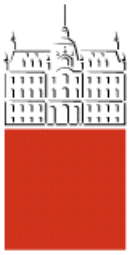
Univerza
v Ljubljani
Fakulteta
za gradbeništvo
in geodezijo



Jamova cesta 2
1000 Ljubljana, Slovenija
http://www3.fgg.uni-lj.si/

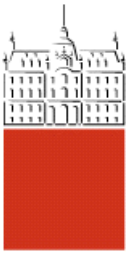
DRUGG – Digitalni repozitorij UL FGG
http://drugg.fgg.uni-lj.si/

To je izvirna različica zaključnega dela.

Prosimo, da se pri navajanju sklicujte na bibliografske podatke, kot je navedeno:

Hren, M. 2013. Spletni vmesnik za izvajanje računskih analiz v visoko-prepustnem računskem okolju. Diplomska naloga. Ljubljana, Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo. (mentor Dolenc, M., somentor Dolšek, M.): 55 str.

University
of Ljubljana
Faculty of
Civil and Geodetic
Engineering



Jamova cesta 2
SI – 1000 Ljubljana, Slovenia
http://www3.fgg.uni-lj.si/en/

DRUGG – The Digital Repository
http://drugg.fgg.uni-lj.si/

This is original version of final thesis.

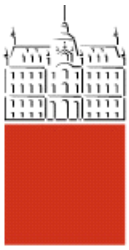
When citing, please refer to the publisher's bibliographic information as follows:

Hren, M. 2013. Spletni vmesnik za izvajanje računskih analiz v visoko-prepustnem računskem okolju. B.Sc. Thesis. Ljubljana, University of Ljubljana, Faculty of civil and geodetic engineering. (supervisor Dolenc, M., co-supervisor Dolšek, M.): 55 pp.

Univerza
v Ljubljani

Fakulteta za
gradbeništvo in
geodezijo

Jamova 2
1000 Ljubljana, Slovenija
telefon (01) 47 68 500
faks (01) 42 50 681
fgg@fgg.uni-lj.si



UNIVERZITETNI ŠTUDIJ
GRADBENIŠTVA
ORGANIZACIJSKO
TEHNOLOŠKA SMER

Kandidat:

MIHA HREN

SPLETNI VMESNIK ZA IZVAJANJE RAČUNSKIH ANALIZ V VISOKO-PREPUSTNEM RAČUNSKEM OKOLJU

Diplomska naloga št.: 3286/OTS

WEB INTERFACE FOR EXECUTION OF COMPUTATIONAL TASKS IN A HIGH - THROUGHPUT COMPUTING ENVIRONMENT

Graduation thesis No.: 3286/OTS

Mentor:
doc. dr. Matevž Dolenc

Somentor:
izr. prof. dr. Matjaž Dolšek

Član komisije:
prof. dr. Igor Planinc
doc. dr. Primož Banovec
asist. dr. Tomaž Hozjan

Predsednik komisije:
izr. prof. dr. Janko Logar

Ljubljana, 22. 02. 2013

STRAN ZA POPRAVKE

Stran z napako	Vrstica z napako	Namesto	Naj bo

IZJAVA

Podpisani Miha Hren izjavljam, da sem avtor diplomske naloge z naslovom "Spletni vmesnik za izvajanje računskih analiz v visoko-prepustnem računskem okolju".

Izjavljam, da je elektronska različica v vsem enaka tiskani različici.

Izjavljam, da dovoljujem objavo elektronske različice v repozitoriju UL FGG.

Ljubljana, 11. 2. 2013 Miha Hren

BIBLIOGRAFSKO-DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK:	**004:519.6:624(043.2)**
Avtor:	**Miha Hren**
Mentor:	**doc. dr. Matevž Dolenc**
Naslov:	**Spletni vmesnik za izvajanje računskih analiz v visoko-prepustnem računskem okolju**
Tip dokumenta:	**diplomska naloga – univerzitetni študij**
Obseg in oprema:	**48 str., 1 pregl., 36 sl.**
Ključne besede:	**splet, vmesnik, AJAX, HTML, PHP, MySQL, JavaScript, HTCondor, git, baza podatkov, izvorna koda**

Izvleček

Diplomska naloga je osredotočena na izdelavo in vzroke za stvaritev spletnega vmesnika za izvajanje računskih analiz v visoko-prepustnem računskem okolju, ki je vzpostavljeno na UL FGG in deluje na sistemu HTCondor.

Delo je, poleg uvodnega in zaključnega poglavja, razdeljeno na tri večje vsebinske sklope. Prvi sklop predstavlja teoretične osnove, kjer so opisani vsi pripomočki, programski jeziki in orodja, ki so bila uporabljena pri izdelavi tega vmesnika. Predstavljene so tudi razlike med različnimi računskimi okolji.

Drugi del naloge je posvečen sistemu HTCondor, okoli katerega je zgrajen spletni vmesnik. Poleg splošnega opisa tehnologije in njene uporabe je tekom poglavja opisan tudi problem privzete uporabe tega visoko-prepustnega okolja. Ta uporabniku neprijazna uporaba namreč predstavlja tudi glavni vzrok za potrebo po takem vmesniku. Sledi še opis konkretne implementacije sistema HTCondor, ki je vzpostavljen na Fakulteti za gradbeništvo in geodezijo v Ljubljani.

Zadnji del, pred zaključkom in sklepi, pa je opis implementacije, razvoja in uporabniške izkušnje izdelanega spletnega vmesnika. V tem poglavju so spisane vse zahteve, ki so diktirale potrebne elemente za nastanek vmesnika. Na kratko je opisan tudi načrt, kjer so vsi ti elementi povezani v zaključeno celoto. Večji del tega poglavja pa predstavlja opis končne uporabniške izkušnje in konkretno implementacijo, ki pa zavoljo manjše pomembnosti vseeno ne gre v prevelike podrobnosti.

BIBLIOGRAPHIC-DOCUMENTALISTIC INFORMATION AND ABSTRACT

UDC:	**004:519.6:624(043.2)**
Author:	**Miha Hren**
Supervisor:	**Assist. Prof. Matevž Dolenc, Ph.D.**
Title:	**Web interface for execution of computational tasks in a high-throughput computing environment**
Document type:	**Graduation Thesis – University studies**
Scope and Tools:	**48 p., 1 tab., 36 fig.**
Keywords:	**web, interface, AJAX, HTML, PHP, MySQL, JavaScript, HTCondor, git, database, source code**

Abstract

This Graduation Thesis is centred on the making of a web interface, to be used for communicating with and manipulating a high-throughput computing environment HTCondor, set up at UL FGG.

Besides the Introduction and the Conclusion sections, the thesis consists of three main parts. The first part describes the theoretical foundation on which this web interface was built. All the tools, program languages, techniques and technologies that were used in the making of this interface are described here.

The second section is dedicated to the HTCondor system, which represents the basis for this interface. The section includes a general description of the technology, its usage and a description of the problematic default user interface which comes with HTCondor. The very unfriendly nature of the UI is the main reason why this new interface needed to be made. On top of this, the section includes a rough description of the current implementation of the HTCondor system at the Faculty of Civil and Geodetic Engineering.

The last of the main three parts consists of descriptions about the implementation, research and the user experience related to the new user interface. All the requirements, which have dictated the elements of the user interface, are described here. There is also a short description about how said elements are tied into a whole. The larger part of this section is dedicated to describing the end user experience and its implementation. Due to fewer relevancies, the latter doesn't go into too much detail.

ZAHVALA

Za pomoč in nasvete pri izdelavi diplomske naloge se iskreno zahvaljujem mentorju doc. dr. Matevžu Dolencu.

Posebna zahvala gre tudi mojim staršem in sestri, ki so mi omogočili študij ter mi pomagali z napotki iz študijskih časov in življenja na splošno.

Nazadnje bi se rad zahvalil še sošolcem, s katerimi smo preživeli štiri lepa leta, skozi katera smo se z enako zagretostjo zabavali, kot se učili, delali in obiskovali predavanja.

KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK

OKRAJŠAVE IN SIMBOLI

PHP	angl. PHP: Hypertext Preprocessor
HTML	angl. HyperText Markup Language
AJAX	angl. Asynchronous JavaScript And XML
XML	angl. Extensible Markup Language
SQL	angl. Structured Query Language
HTC	angl. High Throughput Computing
CSS	angl. Cascading Style Sheets
HTTP	angl. HyperText Transfer Protocol
DOM	angl. Document Object Model
GNU	angl. GNU's Not Unix
GPL	angl. General Public License
OOP	angl. Object Oriented Programming
IDA	angl. Incremental Dynamic Analysis

Ta stran je namenoma prazna

1 UVOD

V gradbeništvu je uporaba informacijskih tehnologij vedno večja, pogosto pa tudi nujna. Še najbolj je to opazno v inštitutih in na univerzah, kjer stalno potekajo raziskave in iskanje novih rešitev, za kar pa potrebujejo pomoč prav prej omenjenih informacijskih tehnologij. Ena takih so visoko-prepustna računska okolja, ki omogočajo sodelovanje večjega števila naprav pri izračunu nekega kompleksnega problema.

Žal pa uporaba visoko-prepustnih računskih okolij ni trivialna. Sistem HTCondor, ki je vzpostavljen na Fakulteti za gradbeništvo in geodezijo, se namreč upravlja preko konzolnega okna. Ker je ta neizkušenim uporabnikom neprijazen in nepregleden za uporabo, je smiselno izdelati uporabniški vmesnik, čemur je to diplomsko delo posvečeno.

1.1 Motivacija

Primarna motivacija za izdelavo tega je projekta je moje lastno navdušenje nad informacijskimi tehnologijami in programiranjem. S slednjim sem bil seznanjen že v gimnazijskih časih in precej hitro ugotovil, da je, poleg gradbeništva, tudi to smer, v kateri bi se rad več izobrazil. Ker je v gradbeništvu potreb po informacijskih tehnologijah precej, je bila diplomska naloga idealna priložnost, da združim svoj konjiček s strokovno izobrazbo, ki sem jo pridobil tekom študija na fakulteti.

Pred začetkom izdelovanja naloge sicer nisem imel skoraj nič znanja in izkušenj s spletnim programiranjem, zgolj s klasičnim izdelovanjem aplikacij. Posledično motivacija ni izvirala samo iz samega veselja do programiranja, temveč tudi iz želje po učenju in spoznavanju novih tehnologij, postopkov in programskih jezikov, o katerih sem do tedaj le prebiral poljudne članke na spletu.

1.2 Cilji

Končni cilj naloge je izdelava spletnega vmesnika za izvajanje računskih analiz v visoko-prepustnem računskem okolju, ki je vzpostavljeno na UL FGG in deluje na sistemu HTCondor. Ker gre za spletni vmesnik, bo ta ves čas lociran na šolskih strežnikih in dostopen preko spletnih brskalnikov. Želja po vzpostavitvi takega sistema izvira predvsem iz zahteve po konstantni aktivnosti sodelujočih naprav in uporabniku neprijaznega privzetega vmesnika, ki deluje preko terminalnega okna. Ker je za njegovo navigacijo potrebnega kar nekaj specifičnega znanja, je precej bolj smotrno izdelati spletni vmesnik, ki bo intuitiven in ga bodo znali uporabljati tudi manj izkušeni uporabniki.

Vmesnik je prvotno mišljen predvsem za reševanje problemov potresnih inženirjev na fakulteti, s katerimi je katedra za gradbeno informatiko tudi najtesneje povezana. Ker dinamične analize v račun vpeljejo dodatno spremenljivko časa, se precej poveča kompleksnost problemov, z njo pa tudi potreba po večji procesorski moči. Prav ta potreba po dodatni moči je sprožila željo po vzpostavitvi omrežja

HTCondor in posredno po njegovi lažji uporabi. Sicer se bo vmesnik lahko uporabljal tudi v bolj splošne namene in bo na voljo drugim katedram.

1.3 Pomen za uporabnike

Vmesnik bo uporabniku prinesel kar nekaj prednosti v primerjavi z uporabo terminala. Prva je ta, da bodo laže upravljali s sistemom HTCondor, saj bo ta nudil grafični vmesnik tako za predložitev novih izračunov kot tudi brisanje obstoječih. Vse to bo dosegljivo z navigiranjem intuitivnega vmesnika za brskanje datotek, predlagalo in brisalo pa se bo s pritiskom na gumb. Slednje je precej bolj priročno kot tipkanje ukazov v terminalna okna. Datoteke se bodo na izvajanje pošiljale iz strežnika, kar bo preneslo zahtevo po konstantni prižganosti sodelujočih naprav na strežnik. Poleg tega bo vmesnik sam skrbel za to, da bodo vnosi različnih uporabnikov pravilno označeni. S tem bo omogočeno sledenje, komu pripada kateri izračun, kar bi morali sicer uporabniki delati ročno.

Naslednja prednost vmesnika, ki bo prinesla dodano vrednost uporabnikom, je preglednost. Izpisi sistema HTCondor bodo prikazani v preglednicah, ki se bodo samodejno posodabljale v določenem intervalu. Uporabnikom tako ne bo treba vedno znova vpisovati enih in istih ukazov, če bodo želeli videti trenutno stanje njihovega izračuna.

In še tretja prednost, vmesnik bo vplival na učinkovitost dela posameznikov, predvsem če so neizkušeni v uporabi visoko-prepustnih računskih okolij. To je posledica prej omenjenih priročnosti, saj bosta lažja uporaba in bolj pregledno prikazovanje trenutnega stanja večini omogočila hitrejše delo.

1.4 Organizacija diplomske naloge

Diplomska naloga je razdeljena na pet smiselnih poglavij, ki so razdeljena na podpoglavja. Poglavja so Uvod, Teoretične osnove, HTCondor, Uporabniški vmesnik in Zaključek.

Uvod je poglavje, kjer se trenutno nahajamo. Tukaj so opisani vsi cilji in vzroki, ki so pripeljali do nastanka tega projekta. Izražena je tudi moja lastna motivacija za izbiro teme diplomske naloge.

Naslednje in prvo obsežnejše poglavje ima naslov **Teoretične osnove**. Obsega opise različnih računskih okolij, programskih orodij, programskih jezikov in tehnik, ki so bile uporabljene pri izdelavi tega vmesnika. Kjer je smiselno, so podani tudi primeri ali kratka zgodovina nastanka.

V okviru poglavja **HTCondor** je predstavljeno omenjeno visoko-prepustno računsko okolje. Splošnemu opisu sledi še opis uporabe, izpostavljene prednosti in slabosti tehnologije ter implementacija sistema na Fakulteti za gradbeništvo in geodezijo, skupaj z zgodovino.

Četrto poglavje govori o **uporabniškem vmesniku** za HTCondor. Gre že za konkretno implementacijo, kjer je zapisan načrt programa in razložen njegov zgled ter uporabniška izkušnja.

Prav tako se tukaj nahaja razložena struktura vmesnika in različne možnosti njegove uporabe. Gre za najobširnejše poglavje, a vendar v okviru implementacije ne gre v prevelike podrobnosti.

Zadnje poglavje, imenovano **Zaključek**, zavzema sklep in ugotovitve, do katerih sem prišel tekom izdelave diplomske naloge.

2 TEORETIČNE OSNOVE

To poglavje predstavlja teoretične osnove, ki so bile potrebne za razumevanje, nastanek in izdelavo te diplomske naloge. V prvem delu sta opisana dva različna tipa računskih okolij, pri čemer so izpostavljene njune razlike. Eno od takih okolij je vzpostavljeno tudi na UL FGG, vendar pa je tema obsežnejša, zato ima v nadaljevanju svoje poglavje. Drugi sklop tega poglavja pa je namenjen opisu vseh programskih orodij, ki so bila uporabljena za izdelavo vmesnika. Posebna podpoglavja so namenjena tudi uporabljenim programskim jezikom in tehnikam programiranja.

2.1 Računska okolja

Računska okolja predstavljajo skupino računalnikov, procesorjev ali ostalih naprav, ki skupaj tvorijo gručo, katere namen je nudenje svoje računske moči za iskanje skupnega cilja [1]. V ta namen poznamo več tipov računskih okolij, pri čemer sta najpogostejši dve. To sta visoko-zmogljivo računsko okolje in visoko-prepustno računsko okolje. Prvo je bolj konvencionalne narave in ima manj skupnega z omrežjem (angl. grid) kot slednje, katerega podrobnosti bodo podrobneje obravnavane še v tretjem poglavju z naslovom HTCondor.

2.1.1 Visoko-zmogljiva računska okolja

Visoko-zmogljiva računska okolja so okolja, kjer je večje število homogenih naprav, po navadi procesorjev, povezanih v gručo računalnikov na lokalni ravni. To pomeni, da so praviloma naprave na isti geografski lokaciji in skupaj tvorijo tako imenovani superračunalnik.

Glavni namen takih okolij je opravljanje velikega števila izračunov v najkrajšem možnem času. Kot je zapisano na uradni strani HTCondor [2], je glavna mera sposobnosti takega sistema, koliko operacij lahko opravi na enoto sekunde. To se meri v enoti FLOPS (angl. FLoating-point Operations Per Second).

Takega okolja med izdelavo diplomske naloge nisem uspel pobliže spoznati in je manj relevantno za potrebe tega dela. Zanimivo je predvsem zavoljo primerjave z visoko-prepustnim računskim okoljem, ki bo opisano v naslednjem podpoglavju.

2.1.2 Visoko-prepustna računska okolja

V literaturi [1] je omrežje (angl. grid) definirano kot infrastruktura, ki integrira računanje in urejanje podatkov, medmrežje, komunikacijo in informacije ter tako nudi virtualno platformo za opravljanje izračunov in hranjenje podatkov. Ker je v omrežje povezanih veliko število naprav, ki se lahko nahajajo na različnih geografskih lokacijah, je za uspešno delovanje takega omrežja treba razviti kompleksne sisteme storitev in programske opreme. V okviru te diplomske naloge bodo zanimive

predvsem tehnologije, povezane z računanjem kompleksnih problemov, manj pa tiste za hranjenje podatkov. Ena takih tehnologij so visoko-prepustna računska okolja.

Za razliko od prej omenjenih visoko-zmogljivih okolij se visoko-prepustna okolja manj osredotočajo na reševanje problemov v najkrajšem možnem času, ampak dajejo prioriteto čim bolj učinkoviti izrabi dosegljivih virov v nekem daljšem časovnem obdobju. Tako se njihova sposobnost ne meri v izračunih na sekundo, temveč v številu opravljenih operacij na dan, mesec ali leto.

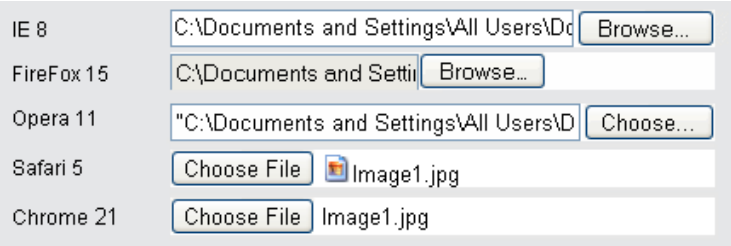
Glavna prednost takega sistema je izkoristek obstoječih računalniških naprav, ki se sicer uporabljajo za druge, bolj vsakdanje namene. Posledično se znižajo stroški, saj ustanovam ni treba kupovati dragih superračunalnikov, katerih namen je izključno reševanje zapletenih problemov v čim krajšem možnem času. Še ena pomembna lastnost takih okolij je, da se viri izrabljajo priložnostno. To pomeni, da ves čas ne bomo imeli na razpolago maksimalnega potenciala omrežja, saj se določeni viri lahko koristijo le, ko se ne uporabljajo za namene z višjo prioriteto. Po drugi strani pa s tem poskrbimo, da naprave nikoli niso v mirovanju, in če se le da, se jih koristi za izvajanje operacij.

2.2 Programski jeziki

2.2.1 HTML

HTML (angl. HyperText Markup Language) je označevalni jezik, s katerim prikazujemo spletne strani [3]. To počnemo s programi, imenovanimi spletni brskalniki, ki pa so pred standardizacijo HTML precej različno interpretirali in prikazovali vsebino HTML-gradiv. Ravno zaradi tega je bilo v interesu standardizirati jezik, kar se je z različico 2.0 leta 1995 tudi zgodilo. Različica 1.0 je bila sicer spisana že leta 1990, a je ostala nestandardizirana. Skozi leta je izšlo še nekaj različic, trenutno aktualna je 4.01, v pripravi pa je že HTML 5.0 standard. Ta bo prinesel kar nekaj potrebnih novosti, ki se nanašajo predvsem na večpredstavnost. Razvoj jezika od leta 1996 dalje vodi organizacija World Wide Web Consortium.

Uporaba jezika HTML je pri izdelavi spletnega vmesnika obvezna, saj gre za standardiziran jezik, ki ga znajo brati vsi današnji spletni brskalniki. Enakovrednih alternativ ni. Treba je le paziti na kompatibilnost z vsemi priljubljenimi brskalniki. Nekatere elemente standarda HTML namreč brskalniki še zmeraj različno interpretirajo. Eden takih je na primer element za nalaganje datotek (Slika 1)



Slika 1: Prikaz privzetega elementa za nalaganje datotek pri različnih brskalnikih

Sintaksa jezika HTML je med preprostejšimi in preglednejšimi. Gradijo ga namreč elementi HTML, ki so sestavljeni iz značk. Poznamo začetno (npr.) in končno značko (npr.), ki praviloma skupaj tvorita element HTML. Vsebina elementa se nahaja med obema značkama, medtem ko atribute posameznega elementa zapisujemo znotraj prve značke (npr. vsebina). Spodaj je primer preproste HTML izvorne datoteke (Slika 2). Datoteke s kodo HTML imajo praviloma končnico ".html" ali ".htm".

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
     <HEAD>
          <TITLE>Untitled</TITLE>
     </HEAD>
     <BODY>
          <P>paragraph</p>
     </BODY>
</HTML>
```

Slika 2: Primer preproste HTML izvorne datoteke

V okviru HTML velja omeniti še CSS (angl. Cascading Style Sheets), ki je z njim neposredno povezan. Gre za poseben način zapisa atributov, povezanih s formatiranjem dokumentov HTML. Sicer jih lahko pišemo tudi znotraj prve značke elementa HTML, vendar jih zaradi preglednosti raje zapišemo znotraj <head> elementa ali še bolje, v svojo datoteko. Taka datoteka ima končnico ".css".

2.2.2 PHP

PHP (angl. PHP: Hypertext Preprocessor) [4] je skriptni jezik, ki se izvaja na strežniku. Njegov glavni namen je ustvarjanje datotek HTML na osnovi vnesenih vhodnih podatkov. Z drugimi besedami, s pomočjo PHP lahko ustvarjamo dinamične spletne strani. Trenutno je PHP v različici 5, v prvotni obliki pa je nastal že leta 1995.

Kodo PHP pišemo v isti dokument kot kodo HTML, pri čemer ju lahko ločimo na več načinov. Najpogostejši in tudi način, katerega sem sam uporabljal, je pisanje kode PHP znotraj začetnega (<?php) in končnega (?>) ločila. Značilnost jezika je tudi to, da se vse spremenljivke začnejo z dolarskim znakom $. Ostala sintaksa je precej podobna C-jezikom, od ustvarjanja in klicanja funkcij

pa do zank *if*, *while* ter *for*. Namesto končnice ".html" imajo datoteke s kodo PHP končnico ".php". Na Slika 3 vidimo primer kode PHP z elementom HTML, kjer se opazi razlika med pisanjem komentarjev znotraj obeh jezikov.

```
<?php
     //To je PHP del dokumenta
?>

<!DOCTYPE HTML>
<HTML>
     <!-- To je HTML del dokumenta -->
</HTML>
```

Slika 3: Primer dokumenta PHP s kodo HTML

Pri jezikih, ki se izvajajo na strežnikih, obstaja kar nekaj alternativ, na primer ASP, Java, Perl. Po posvetovanju z mentorjem sem se za PHP odločil iz več razlogov, in sicer:

- jezik je zelo razširjen po Sloveniji in svetu,
- uporablja se ga lahko brez stroškov,
- je zelo dobro dokumentiran,
- ni vezan na določen operacijski sistem.

Glavna kritika, ki sem jo zasledil v literaturi, je letela predvsem na počasnejšo zmogljivost PHP v primerjavi s konkurenco, kar pa za moje potrebe ni bilo ključnega pomena.

2.2.3 JavaScript

JavaScript je skriptni programski jezik, ki je standardno integriran v vseh priljubljenih spletnih brskalnikih. Izvaja se na strani odjemalca za razliko od prej omenjenega PHP, ki se izvaja na strežnikih. Njegova glavna značilnost je, da omogoča precej boljšo interaktivnost in uporabniško izkušnjo, prav tako pa je pomemben člen pri programiranju po principu AJAX.

Jezik se je prvič pojavil leta 1995, ko ga je Netscape ustvaril z namenom poenostaviti programiranje začetnikom. Zaradi tega je doživel kar precej kritik, a je vseeno postal standardiziran v okviru razvijanja spletnih strani, s prihodom principa AJAX pa tudi bolj priljubljen med profesionalnimi razvijalci.

```
<script>
     document.write("<p>My First JavaScript</p>");
     var spremenljivka = 'tekst';
</script>
```

Slika 4: Primer kode Javascript

Na sintakso je precej vplival C-jezik, vendar pa ima Javascript nekaj posebnosti. Na primer, spremenljivkam ni treba definirati tipa, povemo le, da gre za spremenljivko z oznako "var" (glej Slika 4). Prav tako jezik nima klasične podpore objektnega programiranja, pač pa to dosežemo preko posebnih prototipskih funkcij. Za razliko od PHP Javascript ne potrebuje posebnih oznak, ampak lahko pišemo znotraj elementa <script> v kodi HTML.

2.2.4 SQL

Skoraj vsaka spletna stran potrebuje tudi pripomočke za delo z bazami podatkov in prav za ta namen se uporablja SQL (angl. Structured Query Language) [5]. Gre za povpraševalni jezik za delo z bazami podatkov. Njegovi glavni odliki sta standardizacija in razširjenost, kar pomeni dobro dokumentiranost jezika in enostavnost uporabe. Podatki se v bazo zapisujejo v obliki tabel.

Z bazo SQL komuniciramo preko angleških stavkov, ki so zelo podobni naravnemu jeziku. Tako na primer iz neke tabele v bazi poiščemo želen podatek s stavkom "SELECT polje FROM tabela WHERE pogoji". Na podoben način ustvarjamo tabele z ukazom "CREATE" ali pa izberemo bazo podatkov z ukazom "USE".

Jezik je nastal že v zgodnjih sedemdesetih letih prejšnjega stoletja, pomembnejša mejnika pa sta leto 1986 in 1987, ko sta ga organizaciji ANSI (angl. American National Standards Institute) in ISO (angl. International Organization for Standardization) standardizirali.

```
CREATE TABLE users (
     userid int unsigned not null auto_increment primary key,
     username varchar(100) not null,
     email varchar(100) not null,
     password varchar(42) not null
);
```

Slika 5: Primer ustvarjanja tabele v jeziku SQL

2.3 Programska orodja

2.3.1 Apache

Apache, polno ime Apache HTTP Server [6], je strežnik za prenos podatkov preko spleta. Deluje na osnovi HTTP-ja (angl. HyperText Transfer Protocol), najbolj razširjenega protokola za komunikacijo med strežnikom in odjemalcem na spletu. Na uradni spletni strani strežnik opisujejo kot robusten, zmožen, primeren za komercialne namene in verjetno še najpomembneje, odprtokoden. Prav tako je strežnik vsakomur dostopen zastonj in dobro dokumentiran, kar je tudi pripomoglo k njegovi priljubljenosti. Odločitev o izbiri prav tega strežnika za podlago testiranja spletnega vmesnika zato ni bila težka.

Projekt Apache je nastal leta 1995, ko je majhna skupina programskih inženirjev zbrala skupaj vsak svoje popravke takratnega predhodnika Apache z namenom narediti enotno distribucijo strežniškega programa. Ustanovili so The Apache Software Fundation, ki še danes posodablja strežnik. V času tega pisanja je najsodobnejša različica 2.4.3.

2.3.2 MySQL

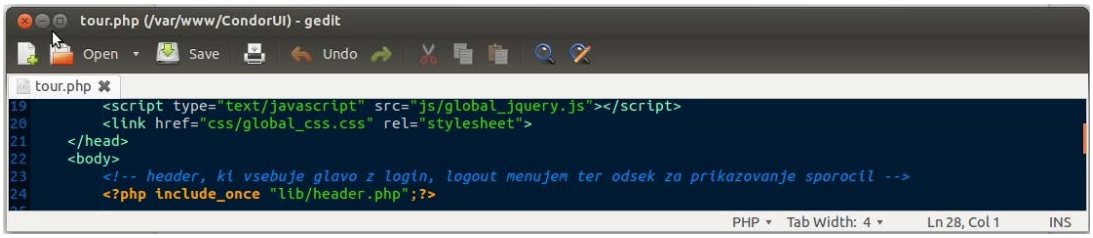
MySQL je skupek orodij za hranjenje, spremljanje in obdelavo baze podatkov. Kot že ime pove, se s strežnikom komunicira s pomočjo jezika SQL. Na voljo je v več paketih, najosnovnejši (MySQL Community Server) je na voljo zastonj. Poudariti je tudi treba, da gre za odprtokodno programsko opremo, zato lahko pri razvoju sodeluje tudi skupnost.

Okrnjena različica sicer ponuja zgolj strežnik z dostopom do baze podatkov, če pa želimo dostopati tudi do vseh naprednih orodij za spremljanje in nadzor naše baze, moramo poseči po plačljivi izdaji Enterprise. Sicer se z MySQL upravlja preko terminala, seveda z uporabo sintakse SQL, lahko pa uporabimo tudi brezplačni program MySQL Workbench, ki omogoča urejanje in upravljanje baze preko spletnega vmesnika.

MySQL ima trenutno v lasti podjetje Oracle, potem ko ga je odkupilo od prejšnjega lastnika Sun Microsystems. V času nastajanja te diplomske naloge je bila aktualna različica 5.5.28.

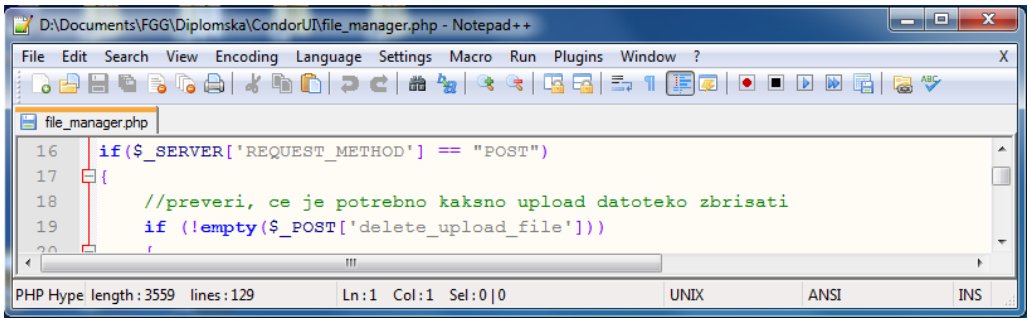
2.3.3 Gedit / Notepad++

Tako Gedit kot Notepad++ sta tekstovna urejevalnika, primarno namenjena pisanju in urejanju izvorne kode. Prvi je pisan za operacijski sistem Linux, drugi pa za okolje Windows.



Slika 6: Urejevalnik besedila Gedit

Glavni prednosti v uporabi pred drugimi tekstovnimi urejevalniki sta sposobnost barvnega označevanja kode različnih programskih jezikov ter odpiranje in samodejno ohranjanje zavihkov, ki omogočajo lažje preklapljanje med različnimi datotekami z izvorno kodo. Oba omogočata tudi označevanje števila vrstic, kar nakazuje, da sta programa primarno namenjena pisanju kode. Imata pa tudi slabosti. Ker gre zgolj za urejevalnika teksta, ni omogočeno samodejno iskanje napak in razhroščevanje, kot to omogočajo bolj profesionalni programi za razvijanje, na primer Visual Studio.

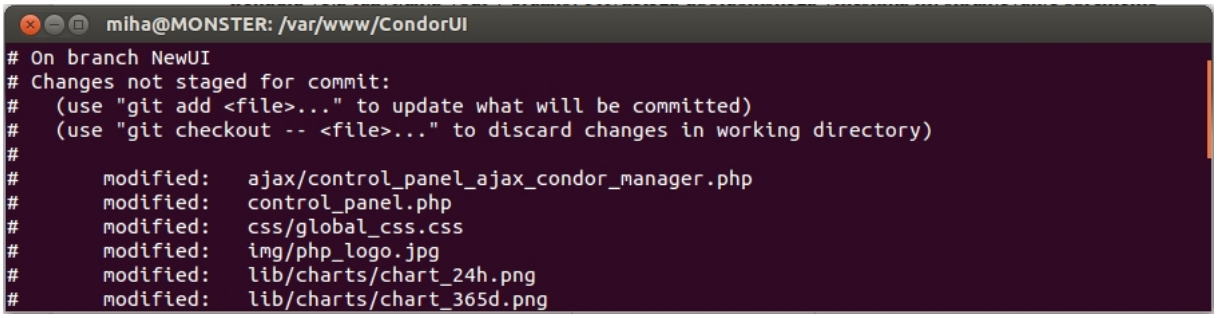


Slika 7: Izgled Notepad++ programa

Oba urejevalnika sta sicer odprtokodna po licenci GNU GPL in sta pod takimi pogoji tudi zastonj dostopna.

2.3.4 GIT

Git je odprtokodni sistem za nadzor in upravljanje z različicami izvorne kode. Na poseben način omogoča shranjevanje kode, in sicer tako, da se shranjuje spremembe glede na zadnjo objavo. Tako imamo vpogled v celotno zgodovino sprememb naše kode in se lahko vrnemo na prejšnje stanje, če trenutna veja razvijanja vodi v prazno. Privzetega uporabniškega vmesnika ni, shranjevanje sprememb se izvaja preko terminalnih ukazov.



Slika 8: Primer izpisa pri git ukazu status

Ukazi za komuniciranje z git sistemom so preprosti. Začnejo se z besedo "git", nadaljujejo pa z želenim ukazom, na primer "add" za dodajanje sprememb, "commit" za dejansko objavo sprememb v sistem. Seveda se moramo prej v terminalu nahajati znotraj delovne mape, kjer so shranjene tudi vse git datoteke za nek projekt.

Sistem je tesno povezan s spletno stranjo github.com, ki omogoča nalaganje projektov in tudi sprememb na strežnik. To ima dodano vrednost, saj imamo s tem grafični vpogled v različne objave, veje in celotno zgodovino projekta. S pomočjo te spletne strani se tudi precej laže poveže v skupino in sledi, kdo je kaj prispeval k projektu.

2.4 Tehnike programiranja

2.4.1 AJAX

Asynchronous JavaScript and XML (glej [7]) je novejši pristop k programiranju spletnih strani, kjer s pomočjo objekta XMLHttpRequest pošljemo strežniku zahtevo, naj si bo to POST ali GET request, lahko pa tudi kaj zahtevnejšega, na primer datoteke. Strežnik potem informacije procesira kot ob normalni zahtevi in jih preko istega objekta vrne nazaj odjemalcu, ki jih prikaže. Glavna prednost takega pristopa je predvsem v uporabniški izkušnji. Celotna izmenjava med strežnikom in odjemalcem namreč poteka asinhrono v ozadju. Uporabnik tako ni izpostavljen vnovičnemu nalaganju celotne strani, ampak se posodobi le njen določen del.

Pri izvedbi principa AJAX se uporablja več tehnologij, predvsem HTML in CSS za prikaz in videz, DOM in XML za dinamični prikaz in interakcijo s podatki ter seveda Javascript, ki vse tehnologije poveže in tudi vsebuje objekt XMLHttpRequest. Tako gre pri AJAX-u za skupek tehnologij, ki nam omogočajo modernejšo in uporabniku prijazno prikazovanje spletnih strani.

2.4.2 Objektno orientirano programiranje

Pri objektno orientiranem programiranju (v nadaljevanju: OOP) gre za paradigmo reševanja problemov, kjer kodo strukturiramo v objekte, s tem pa občutno zmanjšamo količino proceduralne kode in jo naredimo preglednejšo, laže berljivo.

Razred je osnovni koncept objektnega programiranja. Gre za neke vrste načrt oziroma strukturo objekta, kjer so definirane vse spremenljivke (podatki in atributi) in vse funkcije (metode), ki jih lahko nek objekt uporablja. S pomočjo teh funkcij in spremenljivk lahko potem manipuliramo s primerkom razreda, ki mu rečemo objekt.

Koda OOP je tako zgrajena iz številnih objektov, ki se med sabo prepletajo in tvorijo nek smiseln algoritem. Glavne prednosti takega načina programiranja so:

- preglednejša koda, kateri je laže slediti,
- hitrejše iskanje in popravljanje napak,
- omogoča lažje in učinkovitejše delo v skupini,
- koda je laže prenosljiva tudi na druge projekte.

Glavna slabost pa je predvsem ta, da je na konceptualni ravni tako kodo teže dobro strukturirati in oblikovati.

2.5 Knjižnice

2.5.1 jQuery

Kot je zapisano na uradni strani jQuery [8], gre za hitro in jedrnato Javascript knjižnico, ki poenostavi navigiranje po elementih HTML, njihovoanimacijo in interakcijo AJAX. Predvsem pa gre za knjižnico, ki spremeni način pisanja jezika JavaScript. Glavne prednosti sintakse jQuery so enostavnost, preglednost in skrajšanje zapisane kode. Vsi ukazi znotraj knjižnice so tudi kompatibilni s številnimi spletnimi brskalniki in ustrezajo standardom CSS3. Korenine knjižnice segajo v leto 2005, ko je nekaj posameznikov iskalo rešitev za lažjo in bolj skrčeno obliko naslavljanja izbirnikov CSS.

2.5.2 jQuery Form

jQuery Form je dodatek, ki razširi zmožnosti osnovne knjižnice jQuery. Natančneje, dodana je možnost predložitve obrazca HTML preko funkcije .ajax() znotraj osnovne knjižnice jQuery. Taka razširitev omogoča tudi prenašanje datotek preko principa AJAX, kar po statusu privzeto ni mogoče. Dodatna knjižnica je delo Mika Alsupa, ki jo nudi na svoji spletni strani [9].

2.5.3 Twitter Bootstrap

Twitter Bootstrap je priljubljeno ogrodje za razvoj čelnega dela sistema. Deluje na principu prednastavljenih slogov, ki jih z elementi HTML povežemo preko atributa razred. Nastavljamo in spreminjamo lahko tako videz kot interaktivnost spletnega vmesnika. Slednje nam omogočajo tudi integrirane funkcije JavaScript, ki predstavljajo drugi del ogrodja. Za delovanje je nujno potrebna knjižnica jQuery. Vse podatke o tem ogrodju najdemo na njihovi uradni spletni strani Twitter Bootstrap [10].

2.5.4 pChart

Ena od možnih rešitev za risanje grafov znotraj jezika PHP je pChart. Gre za objektno orientirano knjižnico, ki nudi učinkovito glajenje robov in različne vizualne učinke, kot so transparentnost, senčenje grafov in samodejno interpolacijo. Več informacij se dobi na njihovi uradni spletni strani pChart [11].

3 HTCONDOR

V tem poglavju je predstavljen sistem HTCondor, ki je eden od možnih visoko-prepustnih računskih okolij. Opisane bodo njegove splošne značilnosti kot tudi možnosti uporabe. Ne nazadnje pa je tudi vzpostavljen na Fakulteti za gradbeništvo in geodezijo, zato je del tega poglavja posvečen tudi temu. Opisan je sam sistem, povezane naprave in zgodovina postavitve.

3.1 Splošno

Kot že v teoretičnem poglavju omenjeno, je cilj visoko-prepustnih računskih okolij združiti večje število računskih virov in tako omogočiti njihovo učinkovito rabo ter posledično povečati celotno procesorsko moč, ki jo imamo na voljo v danem trenutku. Eno takih okolij je projekt HTCondor, katerega naloga je upravljanje delovne obremenitve za računsko intenzivna opravila [12].

Že več kot desetletje ga razvijajo na ameriški univerzi v Wisconsinu, kjer gre smer razvoja v povečevanje učinkovitosti celotnega sistema naprav, ne pa toliko v povečevanje učinkovitosti posameznih naprav. V času izdelave diplomske naloge je najsodobnejša različica 7.9.1, na kateri sem tudi testiral uporabniški vmesnik.

Za vodenje, sledenje in upravljanje celotnega sistema skrbi večje število manjših procesov, imenovanih *daemons*, ki se izvajajo v ozadju na vsaki izmed povezanih naprav. Ti procesi so uporabnikom nevidni in nimajo robustnega privzetega grafičnega vmesnika, vsa komunikacija z njimi poteka preko terminala s pomočjo konzolnih ukazov. Poleg pregledovanja sistema se lahko tudi briše vnose ali pa dodaja nove preko pripravljenih predložnih datotek. Vanje zapišemo vse parametre in argumente, ki jih želimo posredovati v sistem. Ta uporabniška izkušnja pa me posredno pripelje do enega glavnih vzrokov za potrebo po boljšem vmesniku, ki bo opisan v naslednjem podpoglavju.

3.2 Prednosti in slabosti

Ker gre pri sistemu HTCondor za visoko-prepustno računsko okolje, ima kar nekaj prednosti, ki so neločljivo povezane s takim okoljem. Omogoča na primer optimalen izkoristek virov, ko so ti v prostem teku. Tudi samostojno vodi upravljanje z viri, tako da uporabniku ni treba niti nima možnosti ročno izbirati napravo za izračun. S tem lahko pri večjem številu izračunov kar precej prihranimo na času. Čas pa ni edino sredstvo, ki ga s sistemom HTCondor prihranimo. Tudi s finančnega vidika je ugodnejši, saj so superračunalniki precej dražji od navadnih računalnikov, pri čemer se strošek slednjih porazdeli tudi v druge namene. Na primeru računanja analiz IDA [13] se je izvedlo teste učinkovitosti, ki so primerjali pohitritev računanja s povečanjem števila računalnikov. Kot je razvidno iz Preglednica 1, so s 25 računalniki dosegli faktor pohitritve 24,52. Če vzamemo, da je faktor 25 maksimalna pohitritev, potem gre za 98,08 % učinkovitost sistema HTCondor.

Preglednica 1: Analiza učinkovitosti računskega okolja

Število računskih analiz: 280 Povprečen čas analize: ~ 13 min.		
Št. računalnikov	Čas izvajanja [ure]	Faktor
1	61,3	1,00
5	14,7	4,17
10	7,1	8,63
25	2,5	24,52

A slabosti tehnologije HTCondor so tiste, ki so pripeljale do nastanka te diplomske naloge. Prvi večji problem je privzeti vmesnik HTCondor, ki je konzolno okno. Ta je uporabniku neprijazen, saj zahteva učenje ukazov na pamet, nima možnosti navigiranja z miško in prikazuje rezultate zgolj v obliki znakov AASCI. Druga večja slabost pa je povezana s samim načinom pošiljanja podatkov v izračun. HTCondor namreč zahteva, da je naprava, ki je oddala predlogo za izračun, ves čas opravljanja izračuna prižgana. To pomeni, da bi moral vsak uporabnik pustiti svoj računalnik prižgan ves čas računanja, če bi želel iz njega pošiljati predloge. To je seveda precej stroga omejitev, saj znajo določeni izračuni trajati tudi več dni ali celo tednov.

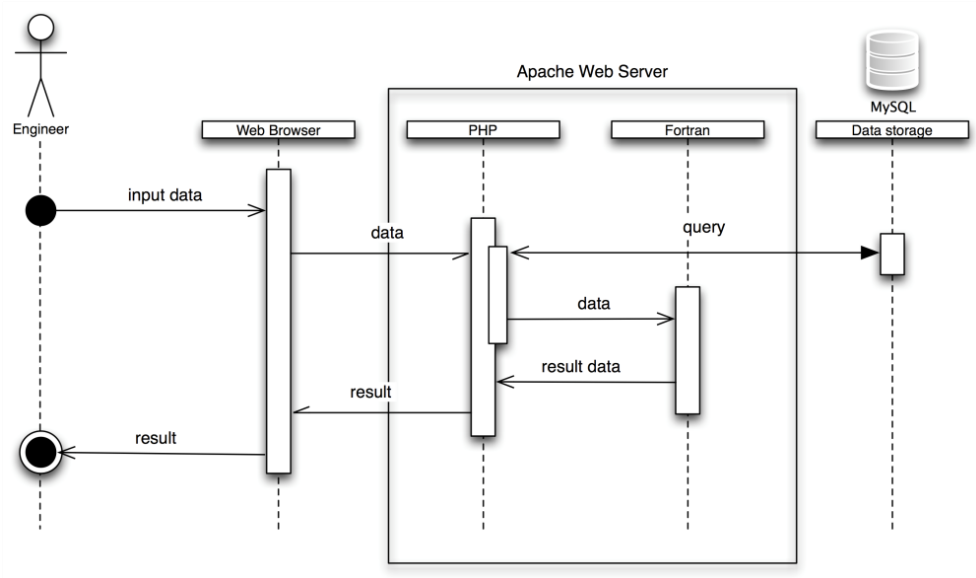
Obe zgoraj opisani slabosti sta pripeljali do zaključka, da bi bilo pametno izdelati nek drug vmesnik. Za prvi problem bi bil dovolj dober že kakršen koli grafični vmesnik, ki po videzu in uporabniški izkušnji spominja na aplikacije v operacijskem sistemu Windows. Druga slabost pa teži k temu, da bi bilo pametneje narediti spletni vmesnik. Ker bo ta lociran na strežniku, bo tudi slednji pošiljal vse predloge v sistem HTCondor. To bo omogočilo, da uporabnik preko spletnega brskalnika na svojem računalniku pošlje datoteke v izračun, ni pa mu ga treba pustiti prižganega, če ta seveda ni del naprav, ki lahko računajo zastavljeni problem. Edina naprava, ki bo tako morala ves čas delovati, je strežnik, na kateremu se nahaja spletni vmesnik.

3.3 Uporaba

Vzpostavljeni sistem HTCondor se bo na fakulteti primarno uporabljal za računanje potresnih analiz. Bolj specifično, v nekem primeru se bodo na osnovi vstavljenih različnih akcelelogramov pridobili rezultati. Ker pa je teh akcelelogramov veliko in izračun vsakega poteka precej časa, je smiselno izračun posredovati v sistem HTCondor. Tako bo za vse številne izračune optimalno poskrbljeno in bodo izračunani v precej krajšem času, kot če bi jih ročno računali na enem računalniku.

Računanje z uporabo omenjenih akcelelogramov je podrobno opisal Matjaž Dolšek [14]. Z uporabo programskega ogrodja OpenSees in zanj spisanega dodatka PBEE toolbox dobi presojo potresnega obnašanja armiranobetonskih stavb. Slednje je skupek funkcij Matlab, s pomočjo katerih lahko uporabimo eno izmed mnogih nelinearnih metod za računanje potresne odpornosti armiranobetonskih konstrukcij. Konkretno v primeru, ki sem ga sam uporabil na vmesniku, je šlo za računanje IDA (angl.

incremental dynamic analysis) krivulj na modelu z eno prostostno stopnjo. Taka krivulja se izračuna na osnovi vstavljenega večjega števila začetnih vhodnih parametrov, vključno s seznamom vseh akcelelogramov, za katere želimo izračunati IDA krivuljo. Rezultati tega izračuna pa predstavljajo podlago za pridobitev približne IDA krivulje, ki bo veljala za poljubne vhodne podatke, a bo prav tako narejena za model z eno prostostno stopnjo. Za ta izračun je bila v okviru drugega projekta že narejena spletna aplikacija in je vzpostavljena na Fakulteti za gradbeništvo in geodezijo v Ljubljani. Na Slika 9 je shematsko prikazan diagram zaporedja te aplikacije, ki uporablja podobne tehnološke rešitve kot spletna aplikacija, razvita v okviru te diplomske naloge.



Slika 9: Diagram zaporedja spletne aplikacije [14]

Sicer pa se lahko visoko-prepustna računska okolja uporabljajo tudi v drugih panogah, ne samo znotraj gradbenega inženirstva. Na primer v biologiji se uporabljajo za računanje molekularnih struktur ali pa elektronske interakcije proteinskih delcev. Tudi v zabavni industriji, bolj specifično industriji videoiger, so se v zadnjih letih pojavili novi načini igranja v oblaku. Ti potrebujejo omrežje računalnikov, ki med sabo sodelujejo in trenutnim uporabnikom nudijo dovolj procesorske moči za poganjanje najnovejših videoiger. Tukaj se pojavi še en problem, ki sicer ni prisoten pri ostalih primerih, namreč dosežena mora biti dovolj hitra odzivnost naprav in posledično nizka zakasnitev.

3.4 HTCondor na UL FGG

V nadaljevanju bodo kronološko opisani nekateri pomembni koraki v razvoju visoko-prepustnega računskega okolja na UL FGG.

2005: V tem letu se je v okviru projekta EU InteliGrid [15], ki so ga koordinirali člani Katedre za gradbeno informatiko (KGI), začel razvoj visoko-prepustnega računskega okolja na UL FGG. Za izvajanje računskih nalog so se uporabljali računalniki v računalniških učilnicah RU-I/5 in RU-III/6

ter starejši računalniki v prostorih KGI. Skupaj je to pomenilo približno 50 računalnikov z operacijskim sistemom Windows XP in 8 računalnikov z operacijskim sistemom Debian GNU/Linux 3.1.

2005–2007: V tem času se je računsko okolje uporabljalo predvsem v namen raziskav na projektu InteliGrid in delno tudi projektu DataMiningGrid [16]. Pridobilo se je potrebno znanje in izkušnje za upravljanje in uporabo računskega okolja HTCondor, predvsem pa so se pokazale možnosti uporabe računskega okolja v potresnem inženirstvu.

2007: Jaka Zevnik je izdelal doktorsko disertacijo z naslovom »Potresna ranljivost armiranobetonskih viaduktov s škatlastimi stebri« [17]. V disertaciji je uporabil računsko okolje za izvedbo analize IDA [18]. Na tem primeru je bila izdelana tudi analiza učinkovitosti računskega okolja (tabela 1) za izvajanje parametričnih študij. Analiza je pokazala, da je v visoko-prepustnih računskih okoljih pomembno predvsem število računalnikov za izvajanje nalog in ne njihova računska moč.

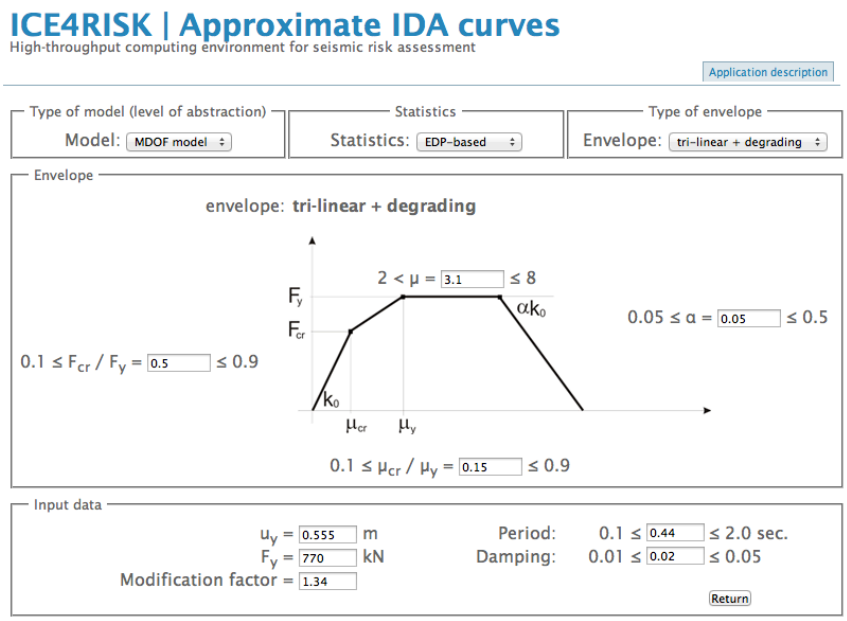
2008: Začetek temeljnega raziskovalnega projekta ICE4RISK [18], ki je bil usmerjen v razvoj visoko-propustnega računskega okolja za analize potresnega tveganja.

2009: V okviru razpisa ARRS za sofinanciranje nakupov raziskovalne opreme je bilo računsko okolje nadgrajeno z desetimi namenskimi strežniki (Slika 11). Računalniki v računalniških učilnicah so se takrat prenehali uporabljati za izvajanje nalog.

2011: Zaključil se je projekt ICE4RISK, v okviru katerega je bilo razvito računsko okolje za analizo potresnega tveganja. V okviru projekta je bila z uporabo računskega okolja izdelana podatkovna zbirka potresnih odzivov sistema z eno prostostno stopnjo. Podatkovna zbirka se uporablja za izračun približnih IDA krivulj (Slika 10).

2013: Visoko-prepustno računsko okolje trenutno vključuje:

- centralni računalnik (kgi-cl.fgg.uni-lj.si), ki upravlja s celotnim računskim okoljem in omogoča oddaljen dostop in uporabo računskega okolja;
- devet namenskih strežnikov (4xIntel Xeon CPU L5520 2.27GHz, 16GB RAM) z operacijskim sistemom Ubuntu Linux 12.04 LTS 64-bit za izvajanje nalog;
- dva namenska strežnika (2xIntel Xeon CPU L5520 2.27GHz, 8GB RAM) z operacijskim sistemom Windows 7 64-bit za izvajanje nalog;
- diskovno polje Fibre Channel 1TB;
- programsko okolje: HTCondor 7.8.7, MATLAB 2012b, OpenSees (več različic) idr.



Slika 10: Izračun približnih IDA krivulj

Slika 11: Računalniki visoko-prepustnega računskega okolja UL FGG

4 UPORABNIŠKI VMESNIK

Poglavje o uporabniškem vmesniku je najobsežnejše, saj bo tukaj zajeta večina dela te diplomske naloge. Podpoglavja so razporejena kronološko, približno tako kot je potekala izdelava vmesnika. Začelo se je pri postavitvi zahtev za vmesnik, katere elemente točno potrebuje, in nadaljevalo pri povezovanju vseh elementov v neko logično celoto. Slednje je pripeljalo do načrta projekta. Nato je bil določen vizualni dizajn strani in postavitev vseh elementov po različnih straneh. Paziti je bilo treba, da je bila uporabniška izkušnja optimalna in intuitivna. Sledila je izvedba projekta, kjer je bilo treba sprogramirati tako programsko zaledje kot čelni del programa. Zadnje podpoglavje pa je posvečeno opisom različnih scenarijev uporabe vmesnika. Za konkretno implementacijo problemov, ki bodo samo opisani, ne pa tudi nazorno prikazani, si lahko bralec pogleda vso izvorno kodo, ki je javno dostopna na spletnem portalu Github, vejitev NewUI [19].

4.1 Zahteve

Prva stvar, ki je potrebna za nastanek vmesnika, je določitev vseh zahtev in ciljev. Z drugimi besedami, določiti je treba vse elemente in koncepte, ki jih bo vmesnik vseboval. To sem z mentorjevo pomočjo določil na osnovi tega, kako in kdo bo vmesnik uporabljal.

Želja po reguliranju dostopa do vmesnika in sledenju uporabnikom je pripeljala do enega prvih potrebnih osnovnih elementov, to je upravljanje z računi. Delovalo naj bi tako, da ima vsak uporabnik svoje uporabniško ime in geslo, s katerima bo identificiran znotraj vmesnika. Poleg te osnovne funkcionalnosti mora biti uporabniku omogočeno, da geslo ali uporabniško ime spremeni, hkrati pa mora biti slednje edinstveno, kar sistem seveda zazna in na to opozarja sam. Tudi zmožnost ustvarjanja novih računov mora obstajati, pri čemer je dobrodošla opcija ustvarjanja časovno omejenih računov za namene preizkušanja sistema. In še kot zadnje, uporabniški sistem mora znati ločiti med različnimi tipi uporabnikov, na osnovi katerih določimo ali onemogočimo dostop do določenih predelov vmesnika (na primer administratorskih strani).

Drugi večji koncept bo hranjenje podatkov na strežniku, ki jih uporabniki sami naložijo. Strukturo datotek in map na disku strežnika je najbolje prikazati kar znotraj tabele. Tam morata biti minimalno omogočeni obe osnovni funkciji, ki ju potrebujemo. To sta nalaganje novih datotek in brisanje obstoječih. Poleg tega bo zelo priročno implementirati tudi navigacijo po mapah. Vse do sedaj opisano mora delovati tako, da lahko z datotečnim sistemom upravljamo tudi preko operacijskega sistema, spremembe pa se morajo samodejno pojaviti znotraj vmesnika. V okviru tega odstavka velja omeniti še potrebo po interakciji do sedaj opisanih sistemov, to sta upravljanje z računi in urejanje datotek. Ker zna biti potencialno veliko število uporabnikov, je smiselno, da se vsakemu uporabniku določi svoj prostor za hranjenje in eksekucijo datotek. To nam pravilna zasnova sledenju in pomnjenju uporabnikov tudi omogoča.

Naslednja zadeva je integracija prej omenjenih konceptov s sistemom HTCondor. V okviru tega je treba urediti kar nekaj zadev. Seveda so nujne vse operacije za pošiljanje HTCondorjevih datotek na izvajanje, a je pred tem še treba poskrbeti, da se bodo vse poti do izvršnih datotek in datotek z rezultati ujemale skladno z našo datotečno hierarhijo. Omogočeno mora biti tudi samodejno razpakiranje večjega števila datotek, skrčenih v en ZIP-arhiv. Ko imamo enkrat pravilno pošiljanje datotek vzpostavljeno, je treba dodati še prikazovanje vseh trenutnih operacij v izvajanju in tistih, ki so na čakanju znotraj HTCondorjevega bazena. To je najučinkoviteje kar znotraj tabel. Seveda morajo biti te dinamične in omogočati poljubno število prikazanih vnosov.

Uporabnikom bo treba prikazati tudi malce statističnih podatkov, na primer število zadnjih HTCondor vnosov v nekem časovnem obdobju. Za to bo treba vzpostaviti svojo bazo, kjer se bodo ves čas zapisovali vsi koraki, ki jih nek uporabnik naredi. Kasneje se ti podatki po potrebi obdelajo tako, da se ven izloči le relevantne podatke.

Vmesnik mora imeti tudi standardizirano mesto za prikazovanje sporočil. Tukaj bo uporabnik videl vsa sporočila za napake, opozorila, kratka navodila in druge napotke.

Zadnja stvar, ki jo bo vmesnik potreboval, je posebna stran za administratorje. Tukaj si bodo lahko pogledali veliko statističnih podatkov o strani in njenih uporabnikih, urejali in dodajali bodo lahko uporabnike, prav tako pa spreminjali njihove podatke. Stran bo vidna in dostopna samo, če ima uporabnik ustrezne pravice za dostopanje do nje.

4.2 Shematski načrt

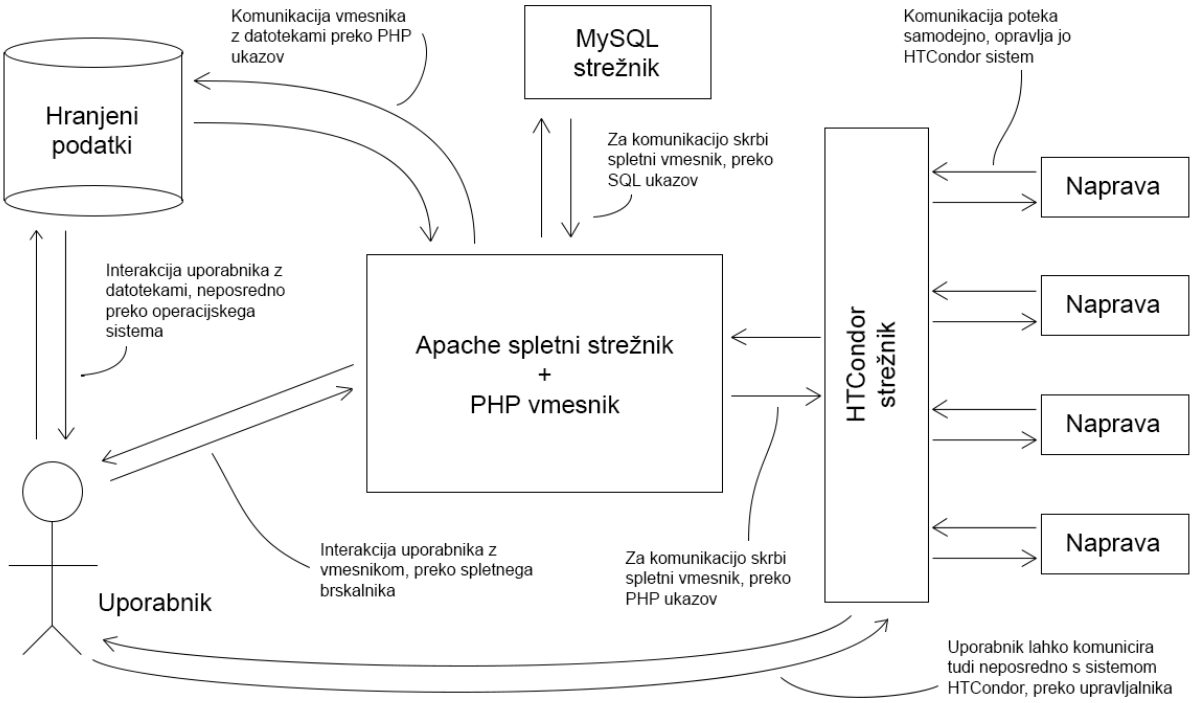
Spletni vmesnik tekom svoje uporabe ne bo zaprt in izoliran, ampak bo komuniciral z zunanjimi sistemi, ki so večinoma že vzpostavljeni in lahko delujejo neodvisno od vmesnika. Poleg povezovanja z uporabnikom preko spletnega brskalnika, ki bo podrobno opisano v okviru podpoglavja Videz vmesnika in uporabniška izkušnja, so taki sistemi trije:

- hranjenje baze podatkov o uporabnikih znotraj strežnika MySQL,
- hranjenje in upravljanje z datotekami na disku,
- sistem HTCondor.

Če želimo vedeti, kaj in kako bo treba implementirati znotraj vmesnika, si je pametno pogledati povezane sisteme na širšem, shematskem nivoju (glej Slika 12). Kot bomo videli kasneje v naslednjem poglavju, ima ta opredelitev kar precejšen vpliv na strukturo programa in njegovo implementacijo.

Centralni sistem, ki bo komuniciral z vsemi ostalimi sistemi, je seveda spletni vmesnik. Natančneje, povezovanje bosta izvajala spletni strežnik Apache in zaledni del programja, ki je spisan s pomočjo PHP in uporabnikom ne bo viden. Glavna povezava bo nedvomno tista med spletnim strežnikom in

sistemom HTCondor. Navsezadnje je prav potreba po tej komunikaciji diktirala nastanek tega vmesnika. PHP za interakcijo s sistemom HTCondor uporablja generične ukaze za izvajanje eksternih programov, kot je na primer funkcija exec(). Funkcija deluje identično, kot če bi ukaz pošiljal preko terminalnega okna v okviru operacijskega sistema. Ker slednji predstavlja privzet vmesnik za HTCondor, bo komunikacija trivialna in ukazi bodo identični tistim iz upravljalnika. Povratno informacijo HTCondor poda urejeno znotraj polja.



Slika 12: Shema komuniciranja vmesnika z ostalimi sistemi

Naslednja pomembna komunikacija je tista med spletnim strežnikom in podatkovno bazo, ki se nahaja znotraj strežnika MySQL. V prejšnjem poglavju je omenjena zahteva po reguliranju dostopa do vmesnika, kar pa posledično pomeni vzpostavitev sistema za prepoznavanje uporabnikov. Zaradi varnostnih razlogov imajo baze podatkov ločen strežnik in niso del spletnega strežnika. Vseeno pa je med obema potrebna komunikacija in v ta namen ima PHP vnaprej pripravljene funkcije za delo s številnimi podatkovnimi strežniki in eden izmed njih je uporabljen v tem vmesniku. Ukazi imajo sintakso SQL in njihovo pošiljanje ter prejemanje povratnih informacij je zelo podobno sistemu HTCondor s to razliko, da se ne uporabljajo generične funkcije, ampak specifično izdelane za izbrano podatkovno bazo. Poleg podatkov o uporabnikih se bodo na omenjenem strežniku hranili tudi različni statistični podatki.

Tretja interakcija, potrebna za vzpostavitev delujočega sistema, pa poteka med strežnikom in datotečnim sistemom. Slednji je potreben za hranjenje vhodnih in izhodnih podatkov, ki jih proizvede sistem HTCondor ali pa naloži uporabnik sam. Tudi za te operacije so znotraj PHP razvite metode,

preko katerih lahko ustvarjamo nove datoteke in mape, optično zajamemo vsebino mape ali pa jih pobrišemo.

Nazadnje bi omenil še neposredno interakcijo med uporabnikom, datotečnim sistemom in okoljem HTCondor. Vsa komunikacija med naštetim lahko poteka tudi izven uporabniškega vmesnika, če le imajo uporabniki znanje in ustrezen dostop. Posledično je treba zagotoviti, da bo interakcija vmesnika s temi sistemi kompatibilna in pravilno delujoča tudi v primeru sprememb, ki jih povzročijo zunanji dejavniki, ki lahko zaobidejo spletni vmesnik.

4.3 Struktura

Ena izmed prvih odločitev pri snovanju programa je izbira strukture datotek, ki bodo vsebovale vse funkcije, razrede, procedure in procese, ki tvorijo uporabniški vmesnik. Po pregledu literature na medmrežju [20] sta se kot najbolj optimalni izkazali dve možnosti.

Pri prvi možnosti osnovne datoteke (to so datoteke, ki jih neposredno kličemo v spletnem brskalniku) postavimo v korensko mapo projekta, vse ostale datoteke pa logično razporedimo v podmapah, na primer vse knjižnice gredo v mapo lib, vse datoteke JavaScript v mapo js, slike v mapo img itd.

Druga možnost pa predpostavlja, da se za vsako glavno sekcijo naredi svojo podmapo, kamor se vključi osnovno datoteko in vse knjižnice, ki so narejene po meri za to sekcijo. Vse globalne knjižnice in globalne datoteke se nahajajo znotraj mape *global*, ki leži v korenu projekta. Taka struktura je pisana na kožo predvsem večjim projektom z veliko količino odsekov, sekcij in kodami po meri.

Za manjše projekte, kakršen je tudi moj, je prva možnost optimalnejša, zato sem se odločil zanjo. Končna struktura datotek ima tako naslednjo podobo:

- /ajax (vsebuje vse datoteke, ki jih bomo klicali po principu AJAX),
- /css (vsebuje vse datoteke Cascading Style Sheet),
- /files (predel, kamor se bodo hranile HTCondor datoteke uporabnikov),
- /fonts (vsebuje uporabljene pisave, ki niso vključene kot privzete),
- /img (vsebuje slikovno gradivo),
- /js (vsebuje vse JavaScript datoteke),
- /lib (vsebuje vse knjižnice, ki se bodo vključevale osnovnim datotekam),
- osnovne datoteke (index.php, links.php, tour.php, status.php, control_panel.php itd.).

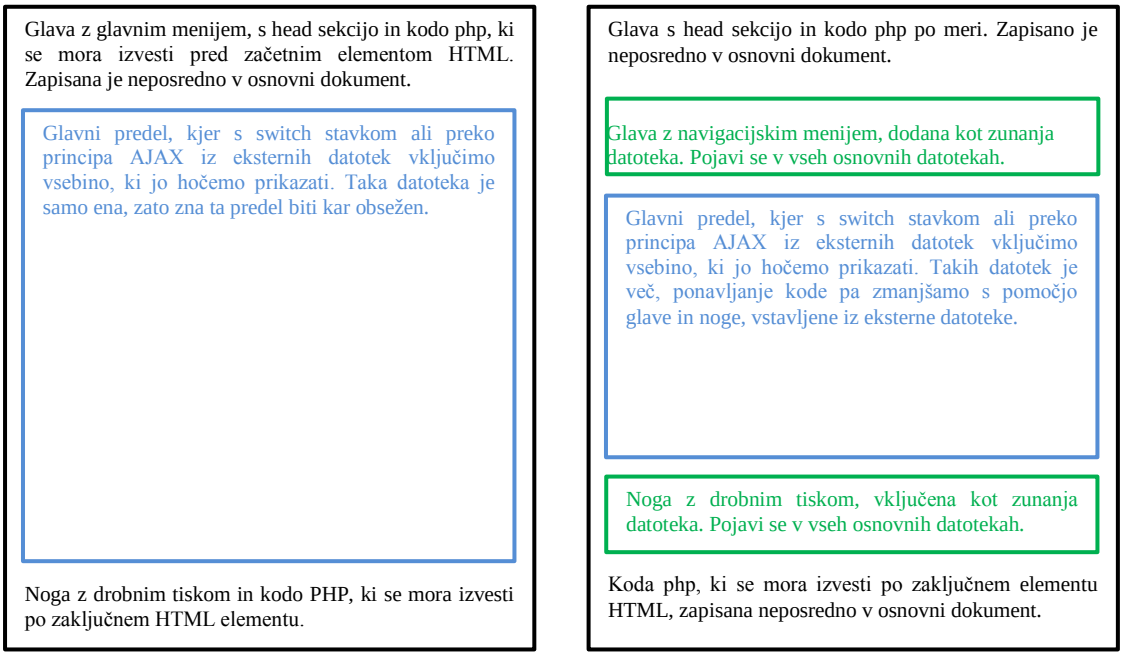
4.3.1 Osnovne datoteke

Tudi pri strukturi osnovne datoteke sta se iz več različnih virov [21, 22] pojavljali dve možnosti za njeno izvedbo, ki sta shematsko prikazani na Slika 13.

Prvo možnost tvori ena osnovna index datoteka, ki neposredno vsebuje zapisane vse metapodatke, prav tako tiste, ki se nahajajo v glavi elementa HTML. Tudi grafične elemente uporabniškega vmesnika, ki se pojavljajo po vseh predelih in so enaki, zapišemo v index datoteko. Ostala vsebina se vnaša ali z vstavljanjem eksternih datotek ali pa preko principa AJAX. Prednost takega načina strukturiranja je, da nimamo enake kode na več mestih, a to gre na račun preglednosti, sploh če imamo razvejano strukturo z veliko podsekcijami.

Druga možnost pa vsebuje več osnovnih datotek, katerih količina je enaka številu glavnih sekcij spletne strani. Vsaka taka osnovna datoteka vsebuje glavo in nogo, ki sta zapisani v eksterni datoteki in vneseni v vsako osnovno datoteko. Vsebujeta predvsem grafične elemente, kot so glavni navigacijski meni ter drobni tisk, ki se pojavlja in je enak na vseh predelih vmesnika. Ostala vsebina se, podobno kot prej, vnaša z vstavljanjem zunanjih datotek ali s pomočjo principa AJAX. Glavni prednosti takega načina izvedbe sta boljša preglednost strukture vmesnika in možnost vnosa določenih elementov po meri, za vsako glavno sekcijo posebej. Po drugi strani pa bomo določeno identično kodo našli v več različnih datotekah.

Po razmisleku sem se odločil za drugi pristop, saj mi zmožnost vnosa kode po meri znotraj vsake glavne datoteke odtehta slabosti te možnosti. Prav tako projekt ni zelo obsežen, zato malenkost več ponavljajoče se kode ni predstavljalo velike ovire.



Slika 13: Shema obeh pristopov strukture osnovne datoteke: pristop 1 (levo) in pristop 2 (desno)

4.3.2 Datoteke AJAX

Datoteke AJAX, ki se nahajajo v mapi ajax, so posebna vrsta datotek, ki bodo neposredno klicane in se posledično izvajale znotraj svoje korenske mape. Rezultati se bodo preko objekta XMLHttpRequest prenesli v osnovne datoteke, iz katerih bodo pred tem prišli ukazi za njihovo izvedbo.

Datotek AJAX je osem, za vsako od dveh glavnih osnovnih datotek (status.php in control_panel.php) po štiri. Struktura je podobna kot glavni predel osnovnih datotek, se pravi s switch stavkom reguliramo, kateri del procesa se bo izvedel na osnovi prejetega zahtevka POST. Če na izbiro ni več kot ene možnosti, switch stavek seveda izpustimo. Ti procesi so predvsem povezani z manipulacijo z datotekami, kot je na primer nalaganje, prenašanje na strežnik, brisanje datotek in map. Tudi precej HTCondor ukazov se izvede preko principa AJAX, navsezadnje pa ga vmesnik uporablja tudi za vso navigacijo po tabelah.

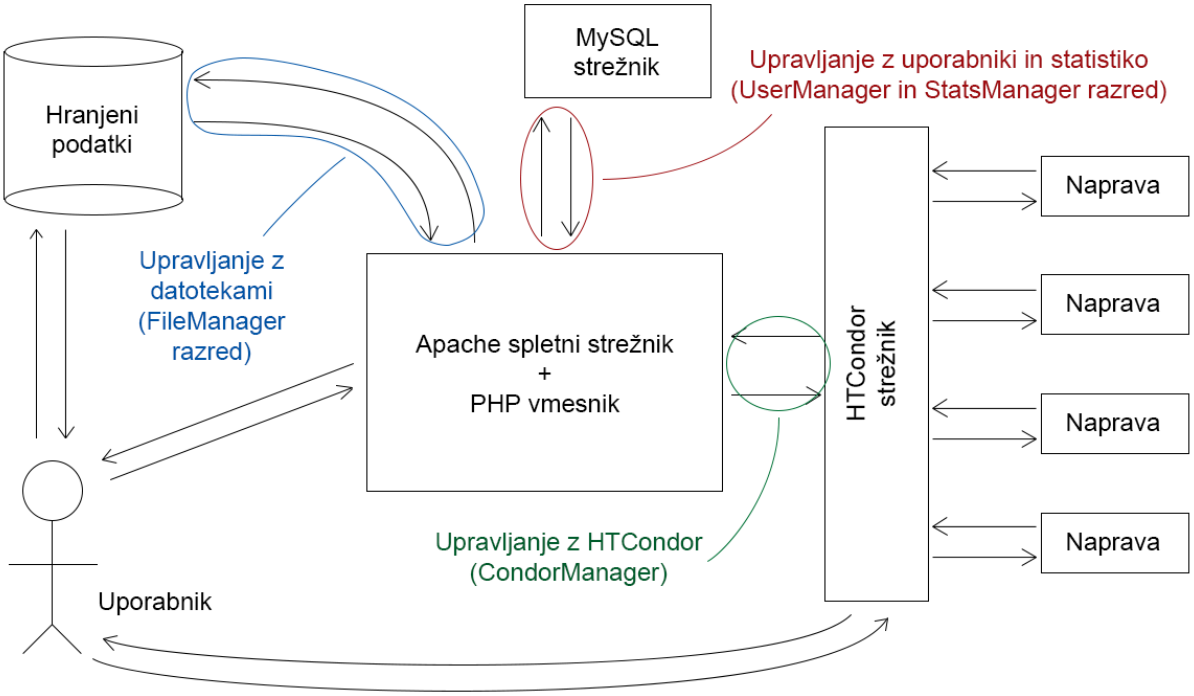
4.3.3 Vključene datoteke

Zadnja kategorija datotek so tiste, ki jih zgolj vključimo in povežemo znotraj drugih datotek in jih tako le posredno kličemo. V to kategorijo spada največ datotek, predvsem gre za razne knjižnice funkcij in razredov ter procedur, ki jih moramo v enaki obliki izvesti v več različnih datotekah (na primer preverjanje prisotnosti uporabnika, sledenje uporabnikom itd.). S tem zmanjšamo količino kode in tudi omogočimo, da se z modifikacijo na enem mestu spremembe poznajo povsod, kamor so te datoteke vključene. Omenjene in bolj podrobno opisane bodo v naslednjem podpoglavju o programskem zaledju.

4.4 Programsko zaledje

Programsko zaledje predstavlja vso kodo, ki jo poganja strežnik in ni vidna uporabniku oziroma odjemalcu. Ta dobi le končni produkt v obliki kode HTML, JavaScript in CSS, ki ga ob izvedbi ustvari programsko zaledje. V primeru tega vmesnika je bila celotna koda na strežniku spisana v programskem jeziku PHP.

Primarni del zaledja predstavljajo razredi, zapisani v datoteki z imenom classes.php ter vstavljeni v vsako datoteko, v kateri se uporabljajo. Program uporablja štiri razrede. To so UserManager razred za upravljanje z uporabniškimi računi, FileManager razred za upravljanje z datotečnim sistemom, CondorManager razred za upravljanje s sistemom HTCondor in StatsTracker za upravljanje s statističnimi podatki. Obstoj točno teh razredov se logično ujema s povezavami med spletnim vmesnikom in ostalimi sistemi, ki so opisani v podpoglavju Shematski načrt. Pri tem je treba poudariti, da tako UserManager kot StatsTracker razred opravljata funkcijo povezovanja vmesnika z bazo MySQL, a imata drug namen in sta posledično ločeno spisana s pomočjo podedovanja. Pregled razredov, njihov namen in mesto upravljanja prikazuje Slika 14.



Slika 14: Vsi štirje razredi in njihovo mesto delovanja

Poleg razredov so v tem poglavju opisane še globalne funkcije, implementacija sistema za prikazovanje sporočil, sistem za risanje grafov s knjižnico pChart, baza podatkov MySQL ter pripadajoče tabele.

4.4.1 Upravljanje z računi (UserManager razred)

Del vmesnika, ki služi za upravljanje z računi, verjetno ni med kompleksnejšimi, je pa gotovo med najpomembnejšimi, saj med drugim nadzoruje tudi dostop do datotek ter podatkov uporabnika in posledično deluje kot varovalka pred nezaželenim dostopom.

Primarni del upravljanja z računi se nahaja znotraj razreda UserManager. Tam so definirane vse metode za preverjanje, prikazovanje, vnašanje, brisanje in spreminjanje podatkov o uporabnikih, ki so zapisani v podani bazi. S slednjo se povezava vzpostavi že v konstruktorju in je tako ves čas delovanja primerka razreda aktivna in zapisana v lokalni zasebni spremenljivki. Konstruktor tudi poskrbi, da se začne nova ali nadaljuje obstoječa seja. Poleg prej omenjenega podatka razred lokalno shranjuje še vse vnose iz baze podatkov za posameznega uporabnika (na primer njegovo uporabniško ime, geslo, ID itd.). Tudi ti so zasebni in posledično nedostopni izven samega razreda. Po drugi strani so vse metode javne, kar pomeni, da jih lahko kličemo izven razreda.

Seveda pa razred sam zase še ni uporaben, imeti mora spisane tudi postopke, kjer se ustvari primerek razreda in je uporabljen v nekem procesu. V okviru upravljanja z računi imamo pet takih datotek, ki pretežno uporabljajo UserManager razred ali pa so kako drugače namenjene upravljanju z dostopom.

Najpomembnejša datoteka je **access_control.php**. Znotraj je spisan kratek, a ključen postopek, ki ustvari nov primerek razreda UserManager (s tem se tudi nadaljuje ali ustvari nova seja) in preveri vneseno uporabniško ime ter geslo oziroma odjavi uporabnika, če je to potrebno. Poleg tega tudi poskrbi, da ne bo prišlo do dvojnega pošiljanja obrazca [23]. Oboje skupaj pa pomeni, da je ta postopek vključen v vse datoteke, ki jih neposredno kličemo, saj povsod potrebujemo podatke o trenutnem uporabniku in hkrati zagotovimo, da bo uporabniška izkušnja povsod gladka, brez problemov z dvojnim pošiljanjem obrazca.

Kot je bilo že prej v besedilu omenjeno, **header.php** vsebuje celoten navigacijski meni, ki se pojavlja v vseh neposredno klicanih datotekah. Del tega navigacijskega menija pa je med drugim tudi obrazec za prijavljanje in odjavljanje uporabnika, ki seveda spada pod upravljanje z računi. Obrazec vnesene podatke, to sta ali uporabniško ime in geslo ali ukaz za odjavo, posreduje prej omenjeni datoteki (access_control.php), kjer imamo spisan postopek za njihovo obdelavo.

Znotraj datoteke **footer.php** se nahaja le zelo majhen segment, povezan s kontrolo dostopa. Namenjen je pravilni ponastavitvi dostopa v primeru, da je bil uporabnik odjavljen na nestandarden način.

Register.php je, kot nakazuje ime, datoteka za registriranje uporabnikov. Razdeljena je na dva ključna dela. Prvi vsebuje postopek, ki preveri veljavnost podatkov in v primeru uspešnosti doda uporabnika v bazo podatkov. Vse to se izvaja s pomočjo UserManager razreda, ki ima že pripravljene metode za taka opravila. Drugi predel datoteke pa vsebuje grafični prikaz obrazca, kamor bodoči uporabnik vnese svoje podatke. Nekateri se morajo tudi ustvariti na osnovi uporabnikove izbire. Seveda sta prvi in drugi del postopka povezana, da se v primeru vnosa obdelajo pravi podatki in ne kakšni iz nekega drugega obrazca.

Podobno kot že register.php pred njo ima tudi **user_editor.php** popolnoma enako strukturo. Edina razlika je, da gre tukaj za postopek, kjer spreminjamo obstoječega uporabnika in ne dodajamo novega. Posledično so uporabljene druge metode znotraj UserManager razreda.

V okviru tega podpoglavja velja omeniti še osnovne datoteke. Konkretneje, njihov glavni predel (glej Slika 13) vsebuje switch stavek, ki na osnovi dostopa, ki smo ga določili znotraj access_control.php, uporabniku omogoči ali pa onemogoči vpogled v določene predele uporabniškega vmesnika.

4.4.2 Upravljanje z datotekami (FileManager razred)

Koda za upravljanje z datotekami je razdeljena na dve primarni sekciji. Razred, poimenovan FileManager, se nahaja znotraj datoteke classes.php in vsebuje potrebne metode za ustvarjanje, brisanje in prikazovanje datotek ter map. Drugi del pa je celotna procedura, ki ustvari nov primerek omenjenega razreda in glede na to, za kakšen ukaz POST gre, sproži izvedbo specifične metode.

Proces je zapisan v datoteki file_manager.php. Tabela za prikazovanje datotek in map, sicer del FileManager razreda, se izvede ob ustreznem klicu ene izmed datotek AJAX.

FileManager je strukturiran tako, da se konstruktorju razreda poda korensko mapo, kjer želimo imeti shranjene vse datoteke. Nato konstruktor poskrbi, da se za vsakega uporabnika ustvari svoja mapa, če ta seveda še ne obstaja. Ime mape je enako uporabnikovi identifikacijski številki. Tako zagotovimo, da ima vsak uporabnik lastno mapo, kamor bo hranil podatke. Razred je tudi sprogramiran tako, da ta datoteka predstavlja izhodiščno korensko mapo za uporabnika in preko nje v višje stopnje map ne more priti preko spletnega vmesnika, lahko pa seveda odpira vse podmape. Metode razreda so vse javno dostopne znotraj vmesnika.

Kot že prej omenjeno, je glavna datoteka, ki skrbi za pravilno izvajanje zgornjega razreda, file_manager.php. Nahaja se znotraj vseh datotek AJAX, kar pomeni, da se bodo vsi prenosi in ukazi, povezani z njimi in urejanjem datotek, izvajali preko principa AJAX. Sama struktura datoteke je zelo preprosta. Najprej se ustvari nov primerek FileManager razreda, ki bo seveda aktivna tekom celotnega izvajanja procedure. Sledi niz *if* stavkov, kjer vsakič preverimo, ali je potrebno klicati in izvesti kakšno metodo iz prej omenjenega razreda. Proces med drugim opravi naslednje kontrole:

- Ali je treba kakšno datoteko zbrisati?
- Ali je treba kakšno datoteko odstraniti iz čakalnice v sistemu HTCondor?
- Ali je treba prenesti kakšne nove datoteke na strežniški disk?
- Ali je treba ustvariti novo .submit datoteko?
- Ali je treba kakšno datoteko poslati v izvajanje sistemu HTCondor?

Podatke dostavljamo preko zahtevkov POST in z njihovo pomočjo tudi preverjamo, kaj je treba izvesti. Opozoriti velja, da tretja in peta kontrola spadata tudi v okvir upravljanja s sistemom HTCondor, kar je tema naslednjega poglavja. Seveda si zaradi tesne povezanosti obeh sistemov nekatere procese med seboj delita.

4.4.3 Upravljanje s sistemom HTCondor (CondorManager razred)

Upravljanje s sistemom HTCondor pravzaprav zajemata dva koncepta, ki se nahajata v dveh ločenih razredih. Prvi je manipulacija oziroma upravljanje z datotekami, ki se izvajajo v sistemu HTCondor. To je že v prejšnjem podpoglavju omenjena metoda za pošiljanje datotek na izvedbo v visoko-prepustno okolje, ki je zapisana znotraj FileManager razreda. Za izvedbo te metode skrbi proces znotraj datoteke file_manager.php, ki pa ima spisano tudi proceduro za brisanje datotek, ki se izvajajo. Drugi koncept pa je zapisan znotraj CondorManager razreda, katerega primarna funkcija je grafično prikazovanje HTCondor oblaka v obliki tabel in omogočanje brisanja vnosov v njem.

Zadnji večji razred je, kot že omenjeno, namenjen prikazovanju podatkov iz sistema HTCondor. Vsebuje metode za generacijo tabel, številčenje strani in pa konstruktor. Slednji odigra pomembno vlogo, saj na osnovi vhodnih podatkov pripravi vrednosti, ki bodo potrebne za izris katere koli tabele. Izris se krmari s pomočjo funkcij JavaScript, saj so vse izrisane znotraj datotek, do katerih se dostopa preko principa AJAX. Te funkcije so zapisane kar medvrstično, da je razred laže prenosljiv in neodvisen od zunanjih virov.

Podobno kot pri upravljanju z datotekami se tudi tukaj primerek razreda izvede znotraj datotek AJAX, le da je tukaj koda zapisana neposredno v datoteke AJAX in ni vključena preko druge, zunanje datoteke. Postopek za izvedbo mora le pripraviti ustrezen seznam podatkov, ki jih bo vmesnik prikazal uporabniku. HTCondor ima priročno metodo zapisa podatkov v obliki XML, kar omogoča enostavno obdelavo podatkov. Ko so podatki obdelani, se pošljejo ustvarjenemu primerku CondorManager razreda in se s sklicem na ustrezne metode izrišejo tabele s podatki.

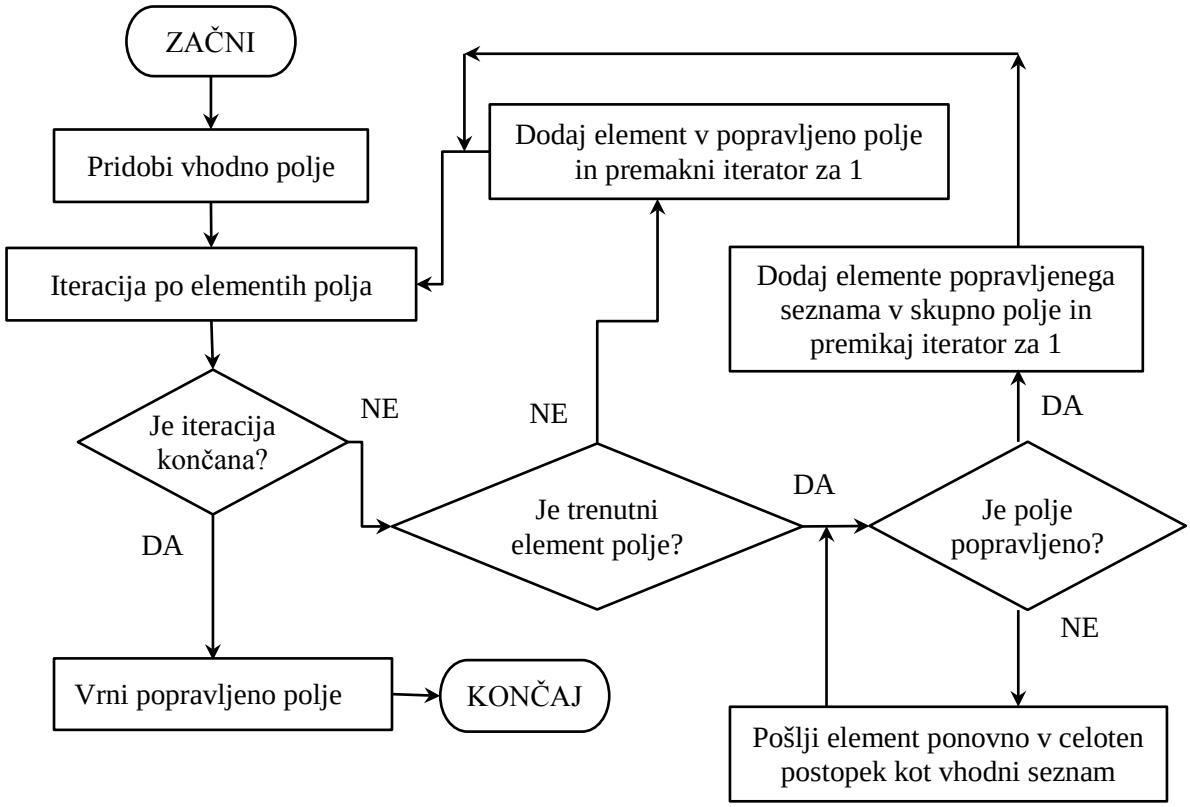
4.4.4 Sledenje statistiki (StatsManager razred)

Zadnji koncept, ki ima še spisan svoj razred za izvajanje, je sledenje statistiki. Gre za procese, ki zapišejo podatke o posameznih prošnjah odjemalca v bazo MySQL. Na primer stran, do katere je uporabnik dostopil, njegov IP ali pa uporabnikova identifikacijska številka. Iz te baze se lahko obdela podatke in ustvari zanimive grafe ali preglednice.

Razred za sledenje statistike, imenovan StatsManager, se od ostalih treh razlikuje po tem, da vsebuje podedovane metode in spremenljivke od razreda UserManager. Predvsem je pomemben konstruktor, ki ima zapisan proces za vzpostavitev povezave z ustrezno bazo podatkov in podano tabelo znotraj baze. Ostalih podedovanih metod StatsManager ne potrebuje, ima pa zato spisane tri lastne, vse javno dostopne metode, ki zapisujejo, berejo ali obdelujejo informacije iz baze podatkov.

4.4.5 Globalne funkcije

Vse splošne, globalne funkcije se nahajajo v datoteki functions.php. Ta datoteka je pripeta v vse ostale datoteke in tako omogoča uporabo funkcij preko celotnega projekta, brez potrebe po vnovičnem zapisu kode. Funkcije imajo opravka predvsem z obdelavo niza znakov, matrik in generičnih HTCondor ukazov. Primer algoritma ene izmed takih funkcij je prikazan na Slika 15.



Slika 15: Primer algoritma funkcije flattenArray()

4.4.6 Prikazovanje sporočil

Osnovni ideji pri zasnovi prikazovanja sporočil sta bili, da se vsa sporočila vse čas prikazujejo na nekem standardnem mestu in da bo znotraj kode, ne glede na datoteko, v kateri se nahaja, trivialno dodajati nova sporočila.

Ključno vlogo pri tem odigra globalna spremenljivka SESSION. To je ista spremenljivka za ustvarjanje sej, ki se uporablja tudi za shranjevanje podatkov o uporabniku, ki je trenutno prijavljen v sistem. Glavni lastnosti spremenljivke, ki ju ta koncept izkorišča, sta dejstvi, da je že vnaprej definirana ob izvedbi katere koli datoteke in ima standardizirano klicanje. Tako lahko v poljubni datoteki shranimo nek niz znakov, v tem primeru sporočilo, kot svoj element znotraj polja. Konkretno v tem vmesniku se vsa sporočila shranijo v obliki, prikazani na Slika 16, kjer »ime_napake« predstavlja identifikacijo sporočila, najpogosteje neko logično ime, »sporočilo« pa je sporočilo, ki se bo izpisalo uporabniku.

```
$_SESSION[˝custom_error˝][˝ime_napake˝] = ˝sporočilo˝;
```

Slika 16: Uporaba SESSION spremenljivke kot sredstvo za prenos sporočil

Seveda to samo po sebi še ni dovolj za delujoče prikazovanje sporočil. Potreben je še proces, ki vsa ta sporočila izpiše. Ta je definiran znotraj datoteke error_tracking.php. Gre za preprost postopek, ki

najprej obdela spremenljivko SESSION s funkcijo flattenArray() (glej podpoglavje Globalne funkcije), nato pa popravljeno polje z izbrisanimi dvojnimi vnosi zapiše znotraj posebnega elementa div HTML, vsako sporočilo v svojo vrsto. Omenjena datoteka error_tracking.php je vključena na koncu vsake datoteke AJAX in osnovne datoteke. Pri slednjih se nahaja znotraj noge.

Zgoraj opisani postopek pa še ne pomeni, da se bodo sporočila izpisala tam, kjer želimo. Zato je bilo treba izvesti še zadnjo stopnjo prikazovanja sporočil, ki pa je zapisana v globalni datoteki JavaScript. Tam se nahajata funkciji errorHandlerDesktop in errorHandlerMobile, ki skrbita za ustrezno prikazovanje sporočil na pravem mestu za namizno in mobilno različico vmesnika. Obe funkciji delujeta po istem principu, in sicer skrijeta zapis HTML s sporočili na koncu datoteke ter preneseta vsebino na ustrezno standardizirano mesto. Funkciji sta klicani po končani generaciji dokumenta HTML in po končanemu procesu AJAX. Ker proces poteka zelo hitro, uporabnik ne opazi zapisa na dnu datoteke, saj se pravočasno skrije.

V okviru tega poglavja velja omeniti še gumb, s katerim vklopimo in izklopimo prikazovanje sporočil. Gumb določi, ali se bodo elementi HTML znotraj datoteke error_tracking.php ustvarili ali ne. Koda, ki stoji za tem, je precej trivialna. Delno je zapisana tudi v header.php, kjer se omenjeni gumb nahaja.

4.4.7 Izris grafov s pChart

Izris grafov je neposredno povezan s poglavjem o sledenju statistiki, saj uporablja prav tisto tabelo znotraj baze podatkov, v katero zapisujemo s pomočjo StatsManager razreda. V tem razredu se nahajajo tudi metode za branje oziroma pridobivanje podatkov iz baze, kar koristimo ob izrisovanju grafikonov.

Grafi se izrisujejo s pomočjo knjižnice pChart. Kot že v teoretičnem delu omenjeno, gre za objektno orientirano knjižnico, ki deluje na osnovi privzete grafične knjižnice GD. Vsak graf ima svojo datoteko, kjer je zapisana procedura za generiranje ustrezne slike. Proces je v vseh primerih razdeljen na dva dela. Prvi del zajema pridobitev ustreznih podatkov iz baze s točno poizvedbo SQL in stvaritev polja, kamor pridobljene podatke shranimo. Drugi del pa te podatke prenese v primerek razreda pData in jih s pomočjo razreda pImage tudi izriše in ustvari sliko. Oba omenjena razreda sta del knjižnice pChart, kar nakazujeta tudi njuni imeni.

Zadnji korak je še prikaz slik znotraj vmesnika. To seveda opravi standardni element za slike HTML. Tako datoteke PHP s procedurami kot tudi elementi s slikami HTML se nahajajo znotraj osnovnih datotek. Natančneje, neposredno na mestu, kjer jih želimo videti prikazane.

4.4.8 Tabele MySQL

V podpoglavju o programskem zaledju sta omenjeni dve tabeli MySQL, ki ju koristi vmesnik. To sta tabeli *users* in *stats*.

Tabela users je namenjena hranjenju podatkov o uporabnikih. Večina teh je tipa celo število (int) ali pa niza poljubnih znakov (varchar). Primarni ključ je identifikacijska številka (userid), ki se tudi samodejno prišteva.

Tabela stats pa je namenjena sledenju uporabnikom oziroma njihovim ukazom in poizvedbam znotraj vmesnika. Tudi v tem primeru je večina zapisov ali tipa celo število ali pa niz poljubnih znakov, pri čemer pa ima dva ključna parametra. Primarni ključ je identifikacijska številka vnosa, ki se seveda samodejno prišteva in je pozitivno celo število, tuji ključ pa uporabniška identifikacijska številka, ki je enakovredna primarnemu ključu iz tabele users. Ukaz za stvaritev obeh tabel v jeziku SQL je prikazan na Slika 17.

```
CREATE TABLE users (
     userid int UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
     username varchar(100) NOT NULL,
     password varchar(42) NOT NULL,
     email varchar(100) NOT NULL,
     isadmin int(1),
     registertime varchar(100) NOT NULL,
     activetime varchar(100) NOT NULL
);

CREATE TABLE stats (
     statid int UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
     userid int UNSIGNED DEFAULT '0',
     browser varchar(255) NOT NULL DEFAULT '',
     ip varchar(15) NOT NULL DEFAULT '',
     date_visited int unsigned NOT NULL DEFAULT '0',
     page varchar(100) NOT NULL DEFAULT '',
     from_page varchar(150) NOT NULL DEFAULT '',
     submit_cluster int unsigned NOT NULL DEFAULT '0',
     submit_proc int unsigned NOT NULL DEFAULT '0',
     FOREIGN KEY (userid) REFERENCES users(userid)
);
```

Slika 17: Ukaz SQL za stvaritev users (zgoraj) in stats (spodaj) tabele

4.5 Čelni del vmesnika

Čelni del sistema ali obličje predstavlja vso kodo, ki je vidna uporabniku oziroma jo interpretira in poganja odjemalec. To vključuje kodo HTML ter pripadajoče datoteke CSS in JavaScript. Na kratko bi lahko rekli, da obličje vmesnika predstavlja ves vidni in interaktivni del vmesnika, preko katerega končni uporabnik komunicira s programskim zaledjem.

4.5.1 HyperText Markup Language

HTML v spletnem vmesniku skrbi za pravilno hierarhijo in identifikacijo elementov. Pravilna postavitev je v večji meri dosežena z gnezdenimi div elementi, ki se preko class in id atributov povežejo z ustreznimi slogi CSS. Atributa sicer uporablja tudi JavaScript za pravilno naslavljanje elementov.

Poleg div se od vidnih elementov pogosto pojavlja tudi table, a ne kot osnova za postavitev ostalih elementov, ampak za prikazovanje podatkov, kar je navsezadnje tudi njen namen. Slog je privzet Twitter bootstrap z zoženimi vrstami. Uporabljeni so tudi thead in tbody elementi, ki določijo glavo oziroma telo tabele. Preko tega se lahko definira poseben slog za vrsto z naslovi stolpcev.

V okviru form elementa za obrazce je uporabljeno več vrst input elementov:

- Text za vstavljanje besedila
- Button za gumb
- Submit za element, ki bo sprožil zahtevek obrazca
- File za prenašanje datotek
- Password za vstavljanje zakritih besedil
- Hidden za nevidne elemente
- Checkbox za potrditveno polje

Uporabljajo se po potrebi, na primer tip hidden se uporabi za vnose, kjer hočemo lepše grafično oblikovan gumb namesto privzetega, ki ima omejeno sposobnost spreminjanja.

4.5.2 Cascading Style Sheets

CSS ima največji vpliv na izgled uporabniškega vmesnika. Določa parametre, kot so dolžina in višina HTML elementov, njihove robove in vse lastnosti, povezane z njimi, tekstura ozadja, transparentnost elementov ali pa njihova sposobnost prilagajanja oblike glede na širino spletnega brskalnika.

Projekt v tej diplomski nalogi ima vse CSS datoteke shranjene znotraj css mape, in sicer so razdeljene v dve kategoriji. Ena kategorija predstavlja vse datoteke, povezane s Twitter Bootstrap knjižnico. Tukaj gre za vnaprej pripravljene stile, ki jih potem samo ustrezno kličemo znotraj HTML kode. Uporabljeni sta tako bootstrap.css, kot tudi bootstrap-responsive.css datoteki. Prva skrbi za splošni izgled in pravilno postavitev, druga pa ima spisana pravila za samodejno prilagajanje različnim širinam spletnega brskalnika, na primer tablicam ali pametnim telefonom.

Druga kategorija pa so ročno narejeni CSS stili, ki se nahajajo znotraj global_css.css datoteke. Vizualni podobi, ki sta tu zapisani, sta za generično obrobo z zavitimi robovi, ki jo najdemo v skoraj

vseh predelih vmesnika, in pa vsi potrebni slogi za izris pojavnega okna za prikazovanje sporočil. Nahaja se še nekaj stilskih sprememb, vezanih na dogodke premika miške, na primer sprememba barve teksta ali pa podčrtano besedilo ob lebdenju miške nad določenim predelom.

4.5.3 JavaScript

Tudi JavaScript, kot že pred njim CSS, ima ločeni datoteki za Bootsrap knjižnico ter splošno datoteko za kodo po meri, imenovano global_jquery.js. Obe datoteki se nahajata znotraj js mape, a nista edini, ki ju vmesnik uporablja. Poleg obeh se namreč preko spleta vključita še jquery-latest.js in jquery.form.js. Prva predstavlja najnovejšo jQuery knjižnico, ki precej poenostavi pisanje JavaScript kode in ima vgraviranih kar nekaj priročnih funkcij. Vključi se preko uradne jQuery spletne strani [24]. Druga datoteka pa je podaljšek jQuery knjižnice, ki omogoča prenašanje datotek preko AJAX principa, nekaj kar osnovna knjižnica privzeto ne dopušča. Tako se v vsak osnovni dokument vključijo štiri datoteke.

Postopek in funkcije za izvedbo potrebnih JavaScript ukazov se nahajajo znotraj global_jquery.js datoteke. Sestavljena je iz definiranih funkcij, ki se bodo uporabile znotraj drugega dela, kateri zajema glavne ukaze. Le ti se izvedejo po končani generaciji HTML kode in skupaj tvorijo celoten postopek izvedbe. Pri pisanju tako funkcij kot postopka se pretežno uporablja jQuery sintaksa, v redkih primerih pa tudi malce privzete JavaScript kode.

Vsi ukazi, ki predstavljajo postopek izvedbe celotne JavaScript kode in so nanizani en za drugim, imajo identično strukturo. Pokliče se jQuery funkcija .on(), ki na osnovi dogodka izvrši nek nov postopek (primer je prikazan na Slika 18). Ti novi postopki pa so lahko pošiljanje AJAX poizvedb, navigacija po straneh, predložitev obrazca, itd. Med ukazi je izstopajoča funkcija edino .ajaxComplete(), ki določi postopek, kateri se izvede po končanem AJAX prenosu.

```
$(document).on("click", "#home_file_button", function (){
     $("#home_file_upload").click();
});
```

Slika 18: Primer ukaza, ki se izvede ob kliku na element z ID atributom home_file_button

4.6 Videz vmesnika in uporabniška izkušnja

Za razliko od spletnih strani, katerih naloga je zgolj posredovanje informacij, gre pri izdelanem spletnem vmesniku za spletno aplikacijo. To pomeni, da bo vmesnik za uporabnika opravljal neko storitev, opravilo oziroma nalogo in posledično je potreba po interakciji visoka. Predvsem bo implementacija interakcije, poleg samega videza vmesnika, glavni vidik uporabniške izkušnje.

Pri povprečni interakciji s spletnimi stranmi se pojavljata predvsem dva problema. To sta neobstojnost spletnih strani ob osvežitvi, ko se vse vrednosti vrnejo na svoje privzete vrednosti, in osveževanje

celotne spletne strani, čeprav hočemo posodobiti le manjši del. Slednji pojav je, kljub zmogljivosti današnjih naprav, še zmeraj subtilno opazen. Če želimo ustvariti izkušnjo, podobno namiznim aplikacijam, je treba te probleme razrešiti, saj vplivajo na uporabnikovo dojemanje spletne strani ali spletne aplikacije.

Vmesnik ima skupaj dostopnih osem unikatnih osnovnih strani, preko katerih se pridobiva informacije ali pa komunicira z enim izmed sistemov, bodisi datotečnim ali HTCondor. Pet izmed osmih strani je dostopnih vsem uporabnikom in za njihovo prikazovanje ni treba pridobiti pravic. To so uvodna stran, stran s povezavami, predstavitvena stran, stran za registracijo in kot zadnje še stran, na kateri si lahko pogledamo različne informacije o trenutnem stanju sistema HTCondor. Nimamo pa dostopa do nobenih strani, kjer bo potekala interakcija. Za te je treba pridobiti ustrezen dostop. V nadaljevanju poglavja bosta podrobno opisana in prikazana videz ter uporabniška izkušnja za vsako izmed omenjenih strani, ki skupaj tvorijo spletno aplikacijo.

4.6.1 Uvodna stran

Uvodna stran je hranjena v datoteki index.php in zgolj prikazuje predstavitveno okno s sliko, navigacijskim menijem in gumbom do predstavitvene strani. Drugih funkcij ta spletna stran nima, prikazana pa je na Slika 19.

Slika 19: Uvodna index stran

4.6.2 Predstavitev

V tem podpoglavju je predstavljena vsa pomembna funkcionalnost spletne aplikacije. Tako s sliko kot tudi z opisom so predstavljeni vsi pomembni predeli spletnega vmesnika. Namen te strani je podati uporabniku neke vrste predogled opravil, ki jih bo lahko uporabljal, če se želi prijaviti.

Slika 20: Skrajšana predstavitev

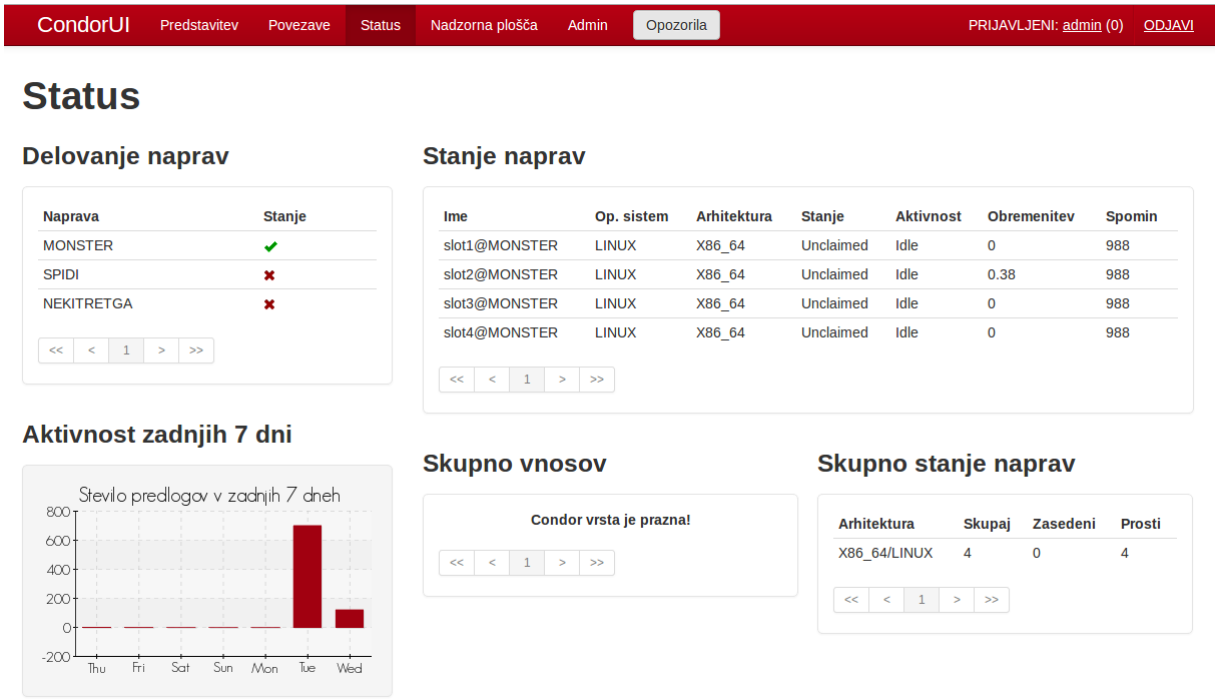
4.6.3 Povezave

Na strani s povezavami lahko uporabnik priročno dostopa do spletnih strani, ki podajo več informacij o tehnologijah, ki so bile uporabljene pri izdelavi tega vmesnika. Povezave so podane v obliki uradnih logotipov (Slika 21).

Slika 21: Stran s povezavami do spletnih tehnologij univerze in fakultete

4.6.4 Status

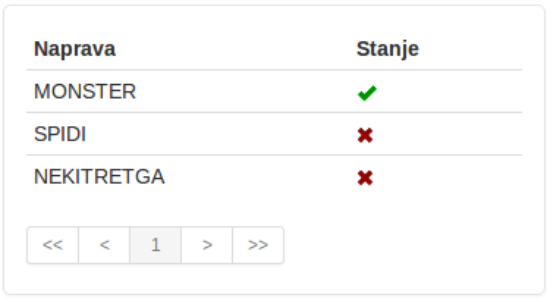
Prva večja stran, ki je tudi dostopna vsem in ne potrebuje posebnih pravic, je Status. Primarni namen te strani je pregled trenutnega stanja sistema HTCondor, pri čemer pa večina informacij še ni vezanih na posameznega uporabnika.



Slika 22: Status stran

Na Slika 22 je prikazan celoten pregled omenjene strani in podatkov, ki jih prikazuje. Kot je razvidno, je stran razdeljena na pet področij, ki so logično razdeljena glede na posredovane informacije. To so delovanje naprav, aktivnost v zadnjih sedmih dneh, stanje naprav, skupno število trenutnih vnosov HTCondor in stanje naprav na ravni arhitekture strojne opreme. Glavna značilnost v implementaciji, ki jo velja poudariti znotraj tega podoglavja, je ta, da so vsi posamezni odseki pridobljeni s pomočjo principa AJAX. To nam omogoča, da tudi navigacija strani poteka preko principa AJAX, kar izkušnjo naredi prijetnejšo za uporabnika.

Tabela Delovanje naprav prikaže vse naprave, ki so povezane v sistem HTCondor. Prav tako pokaže njihovo stanje, ali je naprava vklopljena (zelena kljukica) ali pa ugasnjena (rdeč križec). Podatke o vseh računalnikih vmesnik dobi iz znakovne datoteke, medtem ko podatke o delujočih napravah dobi kot povratno informacijo sistema HTCondor. Prikaže se največ 15 vnosov na stran, število strani pa ni omejeno.



Slika 23: Seznam delujočih naprav

Slika 24: Graf najnovejših vnosov

V področju Aktivnost zadnjih sedem dni se s pomočjo pChart izriše graf, ki prikaže število novih vnosov HTCondor, predloženih v zadnjih sedmih dneh. Predstavljeno je s pomočjo stolpčnega grafikona, pri čemer vsak stolpec zavzema obdobje 24 ur. Interakcije z grafom ni, celotno izrisovanje poteka samodejno.

Največji predel znotraj te strani podaja stanje naprav, to so podrobni podatki o napravah, ki so povezane in priklopljene v sistem HTCondor. Poleg naprave so informacije podane na jedro procesorja natančno, kar je označeno in oštevilčeno s slot1, slot2 itd. (Slika 25). Prikazani podatki o napravi oziroma njenih jedrih so operacijski sistem, arhitektura procesorskih jeder, njegova zasedenost, v kakšnem aktivnem stanju je, kolikšna je relativna obremenjenost in koliko notranjega spomina uporablja. Podobno kot že pri prejšnjem primeru se lahko na stran izpiše največ 15 naprav oziroma jeder, po straneh pa krmarimo z generičnim vmesnikom za straničenje enakega videza, ki se večkrat pojavi znotraj spletnega vmesnika.

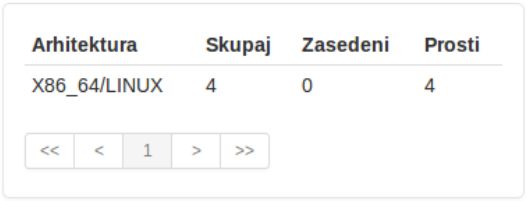
Ime	Op. sistem	Arhitektura	Stanje	Aktivnost	Obremenitev	Spomin
slot1@MONSTER	LINUX	X86_64	Unclaimed	Idle	0	988
slot2@MONSTER	LINUX	X86_64	Unclaimed	Idle	0.38	988
slot3@MONSTER	LINUX	X86_64	Unclaimed	Idle	0	988
slot4@MONSTER	LINUX	X86_64	Unclaimed	Idle	0	988

Slika 25: Stanje delujočih naprav na jedro procesorja natančno

Za vsakega uporabnika posebej je prikazano skupno število vnosov, ki jih ima ta uporabnik trenutno znotraj HTCondor vrste. Podrobnejših informacij iz te tabele še ne dobimo, saj je njen namen v osnovi prikazati trenutno zasedenost sistema HTCondor in kateri uporabniki so tisti, ki uporabljajo največ sredstev. Tudi tukaj najdemo standardni vmesnik za paginacijo.

Lastnik	Predložene dat.	Vsa opravila
admin	1	20

Slika 26: Slika prikazuje skupno število trenutnih vnosov za uporabnika admin



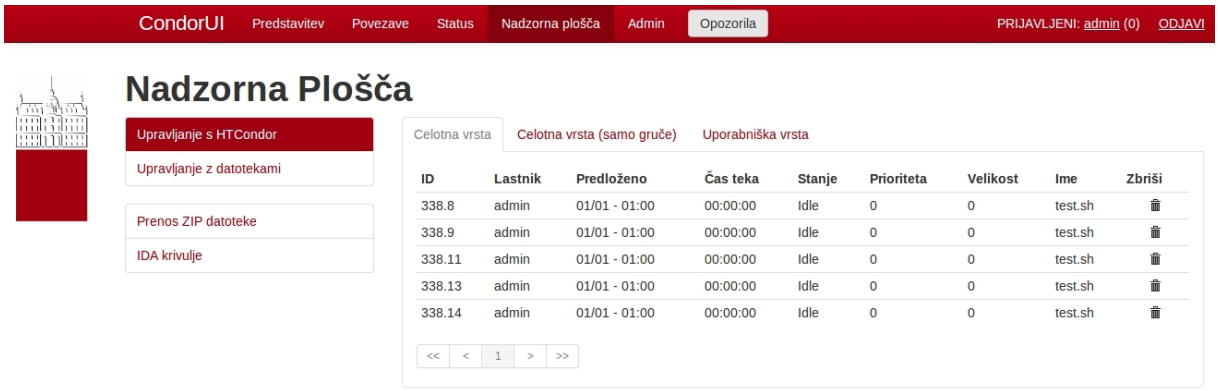
Slika 27: Prikaz skupnega stanja naprav

Zadnja tabela znotraj te sekcije prikazuje zasedenost naprav oziroma procesorskih jeder glede na arhitekturo procesorja. HTCondor namreč med drugim omogoča, da omejimo računanje problema samo na naprave z določeno arhitekturo. Taka tabela nam da vedeti, ali imamo proste želene naprave, ne da nam je treba brskati skozi podroben seznam vseh naprav in seštevati. Tudi tukaj se ustvari grafični vmesnik za listanje strani.

4.6.5 Nadzorna plošča

Nadzorna plošča predstavlja glavni del spletne aplikacije, preko katere se bo komuniciralo s sistemom HTCondor. Ta predel aplikacije je že zaščiten, kar pomeni, da moramo biti za njegovo uporabo registrirani in prijavljeni. Medtem ko se je v predelu Status v večji meri samo pregledovalo stanje sistema, bo v okviru nadzorne plošče možno dodajati nove predloge, brisati obstoječe, pogledati čakalno listo operacij, ustvarjati nove predložne datoteke in upravljati z datotečnim sistemom. Tukaj bodo na voljo tudi vse aplikacije, izdelane za specifična opravila.

Področje je razdeljeno na dva glavna dela. Na levi se nahajajo navigacijski gumbi, s katerimi izbiramo med različnimi možnostmi, večji del na desni strani pa bo prikazoval izbrano možnost (Slika 28). To prikazovanje poteka preko principa AJAX in nudi bolj zvezno uporabniško izkušnjo, saj spletne strani ni treba znova v celoti naložiti. Prav tako programsko zaledje skrbi, da si vmesnik zapomni izbrane možnosti navigacije, v primeru da uporabnik vseeno osveži spletno aplikacijo ali pa začasno zapusti ta del strani.



Slika 28: Privzet prikaz ob zagonu nadzorne plošče

Krmilni del je razdeljen na dva sklopa, od katerih vsak vsebuje po dva gumba. Zgornja gumba, Upravljanje s HTCondor in Upravljanje z datotekami, predstavljata splošni del nadzorne plošče, kjer se bo upravljalo vsa generična opravila z datotečnim in HTCondor sistemom. Spodnji del pa je namenjen aplikacijam za specifična opravila, ki so povezane z obema prej omenjenima sistemoma. Trenutno sta razviti le dve posebni aplikaciji, to sta aplikacija za samodejno razpakiranje ZIP-datotek in aplikacija za ustvarjanje predložnih datotek za račun IDA krivulj.

Slika 29: Navigacijski meni za nadzorno ploščo

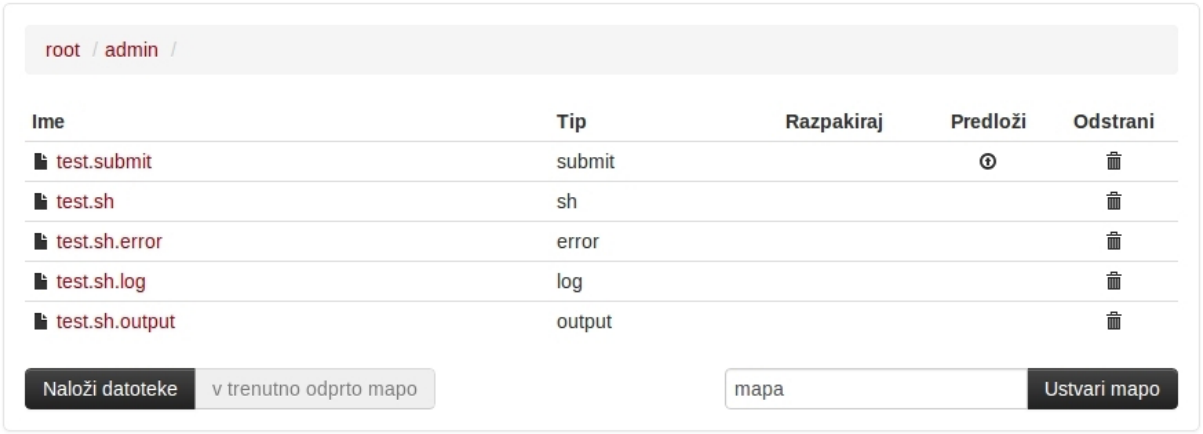
Prikazovanje vseh naprav, povezanih v okolje HTCondor, se nahaja že v okviru sekcije Status. Nismo pa še nikjer videli prikazanih opravil, ki so trenutno v izvajanju, in tistih, ki se še bodo izvedla. Za ta namen obstaja HTCondor vrsta, ki se nahaja pod menijem Upravljanje s HTCondor.

Slika 30: Vmesnik za upravljanje s HTCondor vnosi

Na voljo imamo tri različne poglede na vrsto (Slika 30), ki so dostopni preko klikov na zavihke. Celotna vrsta nam, kot že ime nakazuje, prikaže vrsto brez filtrov. Izpišejo se vsi procesi vseh uporabnikov. Za vsak proces je na voljo več podatkov, kot na primer njen lastnik, čas poteka opravila, njegovo trenutno stanje itd. Ker zna biti seznam kar dolg, je število vnosov omejeno na 15 na stran, za krmarjenje strani pa skrbi že znani generični vmesnik za straničenje. Naslednja možnost, ki je na voljo, je prikaz gruč procesov. To pomeni, da so vidne samo gruče opravil, ne pa tudi posamezni procesi znotraj njih. Zadnja možnost nam po strukturi prikaže enake rezultate kot prva, le da so filtrirani vsi uporabniki z izjemo prijavljenega.

Tabele za prikazovanje HTCondor vrste pa imajo še eno funkcionalnost, in sicer odstranjevanje vnosov. Za to obstaja gumb oblike koša, ki je načeloma viden samo zraven procesov prijavljenega uporabnika. Edina izjema so administratorski računi, ki lahko brišejo katere koli vnose, tudi vnose drugih uporabnikov.

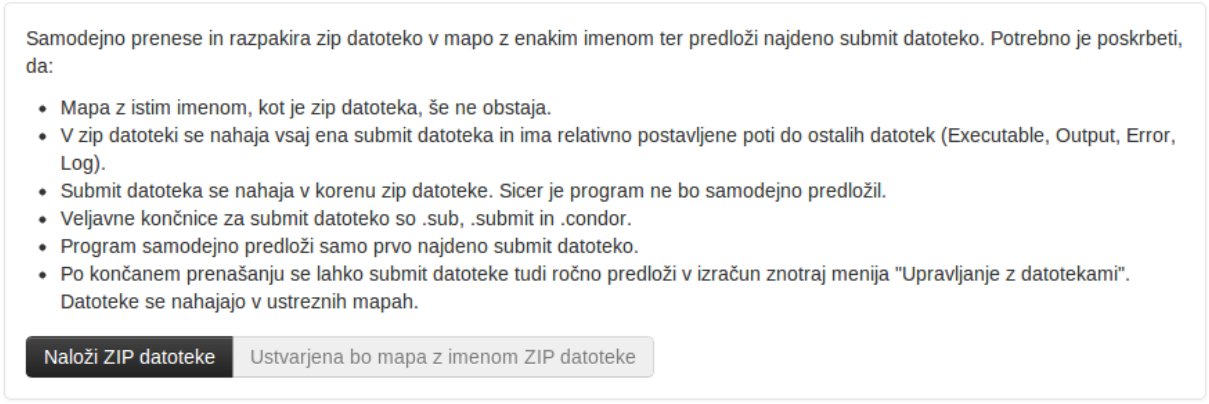
Vmesnik za upravljanje z datotekami deluje po principu AJAX in omogoča kar nekaj funkcionalnosti. Primarna je gotovo snemanje izhodnih datotek, ki jih proizvede HTCondor ob uspešnem zaključku računanja. Poleg tega omogoča tudi nalaganje novih datotek, katere lahko pošljemo v izvajanje, če vsebujejo pravilne končnice. Podprte so .submit, .sub in .condor. Poleg predložnih datotek imajo posebno funkcionalnost omogočeno tudi datoteke s končnico zip, katere lahko s pritiskom na ustrezni gumb razpakiramo. Zaradi potrebe po boljši organizaciji je omogočeno tudi ustvarjanje novih map. Uporabnik tako lahko brez težav loči projekte med seboj. Ne nazadnje pa je omogočeno tudi brisanje datotek in map, da uporabniku ni treba skrbeti za odvečne datoteke, ki jih več ne potrebuje. Po mapah se krmari preko t. i. breadcrumbs krmil na skrajni zgornji strani. Vse opisano se da videti na Slika 31.



Slika 31: Upravitelj z datotekami

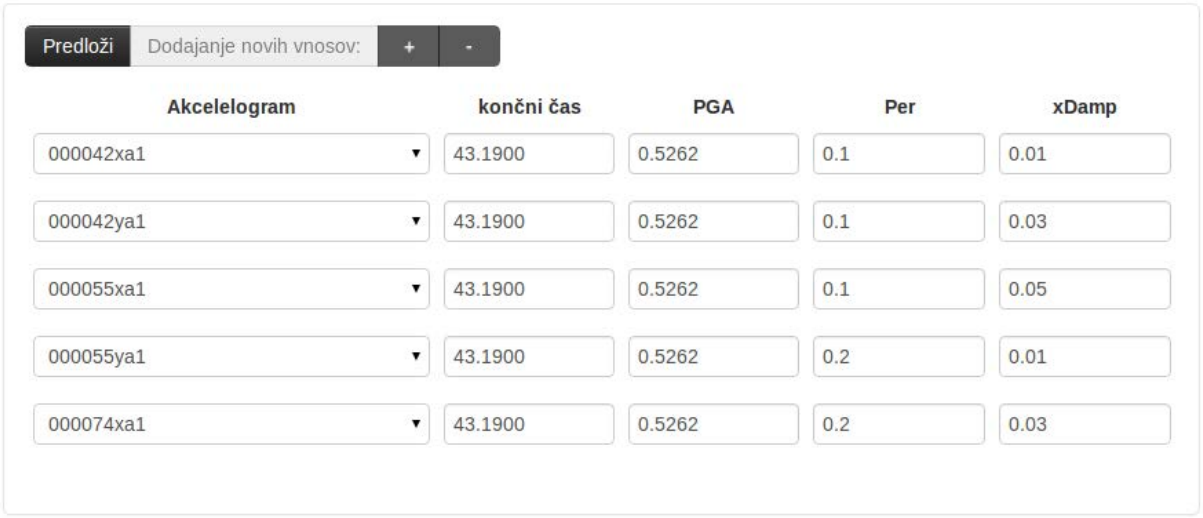
Velja še omeniti, da se vsa navigacija in nalaganje izvaja znotraj korenske mape uporabnika, ki je generirana avtomatsko. Programsko zaledje poskrbi, da uporabnik izven te datoteke ne more upravljati preko tega vmesnika.

Prva izmed dveh izdelanih aplikacij je namenjena samodejnemu razpakiranju naloženih ZIP-datotek in se nahaja pod menijem Prenos ZIP datoteke (**Error! Reference source not found.**). Ta funkcionalnost je sicer mogoča že znotraj upravitelja z datotekami, a je v okviru te aplikacije celoten proces samodejen in ne potrebuje dodatnega vložka s strani uporabnika. Vse, kar uporabnik naredi, je, da naloži ZIP-datoteko, aplikacija pa jo sama razpakira v mapo z enakim imenom ter pošlje v izračun prvo predložno datoteko, ki se v njej nahaja.



Slika 32: Videz aplikacije za nalaganje ZIP-datotek

Druga izdelana aplikacija je malce kompleksnejša od prve, ki je predstavljala le združitev že obstoječih metod. Za računanje IDA krivulj je bilo namreč treba ustvariti nove funkcije za samodejno ustvarjanje predložne datoteke. Ta se ustvari na osnovi parametrov, kot so izbira akcelelograma in končni čas računanja, katere poda uporabnik. Poleg tega mora biti omogočeno podajanje poljubnega števila primerov. Vse to je izvedeno s pomočjo principa AJAX in JavaScript kode.

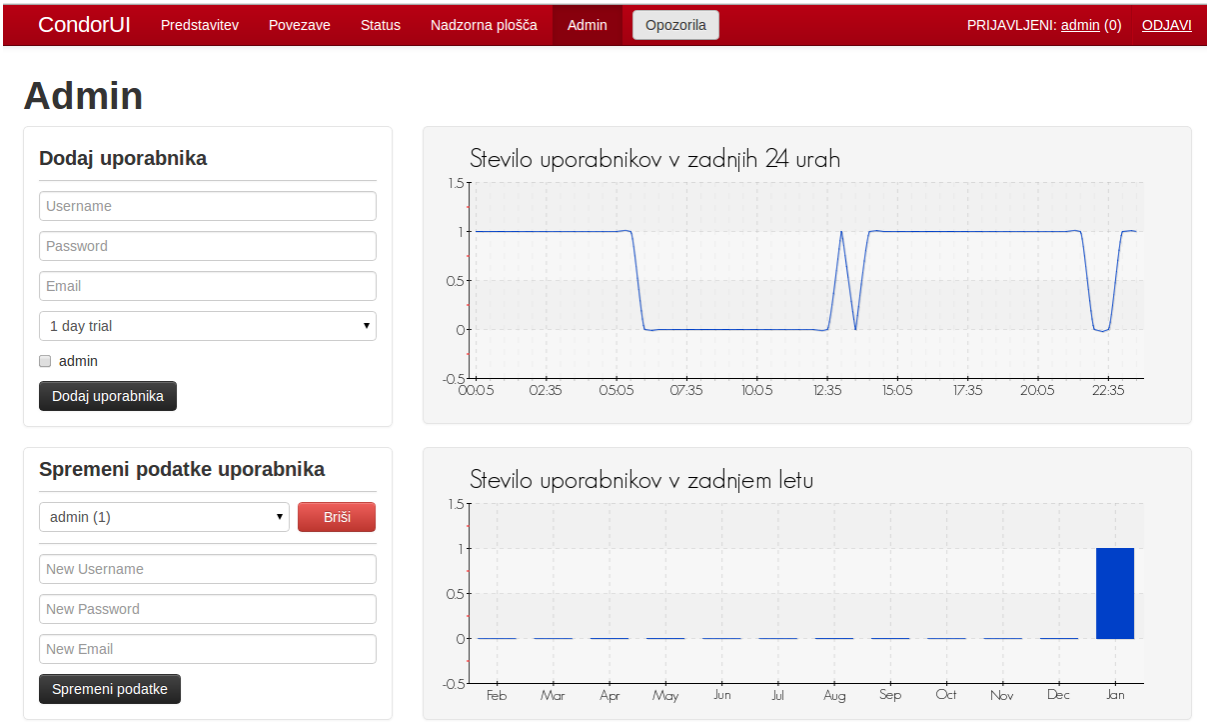


Slika 33: Vmesnik za računanje IDA krivulj

S pritiskom na gumb + ali – se dodaja oziroma odvzema število vnosov v predložno datoteko. Vsak tak vnos sicer predstavlja svoj proces znotraj sistema HTCondor, če pride do predložitve. Izbira parametrov se vnese ročno preko besedilnega polja, le akcelelogram izberemo iz vnaprej pripravljenega seznama, ki se ustvari na osnovi akcelelogramov, ki so na voljo. Aplikacija nato ustvari ustrezno predložno datoteko, jo pošlje v izvajanje, hkrati pa prekopira vse potrebne datoteke za izvedbo v mapo idacurves, kamor bodo prišli tudi rezultati. Uporabnik do njih dostopi preko prej omenjenega upravitelja z datotekami.

4.6.6 Stran za administratorje

Gre za posebno stran, ki bo vidna samo uporabnikom z administratorskimi pravicami. V večji meri gre za vmesnik, ki omogoča urejanje uporabniških računov, kot je na primer ustvarjanje novih administratorjev, brisanje uporabnikov, spreminjanje podatkov itd.



Slika 34: Stran za administratorje, odrezana zaradi večje kompaktnosti

Stran omogoča tudi dostop do nekaj zanimivih statističnih podatkov, ki so prikazani s pomočjo grafov. Gre za informacije o uporabnikih, na primer število prijavljenih uporabnikov po različnih mesecih v letu ali pa odstotek najbolj obiskanih strani.

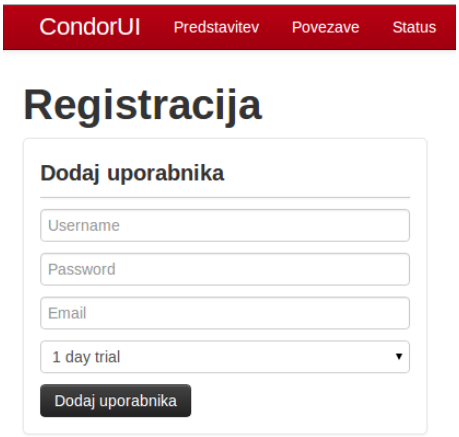
4.6.7 Profil uporabnika

Profil omogoča uporabnikom nekaj osnovnih operacij na svojem računu, kot so spreminjanje uporabniškega imena, gesla ali elektronske pošte. Uporabnik lahko tudi preveri, na katerih straneh vmesnika se je največkrat zadrževal.

Slika 35: Primer profilne strani

4.6.8 Stran za registracijo

Verjetno najpreprostejša stran je stran za registracijo. Vse, kar vsebuje, je obrazec za stvaritev novega uporabnika, kamor vpišemo uporabniško ime, geslo, elektronsko pošto in izberemo čas aktivnosti računa. Seveda programsko zaledje preveri, če uporabniško ime ali elektronski poštni naslov že obstajata, in po potrebi zavrne kreacijo novega računa.



Slika 36: Obrazec za registracijo

4.7 Scenariji uporabe

Če sta bila v prejšnjem poglavju opisana videz in uporabniška izkušnja na splošno, bo to krajše poglavje namenjeno opisu treh različnih uporab vmesnika, povezanih s pošiljanjem predložnih datotek v sistem HTCondor. Dve izmed teh uporab imata tudi posebej zase izdelana spletna vmesnika znotraj te aplikacije.

4.7.1 Generična uporaba

Prva uporaba je najbolj splošna in omogoča, da predložimo v izračun katero koli izvršljivo datoteko in kolikor jih hočemo. Izvaja se preko upravitelja z datotekami, kamor ročno naložimo vse potrebne

datoteke za izračun. Paziti moramo na pravilno strukturo map, kakor je zapisano v predložni datoteki, ki jo moramo ustvariti sami. Če smo kot končnico predložne datoteke izbrali eno izmed podprtih možnosti (sub, submit ali condor), se nam izriše gumb in posledično lahko predložimo program v izračun.

4.7.2 Pošiljanje ZIP-datoteke

Druga možnost poteka je preko vmesnika za ZIP-datoteke. Glavna prednost tega sistema pred prvim je ta, da nam ni treba nalagati več datotek vsako posebej, ampak lahko strukturo map in potrebne datoteke razporedimo že znotraj operacijskega sistema. Za večino bo to dobrodošlo, saj so tega navajeni in učinkoviteje datoteke in mape urejajo v znanem okolju. Pri razvrščanju datotek moramo paziti tudi na to, da se bo predložna datoteka nahajala v korenu ZIP-datotek. Ko je enkrat struktura datotek narejena, se jih stisne v ZIP-datoteko in pošlje na izvajanje. Vmesnik sam poskrbi, da se datoteke ustrezno razširijo v mapo z imenom ZIP-datoteke, znotraj te mape pa datoteke ohranijo svojo hierarhijo. Glavna omejitev takega načina pošiljanja je nezmožnost predlaganja več kot ene predložne datoteke hkrati. Sicer tudi v tem primeru lahko pošiljamo poljubne izvršljive datoteke.

4.7.3 Računanje IDA krivulj

Zadnja možnost za pošiljanje programov v izračun je uporaba vmesnika za IDA krivulje. Gre za specifičen vmesnik, ki ustvari vnaprej pripravljeno predložno datoteko. Uporabnik ne more izbirati izvršnih datotek, saj so te že vnaprej določene, a mu po drugi strani ni treba izdelati predložne datoteke, ker se ta ustvari sama. Interakcija uporabnika z vmesnikom je omejena na izbiro akcelelograma in nekaj parametrov, na osnovi katerih se samodejno ustvarijo in pošljejo v izvedbo potrebne datoteke. Vmesnik ni uporaben v splošne namene, ampak zgolj za računanje podatkov, povezanih z IDA krivuljami.

5 ZAKLJUČEK

Okolje HTCondor, ki je vzpostavljeno na Fakulteti za gradbeništvo in geodezijo, se uporablja za optimalno rabo prostih naprav v računske namene. Poleg tega imajo na fakulteti postavljene še dodatne računalnike specifično za nudenje računske moči. Vendar pa se je izkazalo, da ima sistem HTCondor dve glavni slabosti. Prva je neprijazen uporabniški vmesnik, ki kot privzet poteka preko terminalnega okna, druga slabost pa je zahteva, da mora biti naprava, iz katere je poslana predložna datoteka, ves čas računanja prižgana. Predlagana rešitev je bila ustvariti spletni vmesnik, ki bo hkrati razrešil oba problema. Ker pa sem bil še popolnoma neizkušen v programiranju spletnih aplikacij, je proces izdelave verjetno trajal precej dlje kot bi sicer. Soočal sem se z mnogimi problemi, ki se mi sedaj zdijo precej trivialni, takrat pa so mi povzročali precej preglavic. A mi je na koncu vendarle uspelo ustvariti spletni vmesnik, za katerega upam, da ga bodo na fakulteti tudi uporabili.

5.1 Ugotovitve in sklepi

Med izdelovanjem naloge sem ugotovil, da se spletno programiranje v marsičem razlikuje od klasičnega. Primarna razlika je ta, da načina grafičnega prikazovanja ne moramo izbirati, ampak je standardiziran preko jezika HTML in slogov CSS. Prav tako je večji del programskega zaledja namenjen zgolj generaciji kode HTML, medtem ko JavaScript skrbi predvsem za interakcijo in omogočanje principa AJAX.

Spoznal sem tudi, kako pomembno je planiranje aplikacije že od samega začetka. Čeprav sem vedel, da bom zaradi meni osebno bolj prijetne uporabniške izkušnje kar precej uporabljal princip AJAX, vseeno nisem že v prvem poizkusu dovolj dobro zasnoval strukture datotek in postavitve elementov znotraj datotek, ki bi mi omogočila enostavno integracijo strani po principu AJAX. Kmalu sem ugotovil, da si na primer ob osveževanju zaradi uporabe principa AJAX aplikacija ne zapomni menijev, ki so bili pred tem označeni.

Glede sistema HTCondor sem ugotovil, da gre za sistem, ki optimizira rabo velikega števila virov, ne izvleče pa dodatne moči iz posameznih naprav. Kot so pokazale analize, lahko HTCondor doseže tudi do 98 % učinkovitost. Ker imajo na Fakulteti za gradbeništvo in geodezijo kar precej računalnikov, je bilo cenovno smiselno izrabiti njihovo računsko moč, preostanek potrebe pa nadomestiti z namensko gručo naprav.

Izkusil sem tudi veliko stvari, ki niso neposredno povezane z izdelavo vmesnika. Ker sem aplikacijo razvijal na operacijskem sistemu Linux, sem se bil primoran seznaniti z njim. Tudi HTCondor, MySQL in git kot privzete delujejo s pošiljanjem ukazov preko konzolnega okna. To je koncept, ki mi je precej tuj in sem ga pred tem redko uporabljal. To spoznanje je bilo sicer kar pomembno, saj sem še toliko bolj cenil pomembnost in potrebo po grafičnih vmesnikih za povprečne uporabnike.

Gledano v celoti je bila izdelava vmesnika odlična izkušnja. Veliko sem se naučil in ni mi uspelo doseči samo cilja potrebe po spletnem vmesniku, ampak sem dosegel oziroma celo presegel tudi lastne cilje po pridobitvi izkušenj in znanja v spletnem programiranju. Upam, da mi bo še kdaj koristilo v življenju, saj sem pri izdelavi, kljub občasnim frustracijam, neizmerno užival.

5.2 Smernice za nadaljnji razvoj

Ker ima vmesnik vso splošno oziroma osnovno funkcionalnost za komuniciranje s sistemom HTCondor že implementirano, bi nadaljnji razvoj aplikacije potekal predvsem v smeri implementacije novih specifičnih vmesnikov, kot je ta za IDA krivulje. Verjetno še kaj povezanega s potresnim inženirstvom ali bolj s splošno dinamičnimi analizami, saj dodana dimenzija čas ustvarja potrebo po večji računski moči. Lahko se implementira tudi več statističnih podatkov, na primer graf, ki prikazuje porabo povezanih naprav skozi čas. Ker so osnove že sprogramirane, bi bilo treba le nadgraditi že obstoječo kodo. Še eno področje, kateremu se nisem preveč posvečal, je varnost. Verjetno baza podatkov ni optimalno zaščitena, zato bi bilo dobro, če bi se bolj izkušeni programerji kdaj posvetili tudi temu.

VIRI

UPORABLJENI VIRI

[1] Berman, F., Fox, G., Hey, T. 2003. Grid Computing. Making the Global Infrastructure a Reality. Chichester, John Wiley & Sons Ltd: 1012 str.

[2] HTCondor. 2013. High Throughput Computing (HTC). http://research.cs.wisc.edu/htcondor/htc.html (Pridobljeno 22. 1. 2013.)

[3] World Wide Web Consortium. 1999. Introduction to HTML 4. What is HTML?. http://www.w3.org/TR/html4/intro/intro.html#h-2.2 (Pridobljeno 22. 1. 2013.)

[4] PHP: Hypertext Preprocessor. 2013. PHP Manual. General Information. http://si1.php.net/manual/en/faq.general.php (Pridobljeno 22. 1. 2013.)

[5] w3schools. 2013. SQL Basic. Introduction to SQL. http://www.w3schools.com/sql/sql_intro.asp (Pridobljeno 22. 1. 2013.)

[6] The Apache Software Foundation. 2013. Apache HTTP Server Project. About Apache. http://httpd.apache.org/ABOUT_APACHE.html (Pridobljeno 22. 1. 2013.)

[7] Ullman, C., Dykes, L. 2007. Beginning Ajax. Indianapolis, Wiley Publishing, Inc.: 498 str. http://www.wrox.com/WileyCDA/Section/id-303217.html (Pridobljeno 22. 1. 2013.)

[8] jQuery. 2013. jQuery Project. http://jquery.com/ (Pridobljeno 22. 1. 2013.)

[9] Alsup, M. 2013. jQuery Form Plugin. http://malsup.com/jquery/form/ (Pridobljeno 22. 1. 2013.)

[10] Twitter Bootstrap. 2013. Introducing Bootstrap. http://twitter.github.com/bootstrap/index.html (Pridobljeno 23. 1. 2013.)

[11] pChart. 2013. What can pChart do for you?. http://www.pchart.net/ (Pridobljeno 23. 1. 2013.)

[12] HTCondor. 2013. What is HTCondor?. http://research.cs.wisc.edu/htcondor/description.html (Pridobljeno 23. 1. 2013.)

[13] Dolenc, M. 2011. Grid Technology for Engineers. Condor Overview. http://media.matevzdolenc.com/hpc4e-2011/hpc4e-dolenc-07-condor.pdf (Pridobljeno 5. 2. 2013.)

[14] Dolšek, M. 2011. Protection of Built Environments Against Earthquakes. Ljubljana, Springer: 338 str.

[15] Interoperability of Virtual Organizations on a Complex Semantic Grid (InteliGrid). 2013. http://inteligrid.eu-project.info (Pridobljeno 5. 2. 2013.)

[16] Data Mining Tools and Services for Grid Computing Environments (DataMiningGrid). 2013. http://www.datamininggrid.org (Pridobljeno 5. 2. 2013.)

[17] Zevnik, J. 2007. Potresna ranljivost armiranobetonskih viaduktov s škatlastimi stebri. Doktorska disertacija. Ljubljana, Fakulteta za gradbeništvo in geodezijo (samozaložba J. Zevnik): 182 str.

[18] Visoko-propustno računsko okolje za analize potresnega tveganja (ICE4RISK). 2013. http://ice4risk.slo-projekt.info (Pridobljeno 1. 7. 2013.)

[19] Github. 2013. CondorUI. NewUI. https://github.com/mihahren/CondorUI/tree/NewUI (Pridobljeno 1. 7. 2013.)

[20] Warm Forest. 2013 Folder Structure and Project Organization Best Practices. http://www.warmforestflash.com/blog/2009/10/folder-structure-and-project-organization-best-practices/ (Pridobljeno 1. 7. 2013.)

[21] Nettuts+. 2013. Organize Your Next PHP Project the Right Way. http://net.tutsplus.com/tutorials/php/organize-your-next-php-project-the-right-way/ (Pridobljeno 1. 7. 2013.)

[22] Stavkoverflow. 2013. Simple PHP question about menu & content best practice. http://stackoverflow.com/questions/4997350/simple-php-question-about-menu-content-best-practice (Pridobljeno 1. 7. 2013.)

[23] Wikipedia. 2013. Post/Redirect/Get. http://en.wikipedia.org/wiki/Post/Redirect/Get (Pridobljeno 1. 7. 2013.)

[24] jQuery. 2013. jQuery Latest. http://code.jquery.com/jquery-latest.js (Pridobljeno 1. 7. 2013.)

OSTALI VIRI

Zaveršnik, M. 2013. Programski jezik JavaScript. Ljubljana, Univerza v Ljubljani, Fakulteta za matematiko in fiziko.
http://zaversnik.fmf.uni-lj.si/gradiva/JavaScript/ (Pridobljeno 22. 1. 2013.)

W3Schools. 2013. PHP MySQL Introduction.
http://www.w3schools.com/php/php_mysql_intro.asp (Pridobljeno 22. 1. 2013.)