343283582089	0.7782625334864139	1.167393800229621

Slika 23. Prikaz tabel z delnimi izračuni.

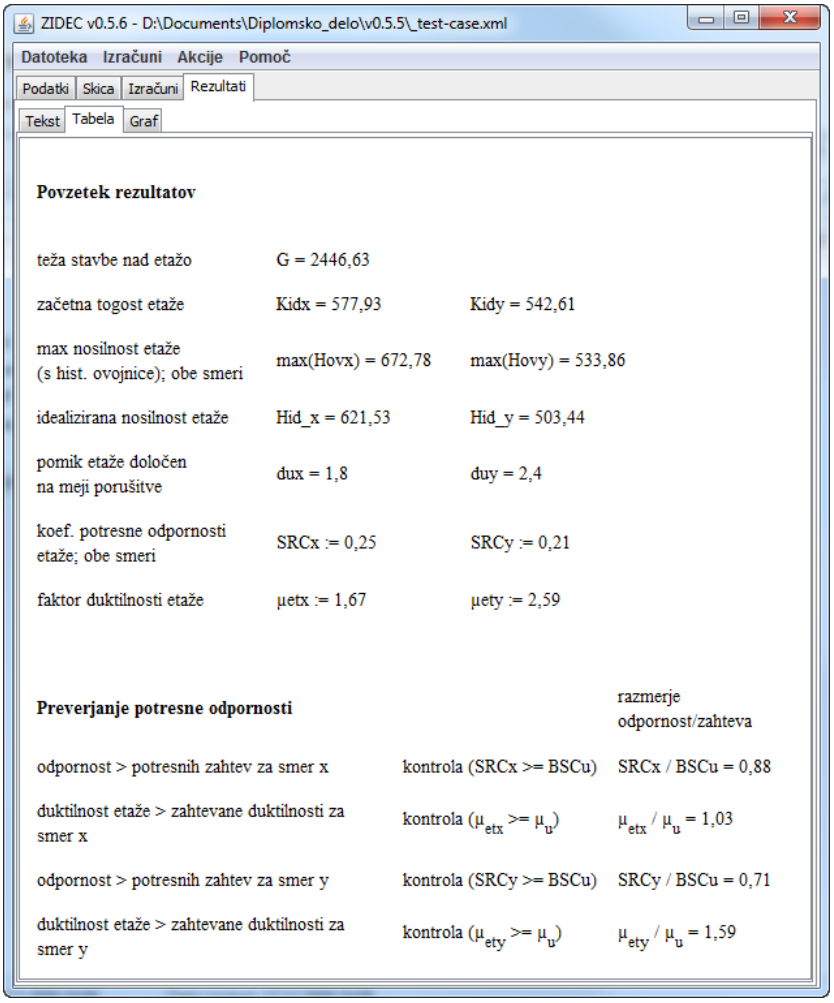
4.4.4. Rezultati

Zavihek *Rezultati* vsebuje tri podzavihke. Vsak od njih prikazuje končne izračune v svoji obliki. V zavihku *Tekst* so končni izračuni podani v obliki neurejenega izpisa (Slika 24).



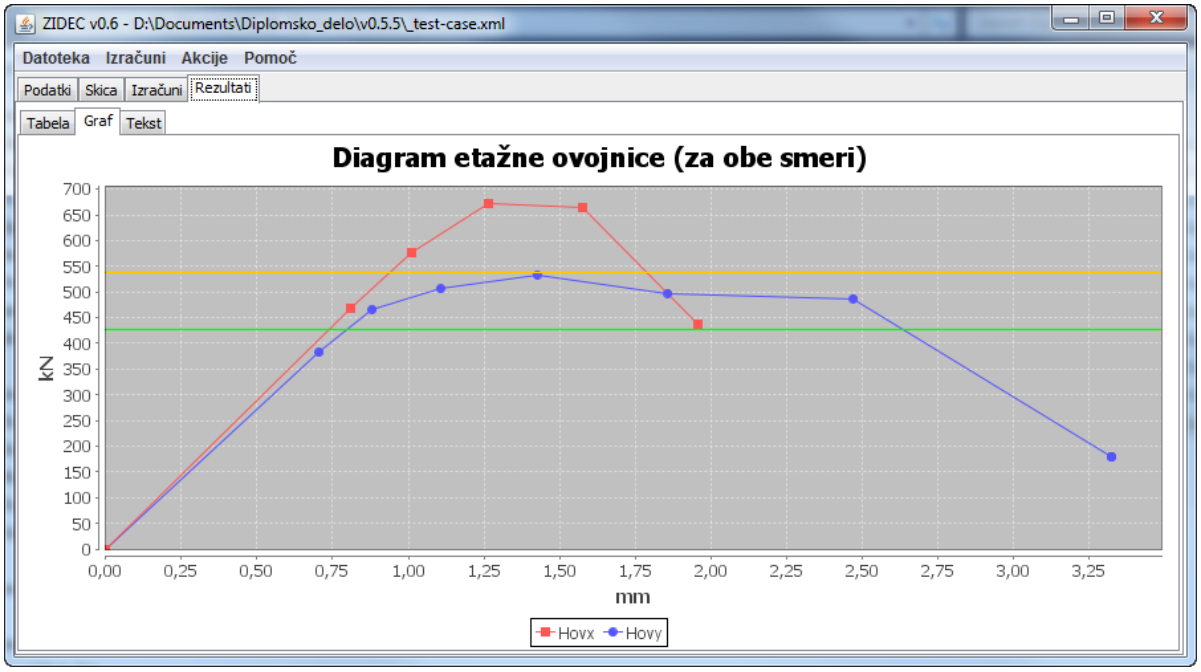
Slika 24. Končni izračuni v neurejenem izpisu.

V zavihku *Tabela* je besedilo končnih rezultatov oblikovano v tabeli in primerno za uvoz v npr. Microsoft Word (Slika 25).



Slika 25. Končni izračuni prikazani v obliki tabele.

V tretjem zavihku *Graf* so končni izračuni prikazani grafično v obliki grafa (Slika 26).



Slika 26. Rezultati izračunov v grafični obliki.

Diagram za obe ortogonalni smeri prikazuje odvisnost nosilnosti etaže v odvisnosti od pomikov etaže. Rumena horizontalna črta predstavlja 80% maksimalne vrednosti sile etažne ovojnice v smeri x, zelena pa 80% maksimalne vrednosti sile etažne ovojnice v smeri y. Ko grafa funkcij Hovx in Hovy padeta pod 80% svoje max. vrednosti, se iteracija računanja konča. Za končne rezultate (koeficient potresne odpornosti etaže - SRC in faktor duktilnosti etaže - μ_{et}) je potrebno izračunati ploščino, ki je omejena z funkcijo Hov ter 80% njene max. vrednosti.

5. Sklepne ugotovitve

Pri načrtovanju in izdelavi aplikacije ZIDEC sem ugotovil, da je za izdelavo večje in kompleksnejše aplikacije potreben celovit in načrtovan pristop pri postavitvi razvojnega okolja in izbiri tehnologij. Pri izdelavi aplikacije sem si pomagal z vsemi opisanimi orodji in pristopi. Do uporabe večine orodij sem prišel po načelu "pojavi se problem, išče se rešitev". Pred pričetkom izdelave aplikacije nisem bil seznanjen z določenimi orodji ali pa sem jih uporabljal v drugačne namene npr.:

- VirtualBox sem uporabljal za testiranje operacijskih sistemov ter testiranje različnih aplikacij na njih, nikoli pa za postavitev razvojnega okolja za izdelavo lastnih aplikacij.

- Rešitev za revizijo izvorne kode v obliki SVN sem začel iskati po tem, ko sem izgubil nekaj že napisane kode in takrat, ko sem opazil, da je bila funkcija napisana pred nekaj dnevi ustreznejša.

- Sistem Trac sem začel uporabljati takrat, ko sem videl, da potrebujem pregleden sistem za:
  - beleženje opravljenega dela,
  - dodajanje beležk,
  - kaj je še potrebno narediti in
  - vpisovanje napak, ki sem jih opazil med testiranjem.

Večino drugih orodij sem spoznal in uporabljal že med študijem.

Poudarek je bil na prosto dostopni programski opremi, saj menim, da so lahko tovrstne rešitve enakovredne plačljivim programom. Takšen pristop izdelave aplikacije je cenovno ugoden, saj ni potrebno kupiti licenc.

Programska rešitev ZIDEC je uspešno izvedena in se uporablja v podjetju Karlovšek d.o.o. Do sedaj je bila uporabljena pri izračunih protipotresnih zahtev šestih objektov. Pred tem sem pravilno delovanje aplikacije testiral na že izračunanih primerih, ki so mi jih posredovali iz gradbenega inštituta ZRMK.

Izboljšano in popravljeno kodo odprtokodne aplikacije Java Log Viewer sem posredoval avtorju.

Z nadaljnjim razvojem bi lahko aplikacijo izboljšal:

- S funkcijama razveljavi (ang. undo) in uveljavi (ang. redo).
  Med testiranjem aplikacije in v praksi se je izkazalo, da se včasih pomotoma vpišejo oz. prepišejo stare vrednosti v vnosnih poljih in bi uporabnik želel povrniti prejšnjo vrednost.

- Z naknadno dodajanje in odstranjevanje zidov.
  Ko ustvarimo nov projekt, aplikacija ne omogoča nadaljnjega dodajanja ali odstranjevanja zidov. Trenutno je potrebno datoteko XML, v kateri so zapisane vrednosti, "ročno" urediti v tekstovnem urejevalniku.

- S samodejnim shranjevanjem vnesenih podatkov glede na podani časovni interval.
  Funkcionalnost bi omogočala, da uporabnik v nastavitvah določi časovni interval npr. 5 minut, v katerem bi se vse nastavitve in vnešeni podatki samodejno shranjevali.

Največ težav sem imel z dokumentom *Operativno navodilo Državne tehnične komisije*, ki je služil kot osnova za izdelavo aplikacije, saj je vseboval dva nepopolna primera izračunov, za katera sem sprva mislil, da gre za en sam primer. Slaba dokumentacija, nenatančno določeni algoritmi in izračuni, ter kompleksnost samega problema so bili glavni razlog zamudnega in počasnega razvoja. Časovno najbolj zahtevno je bilo poglabljanje v tematiko s področja gradbeništva in iskanje pravilnega poteka izračunov.

Med postavitvijo razvojnega okolja nisem naletel na večje težave, ki se je jih ne bi dalo primerno hitro rešiti z iskanjem po spletu. Največji izziv je bilo povezati vse programe in tehnologijo v delujočo celoto. V veliko pomoč mi je bilo znanje o operacijskem sistemu GNU/Linux, ki sem ga samoiniciativno pridobil med študijem ter programerske izkušnje in navade, ki sem jih pridobil na opravljanju obvezne študijske prakse v podjetju Marg d.o.o.

6. Dodatek A

Seznam slik

7. Dodatek B

Povzetek glavnih enačb in formul za izračun potresne odpornosti zidanih zgradb [3].

Izračun mehanskih karakteristik (za i-ti zid)

površina prereza zidu $Aw_i = dx \cdot dy$

napetost zaradi vertikalne obtežitve $A = \frac{\sum V}{Aw_i}$

račun karakteristik za smer X

efektivna togost zidu $K_{ef\_x_i} = \frac{G_W \cdot Aw_i}{1{,}2 \cdot h\left[1 + \frac{G_W}{c * E_W} \cdot \left(\frac{h}{dx}\right)^2\right]}$

upogibna nosilnost $H_{wf} = \frac{\sigma_{o_i} \cdot dy \cdot dx^2}{2 \cdot \alpha \cdot h} * \left(1 - \frac{\sigma_{o_i}}{f_{wc}}\right)$

strižna nosilnost $H_{ws} = C_R \cdot A_{w_i} \cdot \frac{f_{wt}}{b(dx)} \cdot \sqrt{1 + \frac{\sigma_{o_i}}{f_{wt}}}$

mejna nosilnost i-tega zidu $H_{wu\_x_i} = \min(H_{wf}, H_{ws})$

pomik na meji elastičnosti $d_{ex_i} = \frac{H_{wu\_x_i}}{K_{ef\_x_i}}$

mejni pomik $d_{ux_i} = \mu \cdot d_{ex_i}$

za smer y zamenjamo indekse y ⇔ x in x ⇔ y

Izračun masnega središča

teža stavbe nad etažo $G = \sum_i (\sigma_{o_i} \cdot A_{w_i})$

koordinatne težišča $x_m = \frac{\sum_i (\sigma_{o_i} \cdot A_{w_i} \cdot x_i)}{G}$

$y_m = \frac{\sum_i (\sigma_{o_i} \cdot A_{w_i} \cdot y_i)}{G}$

Izračun strižnega središča (obe smeri)

prva iteracija začetne togosti $K_x = K_{ef\_x}$ $K_y = K_{ef\_y}$

togost etaže je vsota togosti zidov za posamezno smer $K_{etx} = \sum_i K_{x_i}$ $K_{ety} = \sum_i K_{y_i}$

koordinate strižnega središča $x_s = \frac{\sum_i \left(K_{y_i} \cdot x_i\right)}{K_{ety}}$ $y_s = \frac{\sum_i \left(K_{x_i} \cdot y_i\right)}{K_{etx}}$

ekscentričnost $e_x = x_m - x_s$ $e_y = y_m - y_s$

slučajna ekscentričnost $e_{ax} = 0{,}05 \cdot L_x \cdot \frac{e_x}{|e_x|}$

$e_{ay} = 0{,}05 \cdot L_y \cdot \frac{e_y}{|e_y|}$

povečanje zaradi vpliva slučajne ekscentričnosti $e_{ax} = x_m - x_s + e_{ax}$

$e_{ay} = y_m - y_s + e_{ay}$

torzijska vztrajnost tlorisa $l_x = \sum_i \left(K_{x_i} \cdot {y_i}^2\right) - {y_s}^2 \cdot K_{etx}$

$l_y = \sum_i \left(K_{y_i} \cdot {y_i}^2\right) - {x_s}^2 \cdot K_{ety}$

$l_t = l_x = l_y$

faktor povečanja pomikov posameznih zidov zaradi torzije

smer x $\rho_{x_i} = 1 + e_y \cdot \frac{K_{etx}}{l_t} \cdot (y_i - y_s)$

smer y $\rho_{y_i} = 1 + e_x \cdot \frac{K_{ety}}{l_t} \cdot (x_i - x_s)$

Začetna iteracija (meja elastičnosti etaže)

pomik zidov na meji elastičnosti $d_{el\_x_i} = \frac{d_{ex_i}}{\rho_{x_i}}$ $d_{el\_y_i} = \frac{d_{ey_i}}{\rho_{y_i}}$

računamo za vsak zid

pomik etaže na meji elastičnosti $d_{et\_x} = \min(d_{el\_x})$ $d_{et\_y} = \min(d_{el\_y})$

pomik zidov glede na pomik etaže pri mejni elastičnosti $d_{tx_i} = d_{et\_x} \cdot \rho_{x_i}$ $d_{ty_i} = d_{et\_y} \cdot \rho_{y_i}$

dokler je zid v elastičnem območju, to je: $d_{tx_i} < d_{ex_i}$ je trenutna togost K_{ef}, drugače H_{wu}/d_t

$$K_{tx\_i} = if\left(d_{tx_i} < d_{ty_i}, K_{ef\_x_i}, \frac{H_{wu\_x_i}}{d_{tx_i}}\right)$$

dokler je zid v elastičnem območju, to je: $d_{tx_i} < d_{ty_i}$ je trenutna slika v zidu $K_{ef} * d_t$, drugače H_{wu}

$$H_{tx_i} = if(d_{tx\,i} < d_{ex\,i}, K_{ef_{x_i}} \cdot d_{tx_i}, H_{wu}\_x_i)$$

ko je duktilnost zidu prekoračena $d_{txi} > d_{uxi}$, je togost in nosilnost zidu enaka 0, sicer ostane vrednost kot je izračunana zgoraj

$$K_{x_i} = if(d_{tx_i} > d_{ux_i}, 0\ \frac{kN}{m}, K_{tx\,i})$$

$$H_{x_i} = if(d_{tx_i} > d_{ux_i}, 0\ \frac{kN}{m}, H_{tx\,i})$$

za smer y zamenjamo indekse y ⇔ x in x ⇔ y

trenutna etažna sila je vsota trenutnih nosilnosti zidov

$$H_{etx} = \sum_i H_{x_i} \qquad H_{ety} = \sum_i H_{y_i}$$

trenutni etažni pomik

$$d_{etm\_x} = d_{et\_x} \cdot \left[1 + e_y \cdot \frac{K_{etx}}{l_t} \cdot (y_m - y_s)\right]$$

$$d_{etm\_y} = d_{et\_y} \cdot \left[1 + e_x \cdot \frac{K_{ety}}{l_t} \cdot (x_m - x_s)\right]$$

ploščine pod histerezno krivuljo etaže (smer x) od pomika d_{ux}

$$A_{hx} = \sum_{i=2}^{mx} \frac{1}{2} \cdot \left(H_{ovx_i} + H_{ovx_{i-1}}\right) \cdot if[d_{ovx_i} < d_{ux}\ , \left(d_{ovx_i} - d_{ovx_{i-1}}\right), (d_{ux} - d_{ovx_{i-1}})]$$

začetna togost etaže

$$K_{idx} = \frac{H_{ovx\,2}}{d_{ovx\,2}}$$

idealizirana nosilnost etaže

$$H_{id\_x} = K_{idx} \cdot (d_{ux} - \sqrt{{d_{ux}}^2 - \frac{2 \cdot A_{hx}}{K_{idx}}})$$

za smer y zamenjamo indekse y ⇔ x in x ⇔ y

koeficient potresne odpornosti (ang. Share Resistance Coeficient)

$$SRC_x = \frac{H_{id\_x}}{G} \qquad SRC_y = \frac{H_{id\_y}}{G}$$

faktor duktilnosti etaže

$$\mu_{etx} = \frac{d_{ux} \cdot K_{idx}}{H_{id\_x}} \qquad \mu_{ety} = \frac{d_{uy} \cdot K_{idy}}{H_{id\_y}}$$

8. Literatura

[1] A. Tanenbaum, *Distributed systems principles and paradigms*, 2002, pogl. 1.5.

[2] M. Podgoršek, *N-NIVOJSKA ARHITEKTURA V OGRODJU .NET VERZIJE 3.5*, 2008, pogl. 3.2., str. 22-23.

[3] R. S. M. z. o. i. prostor, *Operativno navodilo Državne tehnične pisarne za izdelavo seizmične analize konstrukcij v okviru popotresne obnove objektov v Posočju*. 2005.

[4] (2009) Uradni list Republike Slovenije. Dostopno na: http://www.uradni-list.si/1/content?id=58716

[5] (2009) Ubuntu Home Page. Dostopno na: http://www.ubuntu.com/

[6] (2009) The Trac Project. Dostopno na: http://trac.edgewall.org/

[7] (2009) Subversion. Dostopno na: http://subversion.tigris.org/

[8] (2009) Subclipse. Dostopno na: http://subclipse.tigris.org/

[9] (2009) Por 2000 - software for antiseismic consolidation of masonry buildings. Dostopno na: http://www.newsoft-eng.it/Por2000_eng.htm

[10] (2009) Notepad++. Dostopno na: http://notepad-plus.sourceforge.net/uk/site.htm

[11] (2009) JFreeChart. Dostopno na: http://www.jfree.org/jfreechart/

[12] (2009) Java Log Viewer. Dostopno na: http://sourceforge.net/projects/jlogviewer/

[13] (2009) FEDRA Masonry design according to Eurocode 6. Dostopno na: http://www.runet-software.com/FEDRA.htm

[14] (2009) DistroWatch.com: Put the fun back into computing. Use Linux, BSD.. Dostopno na: http://distrowatch.com/

[15] (2009) Virtualbox. Dostopno na: http://www.virtualbox.org/

[16] (2009) The Apache HTTP Server Project. Dostopno na: http://httpd.apache.org/

[17] (2009) About the Java Technology. Dostopno na: http://java.sun.com/docs/books/-tutorial/getStarted/intro/definition.html

[18] (2009) About Java Technology. Dostopno na: http://www.sun.com/java/about/

[19] (2009) Eclipse.org home. Dostopno na: http://www.eclipse.org/

[20] (2009) Web Server Survey Archives - Netcraft. Dostopno na: http://news.netcraft.com/-archives/web_server_survey.html

[21] (2009) The State of Munich's Ongoing Linux Migration. Dostopno na: http://news.-slashdot.org/story/09/06/28/0344234/The-State-of-Munichs-Ongoing-Linux-Migration

[22] (2009) Članki - Brezplačno je vseeno predrago?. Dostopno na: http://www.monitor.si/-clanek/brezplacno-je-vseeno-predrago/

[23] (2009) XML. Dostopno na: http://en.wikipedia.org/wiki/Xml