Tallinna Ülikool

Informaatika Instituut

Programmeerimisoskuste hindamise veebikeskkond

Bakalaureusetöö

Autor: Karmo Rosental

Juhendaja: Jaagup Kippar

Autor: ..... „ ..... „ 2013

Juhendaja: ..... „ ..... „ 2013

Instituudi direktor: ..... „ ..... „ 2013

Tallinn 2013

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

Kuupäev: ..... Autor: .....

Sisukord

Sissejuhatus

Programmeerimisülesannete hindamine käsitsi on palju aega nõudev töö. Lisaks sellele saab ülesande lahendaja tavaliselt tagasisidet oma lahenduse kohta alles hulk aega peale ülesannetega lõpetamist. See juhtub siis, kui hindaja on ülesanded läbi vaadanud ja otsustanud hinded avalikustada. Kas ja kuidas saaks seda protsessi kiiremaks muuta? Siinkohal on abiks automatiseeritud hindamissüsteem, mis lahendaja käsu peale annab mistahes ajahetkel teada tema lahenduskäigu edenemisest ja kus süsteem hindab ülesandeid kohe, kui nende lahendamiseks määratud aeg on täis saanud. Nõnda hoitakse kokku hulga käsitsi ülesannete hindamisele kuluvaid töötunde ja jäävad ära inimlikud vead hindamisel. Seejuures saab kasutaja võimalusel lahendada ülesandeid just talle sobival ajal ja kohas.

Otsustasin kirjutada valitud teemal, kuna tekkis huvi, et kuidas on võimalik automaattestidega testida sama funksionaalsusega koodi erinevates programmeerimiskeeltes nii, et ei peaks automaatteste kõikide erinevate keelte jaoks eraldi kirjutama. Edasi tekkis juba idee arendada selle probleemi lahendusele tuginedes veebikeskkond, kus programmeerimisülesannete lahendusi testitakse selliste automaattestidega, mis on kirjeldatud sõltumata lahendaja valitud programmeerimiskeelest. Sellise veebikeskkonna kasutajateks oleksid õpilased, programmeerimisalasele tööle kandideerijad või lihtsalt programmeerimishuvilised, kes sooviksid oma teadmisi proovile panna. Teadmiste hindamise muudaks selline süsteem kiiremaks ja täpsemaks.

Antud bakalaureusetöö eesmärgiks on arendada veebikeskkond, kus kasutajad saavad lahendada neile meelepärases programmeerimiskeeles programmeerimisülesandeid, mida hinnatakse automaattestide abiga. Eesmärgi saavutamiseks pannakse kõigepealt paika arendatava tarkvara nõuded. Selle põhjal kavandatakse tarkvara arhitektuur. Kavandi järgi arendatakse tarkvara ja lõpus antakse hinnang tehtud arendustööle. Tarkvara aluseks on komponent, mis kompileerib ning hindab kasutaja poolt kirja pandud koodi. Antud rakendus on ka lahendus eelmises lõigus kirjeldatud põhiprobleemile. Selle komponendi ümber luuakse uusimaid tehnoloogiaid kasutades intuitiivne klient-server rakendus, mis võimaldab veebikeskkonnas programmeerimisülesandeid lahendada, kasutades koodi kompileerimiseks ja käivitamiseks serveri ressursse. Valmis tarkvara juurde kuuluvad ka kaks

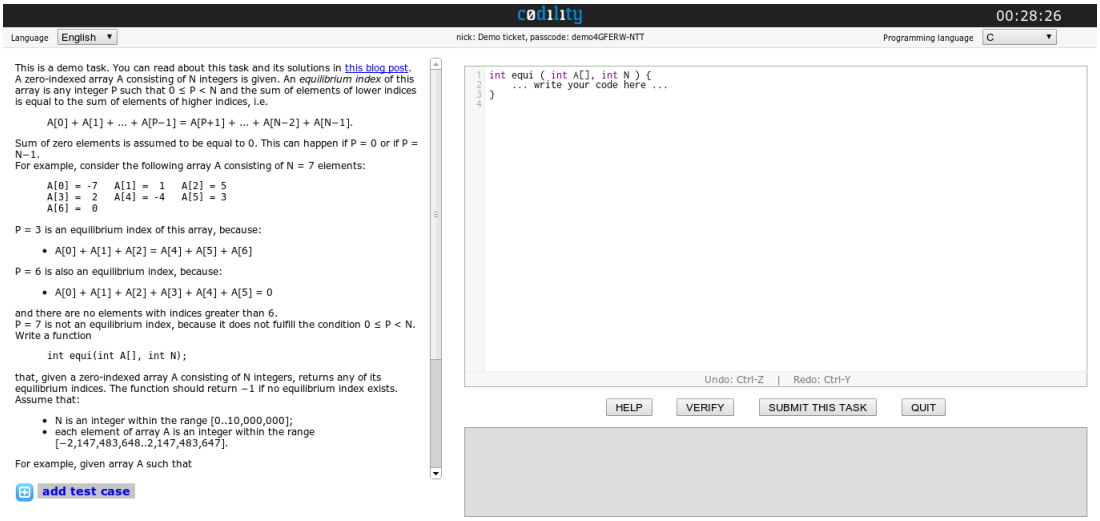
objekt-orienteeritud programmeerimisele keskendunud ülesannet ning nende võimalikud lahendused C++, PHP ja Pythoni programmeerimiskeeltes.

1. Sarnased lahendused

1.1. Codility

Codility (joonis 1) on keskkond programmeerimisülesannete automatiseeritud hindamiseks, mis on pühendunud aitamaks ettevõtetel värvata paremaid tarkvara arendajaid. Codility juured ulatuvad 2005. aastasse, kui selle looja Grzegorz Jakacki, töötades Hiina tarkvaraarenduse firmas Exoweb, lõi uute töötajate intervjueerimisele kuluva aja kokkuhoidmiseks automatiseeritud hindamissüsteemi. Selle eesmärk oli filtreerida välja nõrgemad kandidaadid. Pärast 2500 kandidaadi testimist, hoiti kokku vähemalt 4000 töötundi, mis muidu oleks kulunud nende kandidaatide hindamiseks, kellel on ebapiisavad programmeerimisoskused. Codility sai hoo sisse neli aastat hiljem, pärast osalemist Seedcamp Week 2009 idufirmade konkursil, kus selle meeskond jõudis kuue parima sekka.

Codilitys on toetatud 11 programmeerimiskeelt, milledeks on C, C++, C#, Java, JavaScript, Pascal, Perl, PHP, Ruby, Python ja VB.NET. Codility on tasuline teenus, millel on ka tasuta kasutumise võimalus, kuid see piirdub vaid ühe ainsa ülesandega. Ülesanded on püstitatud nii, et lahenduseks on argumentidega funktsioon, mis lahendab mingisuguse matemaatilise probleemi. Reeglitest on suhteliselt lihtne aru saada ning nendesse süüvimine ei nõua palju aega. Näidisülesanne on lahendatav aadressil *http://codility.com/demo/take-sample-test*. (Codility Ltd., 2013)



Joonis 1. Codility.

1.2. CodeEval

CodeEval (joonis 2) on platvorm, kus tarkvara arendajad saavad üksteisega võistelda auhindadele või hoopis tööpakkujatele oma teadmistega silma jääda. Tööpakkujad saavad kasutada CodeEval keskkonda, et luua programmeerimisülesannete lahendamise konkursse ja leida pakutavatele positsioonidele häid kandidaate. Keskkontoris on võimalik luua pakutavaid töökohti, programmeerimiskonkursse ja kutsuda e-posti teel kandidaate ülesandeid lahendama või hoopis inimesi kandidaatide poolt esitatud lahendusi kaasarvustama.

CodeEvalis on toetatud 13 programmeerimiskeelt, milledeks on C, C++, C#, Clojure, Java, JavaScript, Objective-C, Perl, PHP, Python, Ruby, Scala ja Tcl. Kasutajad saavad laadida oma lahendusi üles failina või kasutada kirjutamiseks veebiredaktorit. Esitatud koodid käivitatakse UNIX-tüüpi masinas. Kui tööpakkuja perspektiivist vaadatuna on CodeEvali kasutamine väga mugav, siis ülesannete lahendajate jaoks on võrreldes Codilityga siin koodi vormistamise reeglistik keeruliseks aetud. Kasutaja hooleks on jäetud kinni pidamine mallidest, mis erinevatel keeltel on kohati erinevad ja ei kuulu lahenduskäigu juurde. (CodeEval Inc., 2013)

Joonis 2. CodeEval.

1.3. CodeChef

CodeChef (joonis 3) on globaalne programmeerijatele mõeldud kommuun, kus toimuvad võistlused ja saab nendeks ka harjutada. CodeChef on mittetulunduslik hariduslik algatus, mida juhib India tarkvaraettevõte Directi. Iga kuu algul toimub seal suurem programmeerimisvõistlus ning kuu keskel väiksem võistlus. Kommuuni kuulub ka foorum, blogi ja wiki. 2013. aasta veebruariks on CodeChef kommuuniga liitunud umbes 25 000 kasutajat üle maailma.

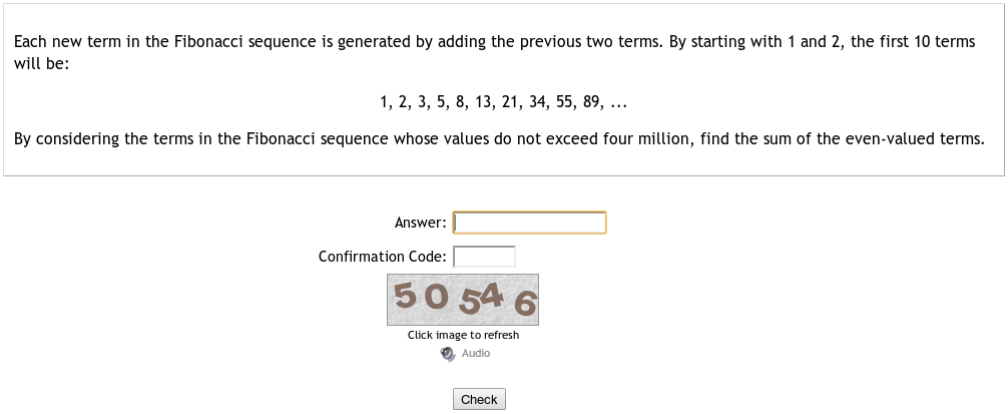
Reeglistik, kuidas peaks koodi vormistama, et see süsteemis valideeruks, on iga ülesande juures detailselt ja arusaadavalt kirja pandud. Lahendaja peab vaid jälgima seda, kuidas peab programmi andmed sisse lugema ja kuidas tulemusi väljastada. Programmi sisendid ja väljundid on sõltumata lahendaja poolt valitud keelest alati samas formaadis. Ülesannete lahendamine on seetõttu võrreldes CodeEval keskkonnaga oluliselt mugavam. (Directi Group, 2013)

Joonis 3. CodeChef.

1.4. Project Euler

Project Euler (joonis 4) on veebileht, kus on kogumik matemaatika ülesannetest. Ülesannete lahendamiseks on enamasti vaja lisaks matemaatikale ja loogikale ka programmeerimisoskust. Kogumikus on üle 400 erineva raskusastmega ülesande, millede lahendamisel on võimalik tasemetel edasi liikuda, just nagu videomängudes. Lehekülg on suunatud eelkõige matemaatika ja informaatika erialade õpilastele, kes soovivad lisaks õppekavale veel mujalt teadmisi omandada. Samas ka kõikidele teistele, kes soovivad oma matemaatilisi oskusi teravdada.

Võrreldes eelneva kolme veebilehega, erineb Project Euler kõige rohkem arendatavast veebikeskkonnast. Project Euler ei paku koodiredaktorit ega mingisugust automaatset koodi hindamist. Programmeerimiskeele valik on jäetud täiesti vabaks. Project Euleris on iga ülesande all vaid tekstikast, kuhu tuleb õige vastus kirjutada. Peale õige vastuse sisestamist avaneb postituste lõim, kus saab konkreetse ülesandega seoses teiste kasutajatega mõtteid ja lahendusi vahetada. (Project Euler, 2013)



Joonis 4. Project Euler.

2. Nõuded

Käesolevas peatükis on kirja pandud hulk erinevaid funktsionaalseid ja mittefunktsionaalseid nõudeid, millele peaks valmiv tarkvara vastama. Nõuded on saadud lähtuvalt autori nägemusest loodavast tarkvarast.

2.1. Funktsionaalsed nõuded

1. Kasutajakonto loomine

Kirjeldus: Kasutaja sisestab andmed ja saab registreeritud kasutajaks.

Eeltingimused: Puuduvad.

Põhivoog:

- Kasutaja vajutab linki "Registreeri"
- Kasutaja täidab nõutud väljad
- Kasutaja vajutab nuppu "Registreeri"

Järeltingimused: Kasutajakonto on loodud ja kasutaja võib nüüd sisse logida.

2. Kasutaja sisse logimine

Kirjeldus: Kasutaja autoriseerib ennast kasutajanime ja parooliga.

Eeltingimused: Kasutaja on eelnevalt loonud omale kasutajakonto.

Põhivoog:

- Kasutaja vajutab linki "Logi sisse"
- Kasutaja sisestab oma kasutajanime ja parooli
- Kasutaja vajutab nuppu "Logi sisse"

Järeltingimused: Kasutaja on sisse logitud ja võib nüüd hakata teste sirvima ja lahendama.

3. Kasutaja välja logimine

Kirjeldus: Sisse logitud kasutaja logib ennast süsteemist välja.

Eeltingimused: Kasutaja on sisse logitud ja asub avalehel või testide sirvimise lehel.

Põhivoog:

- Kasutaja vajutab linki "Logi välja"

Järeltingimused: Kasutaja on välja logitud ja ei saa teste sirvida ega lahendada.

4. Testide sirvimine

Kirjeldus: Kõikide antud ajahetkel aktiivsete (ajalised piirangud pole ületatud) testide sirvimine.

Eeltingimused: Kasutaja on sisse logitud ja asub avalehel.

Põhivoog:

- Kasutaja vajutab linki "Testid"

Järeltingimused: Kasutajale kuvatakse loend hetkel aktiivsetest testidest.

5. Oma testide sirvimine

Kirjeldus: Enda alustatud või juba lõpetatud testide sirvimine.

Eeltingimused: Kasutaja on sisse logitud ja asub avalehel.

Põhivoog:

- Kasutaja vajutab linki "Testid"
- Kasutaja vajutab linki "Minu testid"

Järeltingimused: Kasutajale kuvatakse loend testidest, mille lahendamist ta on alustanud või mille lahendamise juba lõpetanud.

6. Ülesande valimine

Kirjeldus: Kasutaja valib ülesannete loendist ülesande, millega ta hakkab tegelema.

Eeltingimused: Kasutaja on lahendamas testi.

Põhivoog:

- Kasutaja valib rippmenüüst sobiva ülesande

Järeltingimused: Kasutajale kuvatakse ülesande tekst ning ülesandega seotud failide loend. Endise aktiivse ülesandega seotud failid salvestatakse automaatselt andmebaasi.

7. Faili loomine

Kirjeldus: Kasutaja loob uue faili, millesse ta hakkab kirjutama ülesannet lahendavat koodi.

Eeltingimused: Kasutaja on lahendamas testi ja tal on valitud ülesanne millega ta parasjagu tegeleb.

Põhivoog:

- Kasutaja vajutab nupul "Lisa fail"
- Kasutaja sisestab avanenud aknasse faili nime ja valib faili tüübi
- Kasutaja vajutab nupul "OK"

Järeltingimused: Failide loendivaatesse tekkis juurde loodud fail ning seda kuvatakse ka faili detailvaates koheseks koodi kirjutamiseks.

8. Faili ümber nimetamine

Kirjeldus: Kasutaja muudab faili nime põhi- või laiendiosa.

Eeltingimused: Kasutaja on lahendamas testi ja tal on valitud ülesande jaoks loodud vähemalt üks fail.

Põhivoog:

- Kasutaja vajutab paremat hiireklahvi kui kursor on faili nimel
- Kasutaja valib avanenud menüüst "Redigeeri"
- Kasutaja sisestab failile uue nime
- Kasutaja vajutab nupul "OK"

Järeltingimused: Faili nime ja laiendiosa on selline nagu kasutaja selle muutis.

9. Faili kustutamine

Kirjeldus: Kasutaja kustutab ülesandest faili, mida ta ülesande lahendusse enam ei soovi.

Eeltingimused: Kasutaja on lahendamas testi ja tal on valitud ülesande jaoks loodud vähemalt üks fail.

Põhivoog:

- Kasutaja vajutab paremat hiireklahvi kui kursor on faili nimel
- Kasutaja valib avanenud menüüst "Kustuta"
- Kasutaja vajutab avanenud kastis "OK"

Järeltingimused: Kustutatud fail on eemaldatud valitud ülesande failide loendist.

10. Faili valimine

Kirjeldus: Kasutaja valib failide loendist faili, millega ta soovib hakata tegelema.

Eeltingimused: Kasutaja on lahendamas testi ja tal on valitud ülesande jaoks loodud vähemalt kaks faili.

Põhivoog:

- Kasutaja vajutab failil, mis hetkel pole detailvaates.

Järeltingimused: Faili, millel kasutaja vajutas, kuvatakse nüüd detailvaates ja sellesse on võimalik koodi kirjutada.

11. Koodi kirjutamine

Kirjeldus: Kasutaja kirjutab valitud programmeerimiskeeles koodi.

Eeltingimused: Kasutaja on lahendamas testi ja tal on faili detailvaates lahti mingi fail. Faili detailvaade on samal ajal ka failiredaktor.

Põhivoog:

- Kasutaja kirjutab koodiredaktorisse koodi.

Järeltingimused: Faili detailvaatesse ilmub kasutaja kirjutatud kood.

12. Koodi käivitamine

Kirjeldus: Kasutaja käivitab koodi ja kontrollib, kas tema lahenduskäik on korrektne.

Eeltingimused: Kasutaja on lahendamas testi ja tal on failide loendivaates vähemalt üks fail.

Põhivoog:

- Kasutaja vajutab nuppu "Kävita"

Järeltingimused: Enne käivitamist failid salvestatakse automaatselt andmebaasi. Kasutajale kuvatakse, millistes olukordades kood valideerub ja millistes mitte.

13. Testi lahendamiseks järele jäänud aja kuvamine

Kirjeldus: Kasutaja näeb ekraanil aega, mis on testi lahendamiseks järele jäänud.

Eeltingimused: Kasutaja on lahendamas testi, mis on aktiivne.

Põhivoog: Puudub.

Järeltingimused: Puuduvad.

14. Testist väljumine

Kirjeldus: Kasutaja väljub testist, mille lahendamist saab jätkata seni, kuni testi ajalised piirangud pole ületatud.

Eeltingimused: Kasutaja on lahendamas testi.

Põhivoog:

- Kasutaja vajutab nuppu "Välju"

Järeltingimused: Valitud ülesandega seotud failid salvestatakse automaatselt andmebaasi. Kasutaja suunatakse testide loendilehele.

2.2. Mittefunktsionaalsed nõuded

Dokumentatsioon

- Tarkvara peab olema nii arendajate kui lõppkasutajate jaoks dokumenteeritud

Kasutatavus

- Tarkvara peab olema lihtsa ja loogilise kasutajaliidesega
- Tarkvara kasutajaliides peab olema kergesti tõlgitav
- Koodi redaktoris peab olema äramärgitud süntaks

Hallatavus

- Kõik süsteemist tulenevad vead tuleb logida
- Tarkvara konfiguratsiooni peab olema lihtne muuta

Jõudlus

- Koodi kompileerimine ja käivitamine koos võrgu viitajaga ei tohiks kokku aega võtta rohkem kui 5 sekundit

Kvaliteet

- Tarkvara ei tohi sisaldada ühtegi kriitilist viga
- Kõik tarkvarale kirjutatud unit testid peavad alati läbi minema

Laiendatavus

- Uusi, ülesannete lahendamisel kasutatavaid programmeerimiskeeli, peab arendajatel olema lihtne lisada

Ligipääsetavus

- Tarkvara peab olema kasutatav kõikide levinumate personaalarvutite ja mobiilsete seadmete internetilehitsejates

Privaatsus

- Üks kasutaja ei tohi näha teise kasutaja testide tulemusi

Sõltuvus

- Tarkvara võib sõltuda vaid vaba tarkvara pakettidest ja moodulitest võimalikult dünaamilisel viisil

Testitavus

- Tarkvara peab olema täielikult testitav ka arenduskeskkonnas
- Tarkvara koodile peavad olema kirjutatud unit testid

Turvalisus

- Testide lahendamiseks peavad kasutajad ennast autentima

Ühilduvus

- Ülesandeid testivad automaattestid peavad olema süsteemis kirjeldatud nii, et neist saab tuletada automaatteste kõikide levinumate programmeerimiskeelte jaoks
- Tarkvara peab võimaldama kasutajal lahendada ülesandeid järgnevates programmeerimiskeeltes ja nende versioonides: C++11 (GCC 4.7), Java SE 7, PHP 5.4 ja Python 3.2

3. Kavandamine

3.1. Üldine arhitektuur

Kogu süsteemi platvormiks saab Ubuntu. Ubuntu on populaarne tasuta töölaua- ja serveri operatsioonisüsteem. Ubuntu on Linuxi distributsioon, mis põhineb Debian GNU/Linuxil ja kasutab ka selle pakihaldust. Vaikimisi Ubuntu paigaldus sisaldab ainult vaba tarkvara. Valik langes Ubuntule, kuna sellel on suur kasutajaskond ja suur paketivaramu, kus on küllaltki värsked versioonid tarkvara pakettidest. Kasutusele võetakse Ubuntu versioon 12.10. (Canonical Ltd, 2013)

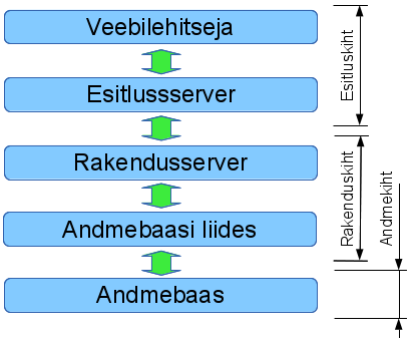
Tarkvara hakatakse arendama Django raamistikule. Django on avatud lähtekoodiga vaba tarkvaraline mudel-vaade-kontroller põhimõttel töötav veebiraamistik kiireks veebirakenduste arendamiseks. Django on kirjutatud programmeerimiskeeles Python ja töötab levinumates operatsioonisüsteemides. Mõned olulisemad Django võimalused on dünaamiline andmebaasiliides, URL dispetšer, mallisüsteem, lokaliseerimine ja internatsionaliseerimine, automaatne administreerimiskeskkonna genereerimine, vahemälu võimalused jne. Kasutusele võetakse Django versioon 1.4, mis uuendatakse peale esimese beta väljatulekut Django 1.5 peale. Django 1.5 soovitab kasutada Pythoni versiooni 2.7. Python 3 toetus on veel eksperimentaalne ja peaks saama stabiilseks järgmises või ülejärgmises Django versioonis. (Django Software Foundation, 2013)

Kuna Django raamistikuga kaasas olev server ei sobi oma halva jõudluse ja skaleeruvuse tõttu toodangusse, siis kasutatakse serveerimiseks võimast Twisted veebiserverit. Twisted on kirjutatud samuti Pythonis, mis muudab tema integreerimise Djangoga lihtsaks. Django raamistik ja Twisted server suhtlevad omavahel läbi WSGI *(Web Server Gateway Interface)* liidese. (Twisted Matrix Labs, 2013)

Andmebaasina võetakse kasutusele PostgreSQL. PostgreSQL on avatud lähtekoodiga vaba tarkvaraline objekt-relatsiooniline andmebaasisüsteem, mis on võimeline töötama kõikides levinumates operatsioonisüsteemides. Psycopg hakkab toimima liidesena PostgreSQL andmebaasi ja programmeerimiskeele Python vahel. (PostgreSQL Global Development Group, 2013; Varrazzo, 2012)

Kui vaadelda loodava tarkvara arhitektuuri kõige üldisemalt, siis see jaguneb kolmeks eraldiseisvaks füüsiliseks kihiks (joonis 5). Teisiti saaks arhitektuuri jagada viieks loogiliseks kihiks (joonis 6). Kõige alumine kiht on PostgreSQL andmebaas. Teine kiht on Psycopg andmebaasi liides. Kolmas kiht on rakendusserver, mis tegeleb äriloogikaga. Neljas kiht on esitlusserver, mis saadab välja HTML märgendikeelt. Viies ja kõige pealmine kiht on kliendi veebilehitseja, mis kuvab visuaalse pildi HTML märgendikeelest, JavaScriptist, CSS laadilehtedest ja rastergraafikast.

Joonis 5. Füüsilised kihid.

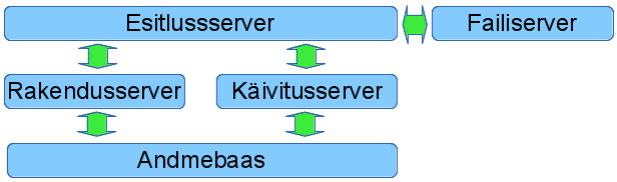


Joonis 6. Loogilised kihid.

N-kihilisel arhitektuuril on mitmeid kasulikke omadusi. Erinevaid kihte on võimalik korraga kasutada mitmetes erinevates rakendustes. Loogilisi kihte saab jagada füüsilisteks kihtideks, mis tihti parandavad jõudlust, skaleeruvust ja vigadele vastupanuvõimet. Koodis orienteerumine, hooldamine ja funktsionaalsuse lisamine on lihtsam. Rakendus on paremini testitav kuna erinevaid kihte on võimalik eraldi testida. Ilma koodi loogilisteks kihtideks jagamata, tuleks lahendada kõik turvalisusega seotud probleemid ühes kohas koos ning see teeks nende lahendamise keerukamaks. (Coster, 2010)

Servereid on plaanis luua viis (joonis 7). Nendeks saavad olema andmebaasiserver, rakendusserver, server testide kopileerimiseks ja käivitamiseks (mida edaspidi nimetatakse

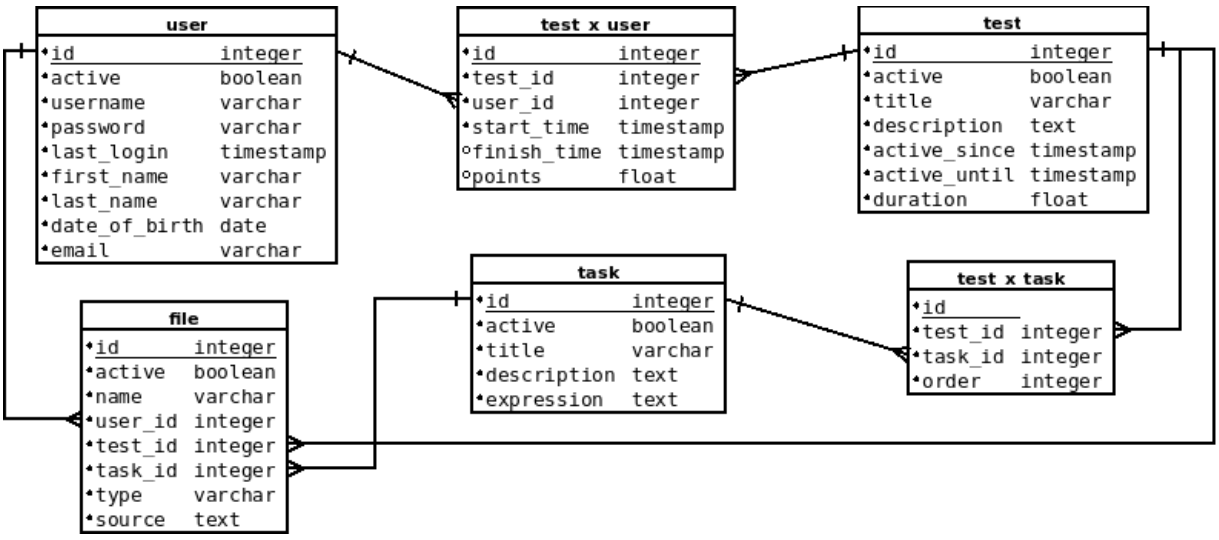
käivitusserveriks), esitlusserver ning server staatiliste failide (CSS laadilehed, JavaScripti failid, meediafailid jne) serveerimiseks (mida edaspidi nimetatakse failiserveriks).



Joonis 7. Serverid.

3.2. Andmebaas

Enne arenduse algust sai valmis ka andmebaasi skeem (joonis 8). Kasutajad hoitakse tabelis *user*. See tabel asendab Django *django_auth_user* tabelit, kuna viimases pole tabelile lubatud lisada juurde uusi veerge. Ülesandeid, mida testides kasutada, hoitakse tabelis *task* ning teste ise tabelis *test*. Kasutajaid seob testidega tabel *test_x_user*, kus on ka veerud testi alustamise ja lõpetamise aja ning punktide arvu kohta. Ülesanded liidetakse testidega *test_x_task* tabeliga, kus on lisaveeruks number, mis tähistab ülesande järjekorda testis. Tabelis *file* hoitakse faile, mida kasutajad loovad ülesannete lahendamisel. Iga fail on seotud ühe kindla kasutajaga, testiga ja ülesandega.



Joonis 8. Andmebaasi skeem.

3.3. Rakendusserver

Rakendusserveri poole võidakse pöörduda siis, kui on vaja andmebaasist midagi lugeda, lisada, muuta või kustutada. Rakendusserveri kood toetub Django ning Django-Tastypie raamistikele. Django-Tastypie lisab Djangole REST *(Representational State Transfer)* liidese kasutamise võimaluse. Nagu juba eespool sai mainitud, siis päringud andmebaasi toimuvad läbi Psycopg liidese. (Lindsley, 2013)

Rakendusserveri poole tuleb pöörduda ainult REST-stiilis päringutega üle HTTP protokolli. REST arhitektuuril on olemiks ressurss. Igal ressursil on oma aadress. Näiteks ressursil *users* on kollektsiooni aadress /users/ ja ühe kindla kasutaja objekti aadress /users/1/, kus 1 tähistab kasutaja ID-d. Ressursi poole pöördutakse meetoditega GET, POST, PUT või DELETE, sõltuvalt sellest kas tahetakse ressursist midagi lugeda, sinna lisada, sealt muuta või kustutada.

Vaikimisi on veebilehitsejates XMLHttpRequest päringud ühest domeenist teise keelatud. Antud juhul taoline päring tehtaks esitlusserveri domeeni pealt JavaScriptist rakendusserveri domeeni peale. Selleks implementeeritakse rakendusserverisse CORS *(Cross-origin resource sharing)*, mis lubab ressursse serveerida teisest domeenist tulevatele päringutele. (Hossain & Hausenblas, 2013)

Autoriseerimiseks peavad HTTP päringud rakendusserverisse sisaldama päises sisse logitud kasutaja sessioonivõtit. Turvalisuse huvides tuleks rakendusserverisse ära keelata kõik ühendused peale nende, mis tulevad esitlusserverist, kuid selleks peavad XMLHttpRequest päringud minema läbi esitlusserveri või spetsiaalselt nende jaoks loodud proksi, mis jääb aga hetkel arenduse ulatusest välja.

Rakendusserverisse sisenevad ja väljuvad andmed on serialiseeritud JSON formaadis. Pythoni objektid serialiseeritakse Python-CJSON mooduli abil. Python-CJSON on kirjutatud C keeles ning on kuni 250 korda kiirem Pythoni implementatsioonidest. (Pascu, 2007)

3.4. Käivitusserver

Käivitusserver on antud bakalaureusetöö raames arendatava tarkvara kõige olulisem server. See vastutab kasutajate kirjutatud koodi kompileerimise, käivitamise ja testimise eest. Server ise nimetatud toimingutega ei tegele, küll aga koostab kasutaja kirjutatud koodi jaoks automaattestid.

Kasutaja kirjutatud koodi ja selle jaoks koostatud automaattestide kompileerimisega tegelevad vastava programmeerimiskeele arendusvahendid. Käivitusserver vaid käivitab Linuxi käsuprotsessoris vastava programmeerimiskeele kompilaatori, mis kasutaja koodi ja automaattestid kompileerib ja käivitab ning salvestab omale kompileerimisel ja käivitamisel saadud väljundi. Seejärel tagastatakse kasutajale see sama kompileerimisel ja käivitamise saadud väljund. Kasutaja näeb, kas tema kood kompileeris, ja kui kompileeris, siis millised automaattestid ei läinud läbi ja proovib neid parandada.

Kui vaadelda erinevaid programmeerimiskeeli, siis nende kirjapildis võib leida palju sarnaseid mustreid. Kattuvad mustrid saab panna baasklassi(desse) ja kõik see, mis on erinev, läheb klassi, mis tegeleb juba konkreetselt mingi kindla programmeerimiskeelega. Kattuvad mustrid on näiteks klasside, funktsioonide ja meetodite definitsioonid, muutujate definitsioonid, andmetüübid, teekide importimine jne. Küsimus on vaid selles, et milline on õige süntaks näiteks klassi defineerimiseks mingis kindlas programmeerimiskeeles.

Käivitusserver hakkab testima koodi GET-päringu peale. Näiteks päring GET /1/?task=2 käivitusserverisse testib koodi, mis on kirjutatud sisse logitud kasutaja poolt testile, mille ID andmebaasis on 1, ja ülesandele, millele ID andmebaasis on 2. Esimese sammuna küsib käivitusserver andmebaasist ülesandele lisatud failide andmed ja loob need ajutiselt serveri kettale /tmp kataloogi. Seejärel koostab tarkvara failidele automaattestid vastavalt sellele, missuguses programmerimiskeeles on failid kirjutatud. Kui programmeerimiskeel nõuab lähtekoodi kompileerimist, siis kompileeritakse failid koos automaattestidega. Järgmine samm on automaattestide käivitamine ja väljundi saatmine kasutajale.

Siinkohal on sobilik tuua välja ka arhitektuurilised erinevused arendatava tarkvara ja sarnaste lahenduste peatükis kirjeldatud süsteemide vahel. Arendatavas tarkvaras on

automaattestimiseks igas keeles kasutusel mõni selle keele unit test raamistik ning automaattestid ise luuakse kasutaja kirjutatud koodi jaoks automaatselt. Võimalikke viise tulemuste hindamiseks on üsna palju (võimaluste hulk sõltub suuremalt jaolt unit test raamistiku võimalustest). Arendatavas tarkvaras on võimalik testida ka objekt-orienteeritud koodi ning koodi, mis on kirjutatud mitmesse faili. Lahendajale antakse rohkem vabadust koodi kirjutamiseks ja raamid ei ole nii kitsad. Uusi programmeerimiskeeli lisades tuleb vaid installeerida vastava keele käitusteek ja unit test raamistitik ning luua keele klass, kus tuleb defineerida mõned keele süntaksi ja keele unit test raamistiku kasutamise iseärasused.

3.5. Esitlusserver

Esitlusserveri ülesanne on kuvada rakendusserverist saadud andmed kasutajale loetaval kujul. Esitlusserveri kood toetub samuti Django raamistikule. Rakendusserverisse mugavamaks päringute tegemiseks kasutab esitlusserver Pythoni standardmoobulil httplib põhinevat *singleton* mustriga klassi *API* (vt koodinäide 1).

```
from common.api import get_api
api = get_api()
test = api.get('/tests/%s/' % test_id)
```

Koodinäide 1. Päringud rakendusserverisse.

Esitlusserveri ja rakendusserveri vahel liiguvad andmeobjektid ja -kollektsioonid serialiseeritud JSON formaadis. Esitlusserveri koodis hoitakse neid aga Pythoni assotsiatiivse massiivina. Kasutaja veebilehitsejasse jõuavad andmeobjektid ja -kollektsioonid nii, et nad esitatakse HTML märgendikeeles, kasutades selleks Django mallisüsteemi.

Kuna lehekülg, kus toimub testide lahendamine, peab olema interaktiivne ja suhtlema rakendusserveriga asünkroonnselt, siis seal võetakse kasutusele Backbone.js JavaScripti teek. Backbone.js on mudel-vaade-esitaja põhimõttel töötav liides *RESTful* rakendusliidestele. Backbone.js teegile tuginevas koodis defineeritakse vastavalt rakendusliidesele mudelid ja kollektsioonid ning vaated, mis mallide abil mudelid ja kollektsioonid veebilehitsejas nähtavaks teevad. Kõik tegevused on võimalik automatiseerida sündmusetöötlejatega (ingl. k. *event handler*). Lisaks Backbone.js teegile läheb veel tarvis Underscore.js teeki, millest sõltub Backbone.js ja mis pakub hulga väikseid abifunktsioone.

jQuery teek teeb Backbone.js vaadete programmeerimise oma elementide selekteerijaga mugavamaks ja võimaldab kasutada pluginaid näiteks küpsistega töötamiseks ja parema hiireklahvi menüü loomiseks. (DocumentCloud, 2013a/2013b; jQuery Foundation, 2013)

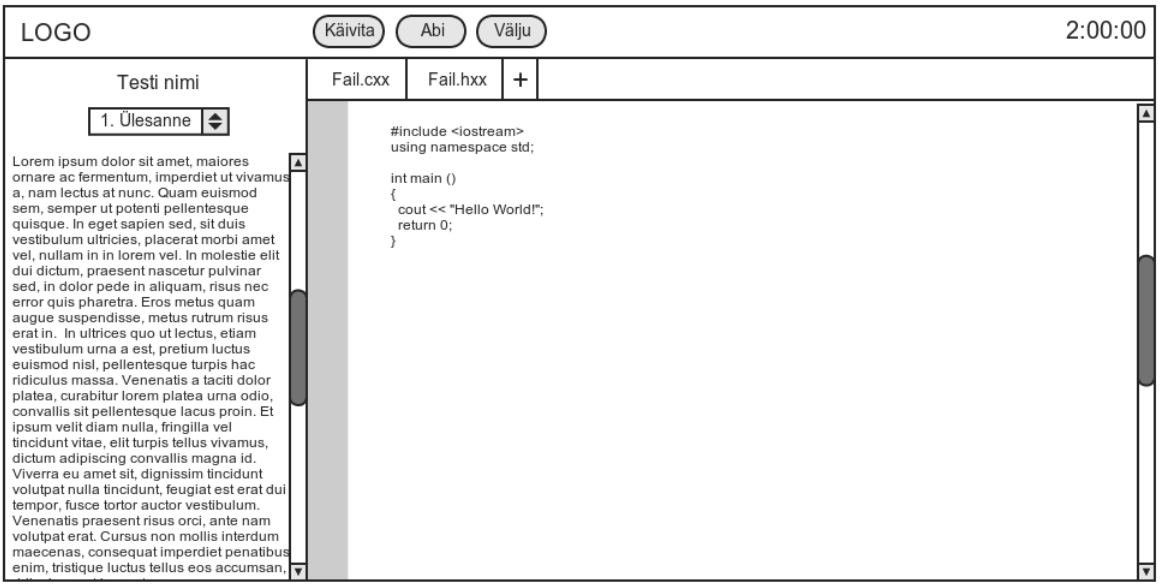
Koodiredaktoriks saab JavaScriptis kirjutatud ACE Editor. Sellel on väga lai süntaksi äramärkimise tugi erinevate keelte jaoks, see on paljude seadistamisvõimalustega, sellega saab kasutada üldtuntud kiirklahvi kombinatsioone, seda on lihtne veebilehele manustada ja kokkuvõtteks võib öelda, et see toimib üldse väga sarnaselt töölaua koodiredaktoritega. (Cloud9ide & Mozilla)

3.6. Failiserver

Failiserver on eespool kirjeldatud serveritest kõige primitiivsema ehitusega. See serverib välja ainult staatiliste failide kataloogi ja ei midagi muud. Failiserveri kood sõltub lisaks Pythoni standardmoobulitele veel ainult Twisted veebiserveri moodulitest. Kataloogi välja serveerimiseks kasutatakse moodulit *twisted.web.static.File*.

3.7. Prototüüp

Visuaalsete elementide paigutuse kujundamisel lähtuti põhimõttest, et vähem on rohkem. Veel oli oluline, et ülesande tekst ja koodiredaktor oleks kasutajale korraga näha ning neile eraldatav pindala oleks võimalikult suur. Lehe täielik laadimine võib toimuda vaid esimesel laadimisel. Aktiivse ülesande või faili vahetamisel peaks uuenema vaid vastav lehekülje osa. Pärast mitmeid kavandeid jäi testide lahendamise lehe paigutuseks selline, nagu kujutab prototüüp joonisel 9.



Joonis 9. Testide lahendamise lehe prototüüp.

4. Arendamine

4.1. Arenduskeskkond

Arendusplatvormina on kasutusel Ubuntu derivaat Kubuntu versiooniga 12.10. Kubuntusse on installeeritud Django 1.5, PostgreSQL 9.1, Twisted 12.3 ja kõik muud vajalikud vaba tarkvara paketid. Arenduskeskkonna seadistamise juhendiga tutvume peatükis 4.3. Peamine arendus toimub programmeerimiskeeltes Python ja JavaScript. Koodi kirjutamiseks kasutatakse koodiredaktorit Kate. Arendatava tarkvara lähtekoodi hoitakse Mercuriali versioonihalduses, mille repositoorium asub pilveteenuses Bitbucket. (Kubuntu community, 2013; Kate, 2013; Mercurial community, 2013; Atlassian, 2013)

Koodi kvaliteedi kontrollimiseks on arenduskeskkonda installeeritud veel Pythoni moodulid Pyflakes ja PEP8. Kui Pyflakes otsib koodist programmilisi vigu, siis PEP8 on koodist stiilivigade leidmiseks. (Frost, 2013; Rocholl, 2013)

4.2. Arendusprotsess

Arendusprotsessi saab jagada kolmeks faasiks. Esimeses faasis loodi tarkvara osa, mis koostab kasutaja kirjutatud koodi jaoks automaattestid. Teises faasis loodi sellele graafiline kasutajaliides, mis võimaldab veebilehitsejas lahendada programmeerimisülesandeid. Viimases, kolmandas faasis, loodi tarkvarale endale automaattestid, mis võimaldavad lihtsamini üles leida tekkinud vigu ning tehti eelmises kahes arendusfaasis valminud süsteemile mitmeid parandusi ja täiustusi. Viimast, kolmandat faasi, polnud algselt üldse plaanis, kuid see tuli juurde tekitada, sest tarkvara käsitsi testimine muutus liigselt aeganõudvaks ning vigade otsimine äärmiselt ebamugavaks.

4.3. Põhisüsteemi arendamine

Esimeses faasis alustati programmeerimisülesande automaattestide JSON formaadis kirjelduse välja töötamisega. Sellest kirjeldusest pidi saama tuletada automaattestid C++, Java, PHP ja Pythoni programmeerimiskeeltes. Nende automaattestidega pidi saama testida nii funktsioone, meetodeid kui klasse. Esimene arendusfaas osutus kohe väga aeganõudvaks, sest paralleelselt automaattestide kirjelduse välja töötamisega tuli arendada rakendusserveri ressursse ja käivitusserverit, mis JSON formaadis automaattestide kirjelduse programmerimiskeeltesse ümber konverteerib. Pärast mitmeid muutmisi ja täiendusi jäeti kasutusele kirjeldusformaat nagu see on tabelis 1 ning koodinäidetes 2 ja 3.

#	Attribuut	Andmetüüp	Kohustuslik
1	files	array	✔
2	files[F].name	string	✔
3	files[F].tests	array	✔
4	files[F].tests[T].name	string	✔
5	files[F].tests[T].object	object	
6	files[F].tests[T].object.name	string	
7	files[F].tests[T].object.args	array	
8	files[F].tests[T].object.kwargs	array	
9	files[F].tests[T].asserts	array	✔
10	files[F].tests[T].asserts[A].type	string	✔
11	files[F].tests[T].asserts[A].callable	object	✔
12	files[F].tests[T].asserts[A].callable.name	string	✔
13	files[F].tests[T].asserts[A].callable.args	array	
14	files[F].tests[T].asserts[A].callable.kwargs	array	
15	files[F].tests[T].asserts[A].expected	string, number array, true, false, null	
16	files[F].tests[T].asserts[A].kwargs	array	

Tabel 1. Programmeerimisülesannete automaattestide JSON formaadis kirjelduse attribuudid.

1. testitavad failid; 2. testitava faili nimi; 3. testitava faili testjuhtumid; 4. testjuhtumi nimi; 5. testitav objekt; 6. testitava objekti nimi; 7. testitava objekti konstruktori argumendid; 8. testitava objekti konstruktori võtmeargumendid; 9. testjuhtumi kinnitused; 10. kinnituse tüüp; 11. testitav meetod või funktsioon; 12. testitava meetodi või funktsiooni nimi; 13.

testitava meetodi või funktsiooni argumendid; 14. testitava meetodi või funktsiooni võtmeargumendid; 15. kinnituse oodatav väärtus; 16. kinnituse lisaargumendid (nt. "delta")

(Crockford, 2006)

```
{
    "files": [
        {
            "name": "FILE_NAME",
            "tests": [
                {
                    "name": "TEST_NAME",
                    "object": {
                        "name": "OBJECT_NAME",
                        "args": [OBJECT_ARG_1, ... ],
                        "kwargs": [["OBJECT_KWARG_1", OBJECT_KWARG_1_VALUE], ... ]
                    },
                    "asserts": [
                        {
                            "type": "ASSERT_TYPE",
                            "callable": {
                                "name": "CALLABLE_NAME",
                                "args": [CALLABLE_ARG_1, ... ],
                                "kwargs": [["CALLABLE_KWARG_1", CALLABLE_KWARG_1_VALUE], ... ]
                            },
                            "expected": EXPECTED_VALUE
                        },
                        ...
                    ]
                },
                ...
            ]
        },
        ...
    ]
}
```

Koodinäide 2. Programmeerimisülesannete automaattestide kirjeldamine.

Kuna kõik attribuudid pole kohustuslikud, siis minimaalse programmeerimisülesande automaattesti kirjeldus on antud koodinäites 3. See testib ainult ühte funktsiooni, mis ei ole ühegi klassi meetod ja millele ei ole antud ühtegi argumenti.

```
{
    "files": [
        {
            "name": "FILE_NAME",
            "tests": [
                {
                    "name": "TEST_NAME",
                    "asserts": [
                        {
                            "type": "ASSERT_TYPE",
                            "callable": {"name": "CALLABLE_NAME"},
                        }
                    ]
                }
            ]
        }
    ]
}
```

Koodinäide 3. Programmeerimisülesannete automaattestide minimaalne kirjeldamine.

Kõik testjuhtumites kasutatavad kinnitustüübid koos selgitustega on välja toodud tabelis 2. Lisaks tabelis toodud kinnitustüüpidele on olemas veel pseudo-kinnitustüüp PASS, mille puhul ei rakendata ühtegi kontrolli ja seda saab kasutada kinnituste vahel funktsioonide või meetodite väljakutsumiseks.

Kinnitustüüp	Kinnitab, et tulemus	Vajab oodatavat väärtust?
TRUE	on tõene	
FALSE	on väär	
EQUAL	on võrdne	✔
NOT_EQUAL	ei ole võrdne	✔
LESS	on väiksem	✔
LESS_EQUAL	on väiksem või võrdne	✔
GREATER	on suurem	✔
GREATER_EQUAL	on suurem või võrdne	✔
NULL	on tühi väärtus	
NOT_NULL	ei ole tühi väärtus	
IN	kuulub hulka	✔
NOT_IN	ei kuulu hulka	✔
IS_INSTANCE	on eksemplar	✔
NOT_INSTANCE	ei ole eksemplar	✔
ALMOST_EQUAL	on peaaegu võrdne	✔
NOT_ALMOST_EQUAL	ei ole peaaegu võrdne	✔
REGEX_MATCHES	kattub regulaaravaldisega	✔
NOT_REGEX_MATCHES	ei kattu regulaaravaldisega	✔
ITEMS_EQUAL	on massiiv ja võrdub oodatava tulemusega (elementide järjekord pole oluline)	✔
LIST_EQUAL	on massiiv ja võrdub oodatava tulemusega (elementide järjekord on oluline)	✔
DICT_EQUAL	on assotsiatiivne massiiv ja võrdub oodatava tulemusega	✔

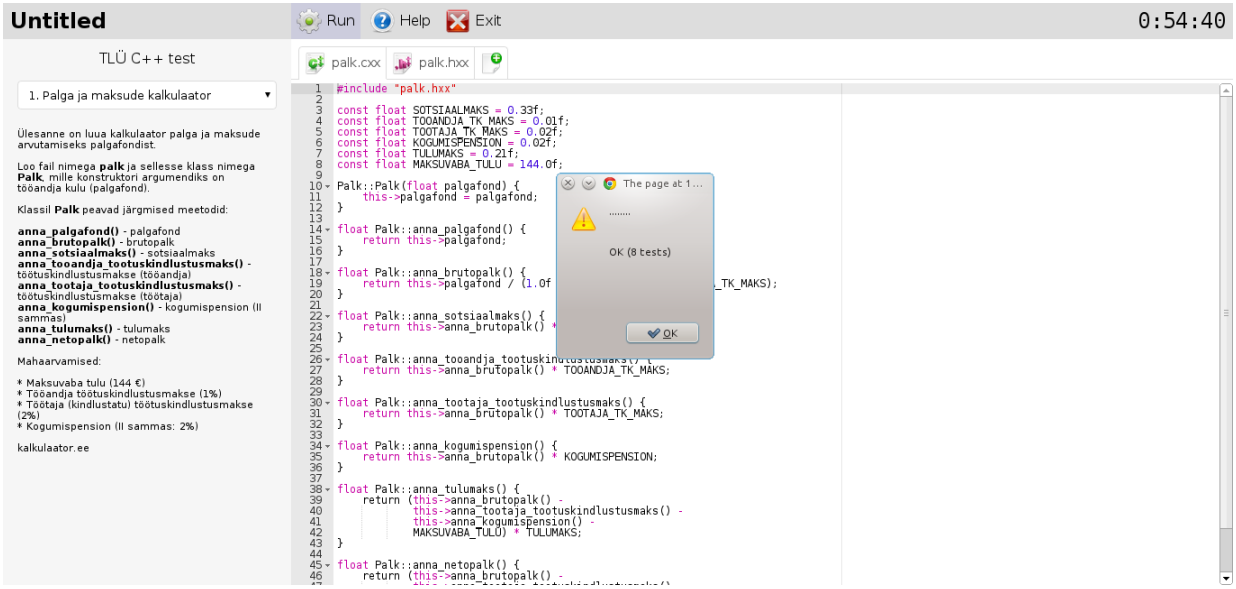
Tabel 2. Implementeeritud kinnitustüübid.

Koodi testimiseks võeti Pythoni keele jaoks kasutusele standardmoodul unittest, PHP keele jaoks PHPUnit teek ja C++ keele jaoks CppUnit teek. Tähtaja lähenedes ja aja puudusel jäeti

hetkel välja Java keele tugi ning mõned vähem olulised kinnitustüübid (näiteks massiivide ja regulaaravaldiste jaoks) pole implementeeritud C++ ja PHP keeltes.

4.4. Kasutajaliidese arendamine

Kui tarkvara põhisüsteem oli enamvähem valmis ehk käivitusserver oskas kasutajate koodi jaoks automaatteste koostada ja nende abil ka koodi testida, hakati teises arendusfaasis arendama graafilist kasutajaliidest. Graafilise kasutajaliidese arendamine osutus samuti väga aeganõudvaks, sest selles arendusfaasis oli vaja korraga arendada rakendusserveri ressursse, esitlusserveri vaateid, failiserverit ning luua joonisel 9 kujutatud prototüübile vastav disain. Oodatust keerulisemaks osutus ka JavaScripti moodulite omavahel integreerimine, sest arendusfaasis otsustati kasutusele võtta JavaScripti teek nimega RequireJS, mis kiirendab JavaScripti moodulite laadimist ja parandab koodi kvaliteeti. See otsus muutis oluliselt plaanitud JavaScripti struktuuri. Testi lahendamise lehekülje disain on kujutatud joonisel 10. (Dojo Foundation, 2013)



Joonis 10. Testide lahendamine.

Graafilise kasutajaliidese arendamisel sai selgeks, et käsitsi kõikide serverite kävitamine ja peatamine ning andmebaasi initsialiseerimine on ebamugav ja kulutab asjatult aega. Nende tegevuste kiirendamiseks loodi tarkvara haldusskript, mis võimaldab hõlpsasti teha järgmisi tegevusi:

- initsialiseerida andmebaasi
- käivitada, taaskäivitada ja peatada servereid (nii eraldi kui kõik korraga)
- kuvada, millised serverid on käivitatud ja millised mitte
- uuendada JavaScripti konfiguratsioonifaili
- analüüsida koodi kvaliteeti ja anda teada vigadest

Haldusskript asub tarkvara juurkataloogis ja on käivitatav nõnda:

```
$ ./manage
```

4.5. Unit testide kirjutamine ja vigade parandamine

Käivitusserveri käsitsi testimine oli aeganõudev ja tülikas. Unit testid on aga kiire ja lihtne viis testida koodi vastupidavust ja leida vigu. Samuti aitavad need hõlpsasti üle kontrollida, kas uute funktsionaalsuse implementeerimisega ei lõhutud ära juba implementeeritud asju. Selleks kirjutatigi arendusfaasi lõpuosas tarkvara kõige tähtsamale osale 178 unit testi. Need panevad proovile kasutaja kirjutatud koodi testivate automaattestide loomise, kasutades erinevaid andmetüüpe kõigis implementeeritud programmerimiskeeltes. AMD E-350 protsessoriga sülearvutil võtab kõikide testide läbimine keskmiselt aega vaid 0.26 sekundit. Suuremaid ootamatuid raskusi selles arendusjärgus ette ei tulnud, kuid unit testide kirjutamisega leiti suurel hulgal vigu ning täiendati oluliset tarkvara vastupidavust.

Tarkvara failid on jagatud järgnevatesse kataloogidesse:

- api - rakendusserveri failid
- common - ühisfailid
- config - konfiguratsioonifailid
- doc - dokumentatsioon
- fo - esitlusserveri failid
- runner - käivitusserveri failid
- static - staatilised failid, failiserveri failid

4.6. Näidisülesannete loomine

Näidisülesandeid sai loodud kaks tükki. Esimeses ülesandes tuleb luua palgakalkulaator palgafondist erinevate maksude ja tasude arvutamiseks. Teises ülesandes tuleb luua klassid taasesitusloendi ja selles olevate lugude jaoks. Nende ülesannete loomisel oli vaja samuti viia tarkvarasse sisse väiksemaid parandusi ja täiendusi. Üheks olulisemaks täienduseks oli varem välja jäetud ALMOST_EQUAL ja NOT_ALMOST_EQUAL kinnitustüüpide lisamine C++ ja PHP keeltesse, mis võimaldab anda ülesannete lahendajale eksimisruumi (tabel 2). Ujukomaga arvutustes võivad erinevad programmeerimiskeeled anda pisut erinevaid tulemusi, seega oli selline täiendus hädavajalik. Teiseks täienduseks oli kaitse käivitusserveri ülekoormamise eest ühe kasutaja poolt. Tarkvara lubab käivitusserverisse saata korraga vaid ühe päringu koodi kävitamiseks ja testimiseks. Uut päringut on võimalik teha alles siis, kui eelmise päringu kohta on tulnud vastus.

4.7. Paigaldamine

Tarkvara paigaldamiseks on vajalik operatsioonisüsteem Ubuntu 12.10 või mõni selle derivaatidest (Kubuntu, Xubuntu jne). Ubuntu 12.10 saab alla laadida aadressilt http://releases.ubuntu.com/ ja juhiseid installeerimiseks leiab aadressit https://help.ubuntu.com/community/Installation. Tarkvara võib tööle saada ka paljudes teistes operatsioonisüsteemides ja nende versioonides, kuid autor annab ametliku toe vaid Ubuntu 12.10 operatsioonisüsteemile.

Pärast Ubuntu paigaldamist ja sellesse sisse logimist, tuleb kõigepealt installeerida Ubuntu paketivaramutest vajalikud tarkvara paketid:

```
$ sudo apt-get install g++ libcppunit-dev pep8 php5-cli php-pear postgresql \
postgresql-server-dev-9.1 pyflakes python python-cjson python-dev python-nose python-pip \
python-psycopg2 python-twisted python3
```

Djangot ja Django-Tastypie pakette ei installeerita apt-get paigaldusvahendi kaudu, kuna Ubuntu 12.10 paketivaramutes ei ole nendest uusi versioone.
Installeeri Django:

```
$ sudo pip install django==1.5
```

Seejärel installeeri Django-Tastypie:

```
$ sudo pip install django-tastypie==0.9.14
```

Viimasena installeeri PHP repositooriumist PHPUnit:

```
$ sudo pear config-set auto_discover 1
$ sudo pear install pear.phpunit.de/PHPUnit-3.7.14
```

Loo uus PostgreSQL kasutaja:

```
$ sudo -u postgres createuser --superuser $USER
```

Taaskäivita PostgreSQL:

```
$ sudo service postgresql restart
```

Määra faili lõpuosas IPv4 ja IPv6 kohalike ühenduste meetodiks *md5* asemel *trust*:

```
$ sudo nano /etc/postgresql/9.1/main/pg_hba.conf
```

Laadi alla tarkvara ja mine selle juurkataloogi:

```
$ wget http://www.tlu.ee/~karmux/src.zip
$ unzip src.zip
$ cd src
```

Loo kohaliku masina jaoks konfiguratsioonifail:

```
$ nano config/dev.py
```

Lisa sinna järgmised read, salvesta (Ctrl + O) ja välju (Ctrl + X):

```
from config.base import SERVERS

SERVERS['db']['user'] = '' # Siin defineeri oma PostgreSQL kasutaja
DEBUG = True
```

Käivita tarkvara automaattestid ja veendu, et kõik testid lähevad läbi:

```
$ nosetests
```

Initsialiseeri andmebaas:

```
$ ./manage init-db
```

Valmista sait ette:

```
$ ./manage prepare
```

Käivita serverid:

```
$ ./manage start
```

Veendu, et kõik serverid said käivitatud:

```
$ ./manage show-servers
+ api              [RUNNING]
+ fo               [RUNNING]
+ runner           [RUNNING]
+ static           [RUNNING]
------------------------------
+ db               [RUNNING]
```

Ja ongi valmis. Nüüd võid avada tarkvara veebikeskkonna esilehe:

```
$ xdg-open http://localhost:8000
```

Sisse logimiseks saab kasutada test kasutajat juku parooliga 123456.

5. Hindamine

5.1. Nõuetele vastavus

Peatükis 2 said kirja pandud nõuded loodavale tarkvarale. Käesolevas alampeatükis vaatleme, millised nendest nõutest said täidetud ja millised mitte.

Kõik üksteist funktsionaalset nõuet ning enamus mitte-funktsionaalseid nõuded said täidetud. Kõige olulisem puudujääk on see, et tähtaja lähenedes ja aja puudusel tuli täielikult välja jätta Java keele tugi. Samadel põhjustel ei ole ka kirjutamise lõpetamise hetkel (märtsis 2013) tarkvara kohta muud dokumentatsiooni, kui antud bakalaureusetöö. Pärast tarkvara käsitsi testimist erinevates personaalarvutite ja mobiilsete seadmete veebilehitsejates, võib öelda, et nõue, kus tarkvara peab olema kasutatav kõikide levinumate personaalarvutite ja mobiilsete seadmete veebilehitsejates, sai peaaegu täidetud. Problemaatilised olid vaid Internet Explorer veebilehitsejad. Kuni versioonini 9 ei suudetud testi lahendamise lehel üldse mingisugust kasutatavat graafilist liidest ette näidata. Versioonis 10 oli tarkvara küll kasutatav, kuid süntaksi äramärkimine ei töötanud ja see muutis koodi kirjutamise ebamugavaks. Käsitsi testimise tulemused erinevate veebilehitsejatega on kokku võetud tabelis 3.

Veebilehitseja	Tulemus
Google Chrome 25	✔ Kõik töötab
Internet Explorer 9	✘ Teste ei saa lahendada
Internet Explorer 10	✔ Kõik töötab v.a. süntaksi äramärkimine
Firefox 19	✔ Kõik töötab
Safari 5.1	✔ Kõik töötab
Opera 12	✔ Kõik töötab
Android 2.3 brauser	✔ Kõik töötab
Android 4.0 brauser	✔ Kõik töötab
iOS 6 brauser	✔ Kõik töötab
Opera Mobile 12	✔ Kõik töötab

Tabel 3. Tarkvara töötamine erinevates veebilehitsejates.

5.2. Automaattestid

Kõik 178 automaattesti lähevad läbi. Seega testidega kaetud funktsionaalsuses vigu ei esine. Testide käivitamise väljund on järgmine:

```
$ nosetests
..........................................................................
..........................................................................
..................
----------------------------------------------------------------------
Ran 178 tests in 0.262s

OK
```

5.3. Ülesannete lahendamine tudengitega

Tarkvara testimine tudengite peal toimus Tallinna Ülikoolis aines objektorienteeritud programmeerimine keeles C++. Praktikumiks valmistas autor ette kaks näidisülesannet, mille lahendamine nõudis objektorienteeritud programmeerimist. Nagu kursuse nimigi ütleb, siis toimus ülesannete lahendamine programmeerimiskeeles C++. Kokku kestis tudengitega ülesannete lahendamine umbes kaks ja pool tundi ning enne seda olid tudengid jõudnud umbes 20 tundi objektorienteeritud C++ keelega tutvuda.

Ülikoolis testimise jaoks paigaldas autor tarkvara koolis asuvasse virtuaalserverisse. Tallinna Ülikooli sisevõrgus olles leiab tarkvara aadressilt http://praktika1.cs.tlu.ee:8000. Ühtegi tehnilist tõrget kahe ja poole tunnise loengu jooksul ei esinenud. Kuigi serverile on C++ võrreldes PHP või Pythoniga koormavam, kuna enne igat koodi käivitust on vaja kood ka kompileerida, siis 14 õpilasega klassis oli süsteem silma järgi sama kiire kui ühe kasutajaga. Usutavasti saaks see sama virtuaalserver kenasti hakkama ka palju suurema õpilaste hulgaga. Kui keskenduda sellele, mis saanuks olla veel paremini, siis CppUnit raamistik oli mõnes kohas üsna napisõnaline õpilastele valede vastuste kohta selgituste andmisel, mis muutis koodi parandamise keerulisemaks. Seda saab parandada CppUniti väljundit veidi ümber tehes.

Ülesannete lahendamine toimus koos autori selgitavate kommentaaridega. Vahepeal said tudengid ka oma peaga lahendusi välja nuputada. Praktikumi lõpuks olid mõlemad

ülesanded koos tudengitega läbi töötatud ning enamustel tudengitel läksid läbi kõik ülesannete automaattestid.

6. Edasiarendus

Siia peatükki on kirja pandud mõtted antud bakalaureusetöös arendatud tarkvara edasiarendamiseks. Ideed on järjestatud prioriteedi järgi. Esimesed nimetatud asjad on need, mis töö autori arvates tuleks kõige enne valmis teha ja viimastena nimetatud ei ole nii olulise tähtsusega.

1. Automaattestid

Nagu eespool sai mainitud, et automaattestid on kiire ja mugav viis võimalike vigade leidmisel, siis selleks tulekski esmalt katta võimalikult suur osa koodist unit ning funktsionaaltestidega.

2. Pakktöötlusserver

Hetkel ei toimu testide lõplikku hindamist süsteemi poolt. Plaan on luua pakktöötlusserver, mis otsib lõpetatud teste, kävitab iga selle ülesande peal automaattestid ja koostab detailse raporti sellest, kui palju automaatteste läheb läbi ja kui palju punkte lahendatud ülesannete eest saadi.

3. Testide tulemused

Kui pakktöötlusserver on valmis, siis on võimalik näidata kasutajale tema lahendatud testide tulemusi testide loendi- ja detailvaadetes. Testi loendivaates kuvataks ainult punktide arv võimalikust punktide arvust. Testi detailvaates oleks juba täpsemini näha, kui palju punkte mingi ülesande eest saadi ning missugused testid läksid läbi ja millised mitte.

4. Keskkontor

Praegu tuleb uued testid ja ülesanded käsitsi andmebaasi sisestada ning nende muutmiseks tuleb neid käsitsi andmebaasis muuta. Nende tegevuste mugavamaks muutmiseks tuleks luua keskkontor, kus on võimalik vormide abil hallata teste ja ülesandeid ning kus oleks näha ka kasutajate tulemused.

5. Laiem programmeerimiskeelte tugi

Hetkel on võimalik ülesandeid lahendada C++, PHP ja Pythoni programmeerimiskeeltes. Nimetatud keeled pole aga ainsad populaarsed keeled. Laiema kasutajaskonna jaoks võiks lisada veel välja jäänud Java keele ning näiteks C, C#, Java, JavaScript ja Ruby keelte toe.

6. Vahemälu

Andmebaasilt koormuse vähendamiseks ja tarkvara üldise skaleeruvuse ja jõudluse parandamiseks läheks tarvis vahemälu rakenduskihi ja esitluskihi vahele. Kahe Pythoni serveri vahele vahemälu lisada pole keeruline, kuid vahemälu implementeerimine rakendusserveri ja JavaScripti vahele nõuab mõningat refaktooringut esitluskihi poole peal.

7. Väiksematele seadmetele optimeeritud kasutajaliides

Tänapäeval muutuvad järjest populaarsemaks nutitelefonid ja teised väikeste ekraanidega seadmed. Nendel seadmetel pole arendatud tarkvara kasutamine kuigi mugav. Et midagi ekraani pealt välja lugeda, tuleb pilti üsna palju suurendada. Väikeste ekraanidega seadmete tarbeks on võimalik astmelised laadilehed optimeerida nii, et veebilehitsejas pole vaja vaadet suurendada ja see muudaks tarkvara kasutamise oluliselt mugavamaks näiteks nutitelefonides.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärk oli arendada programmeerimisoskuste hindamise veebikeskkond, mis hindaks automaatselt kasutaja kirjutatud koodi vastavust ülesannetes püstitatud probleemi lahendusele. Eesmärgi täitmiseks uuriti sarnaseid lahendusi ning koostati nende headest ja halbadest külgedest lähtuvalt arendatava tarkvara nõuded. Seejärel loodi kavand, mille järgi arendati valmis tarkvara ning lõpetuseks anti hinnang valminud tööle.

Hindamisel selgus, et päris kõike, mis algselt oli plaanis implementeerida, ei jõutudki lõpuni valmis teha, kuid sellest hoolimata oli tarkvara esmane proovimine tudengite peal edukas. Tudengite peal katsetatud näidisülesanded on tarkvaras olemas kohe peale selle paigaldamist ning nende võimalikud lahendused erinevates programmeerimiskeeltes leiab antud bakalaureusetöö lisade alt.

Töö lõpuosas toodi välja mõned ideed valminud tarkvara edasiarendamiseks, mis muudaksid tarkvara funktsionaalsemaks ja selle kasutamise mugavamaks. Loetletud ideed on tarkvara autoril plaanis ka ellu viia.

Käesoleva bakalaureusetöö tulemuseks olevat tarkvara oli autoril väga huvitav arendada, sest juba tuttavate tehnoloogiate kõrval sai palju kogemusi juurde ka mitmete teiste raamistike ja teekide kasutamisel arenduses.

Summary

Web Environment for Assessment of Programming Skills

BSc thesis by Karmo Rosental

An author of this thesis had an interest about creating unit tests for some piece of code, regardless of the programming language. This interest grew to an idea to build an environment which automatically tests user's code using that kind of unit tests. The aim of this bachelor thesis is to develop a web environment for assessment of programming skills.

In the beginning, this thesis gives a short overview about some existing solutions that have similarities with planned software. Then it describes requirements and plan for the software that is being developed. After planning there is a chapter about the development process of the web environment, writing sample assessment tests and installing the software. Finally, the author evaluates the software that is the outcome of this bachelor thesis.

Software runs on Ubuntu GNU/Linux operating system and is built using the cutting edge technologies such as Python, Django, Twisted, Backbone.js and jQuery. It depends only on free software frameworks and libraries. Web environment can be used in almost all popular desktop and mobile web browsers.

Kasutatud materjal

- Atlassian. (2013). *Free source code hosting for Git and Mercurial by Bitbucket.* Viimati loetud 09.04.2013 aadressilt: https://bitbucket.org
- Canonical Ltd. (2013). *Home | Ubuntu.* Viimati loetud 23.03.2013 aadressilt: http://www.ubuntu.com
- Cloud9ide & Mozilla. (2013). *Ace - The High Performance Code Editor for the Web.* Viimati loetud 23.03.2013 aadressilt: http://ace.ajax.org
- CodeEval Inc. (2013). *CodeEval - Evaluations Made Easy.* Viimati loetud 24.02.2013 aadressilt: https://www.codeeval.com
- Codility Ltd. (2013). *Automated tests of programming skills. Assessment of software developers. Recruitment software. Codility.* Viimati loetud 24.02.2013 aadressilt: http://codility.com
- Coster, O. (2010). *What are the benefits of an N-layered architecture?* Viimati loetud 11.03.2013 aadressilt: http://stackoverflow.com/questions/2637114/#2637143
- Crockford, D. (2006). *The application/json Media Type for JavaScript Object Notation (JSON).* Viimati loetud 15.03.2013 aadressilt: http://www.ietf.org/rfc/rfc4627.txt?number=4627
- Directi Group. (2013). *Programming Competition,Programming Contest,Online Computer Programming.* Viimati loetud 25.02.2013 aadressilt: http://www.codechef.com
- Django Software Foundation. (2013). *The Web framework for perfectionists with deadlines | Django.* Viimati loetud 12.03.2013 aadressilt: https://www.djangoproject.com
- DocumentCloud. (2013a). *Backbone.js.* Viimati loetud 23.03.2013 aadressilt: http://backbonejs.org
- DocumentCloud. (2013b). *Underscore.js.* Viimati loetud 23.03.2013 aadressilt: http://underscorejs.org
- Dojo Foundation. (2013). *RequireJS.* Viimati loetud 23.03.2013 aadressilt: http://requirejs.org
- Frost, D. (2013). *pyflakes 0.6.1 : Python Package Index.* Viimati loetud 23.03.2013 aadressilt: https://pypi.python.org/pypi/pyflakes
- Hossain, M., & Hausenblas, M. (2013). *enable cross-origin resource sharing.* Viimati loetud 12.03.2013 aadressilt: http://enable-cors.org

- jQuery Foundation. (2013). *jQuery.* Viimati loetud 23.03.2013 aadressilt: http://jquery.com
- Kate. (2013). *Kate | Get an Edge in Editing | The Kate Editor Homepage.* Viimati loetud 24.03.2013 aadresslt: http://kate-editor.org
- Kubuntu community. (2013). *Kubuntu | Friendly Computing.* Viimati loetud 24.03.2013 aadressilt: http://www.kubuntu.org
- Lindsley, D. (2013). *Welcome to Tastypie! — Tastypie 0.9.14 documentation.* Viimati loetud 23.03.2013 aadressilt: http://django-tastypie.readthedocs.org/en/latest/
- Mercurial community. (2013). *Mercurial SCM.* Viimati loetud 09.04.2013 aadressilt: http://mercurial.selenic.com
- Pascu, D. (2007). *python-cjson 1.0.5 : Python Package Index.* Viimati loetud 12.03.2013 aadressilt https://pypi.python.org/pypi/python-cjson
- PostgreSQL Global Development Group. (2013). *PostgreSQL: The world's most advanced open source database.* Viimati loetud 23.03.2013 aadressilt http://www.postgresql.org
- Project Euler. (2013). *Project Euler.* Viimati loetud 26.02.2013 aadressilt: http://projecteuler.net
- Rocholl, J. C. (2013). *pep8 1.4.5 : Python Package Index.* https://pypi.python.org/pypi/pep8
- Twisted Matrix Labs. (2013). *Twisted.* Viimati loetud 23.03.2013 aadressilt: http://twistedmatrix.com
- Varrazzo, D. (2012). *PostgreSQL + Python | Psycopg.* Viimati loetud 23.03.2013 aadressilt: http://initd.org/psycopg/

Lisad

1. Lähtekoodi failide loend

```
api
api/common
api/common/auth.py
api/common/__init__.py
api/common/models.py
api/common/resources.py
api/common/serializers.py
api/common/views.py
api/__init__.py
api/server.py
api/test
api/test/__init__.py
api/test/models.py
api/test/resources.py
api/urls.py
common
common/api.py
common/auth.py
common/consts.py
common/cors.py
common/dataobject.py
common/error.py
common/__init__.py
common/misc.py
common/utils.py
config
config/api.py
config/base.py
config/dev.py
config/fo.py
config/__init__.py
config/runner.py
data.py
doc
doc/INSTALL
fo
fo/common
fo/common/forms.py
fo/common/__init__.py
fo/common/urls.py
fo/common/views.py
fo/__init__.py
fo/server.py
fo/templates
fo/templates/common
fo/templates/common/403.html
fo/templates/common/404.html
fo/templates/common/500.html
fo/templates/common/base.html
fo/templates/common/home.html
fo/templates/common/login.html
fo/templates/common/register.html
fo/templates/test
fo/templates/test/test_detail.html
fo/templates/test/test_list.html
fo/templates/test/test_start.html
fo/test
fo/test/__init__.py
fo/test/templatetags
fo/test/templatetags/__init__.py
fo/test/templatetags/javascript.py
```

```
fo/test/urls.py
fo/test/views.py
fo/urls.py
lib
lib/ace
lib/ace/ace.js
lib/ace/ext-static_highlight.js
lib/ace/ext-textarea.js
lib/ace/keybinding-emacs.js
lib/ace/keybinding-vim.js
lib/ace/mode-asciidoc.js
lib/ace/mode-c9search.js
lib/ace/mode-c_cpp.js
lib/ace/mode-clojure.js
lib/ace/mode-coffee.js
lib/ace/mode-coldfusion.js
lib/ace/mode-csharp.js
lib/ace/mode-css.js
lib/ace/mode-diff.js
lib/ace/mode-glsl.js
lib/ace/mode-golang.js
lib/ace/mode-groovy.js
lib/ace/mode-haxe.js
lib/ace/mode-html.js
lib/ace/mode-jade.js
lib/ace/mode-java.js
lib/ace/mode-javascript.js
lib/ace/mode-json.js
lib/ace/mode-jsp.js
lib/ace/mode-jsx.js
lib/ace/mode-latex.js
lib/ace/mode-less.js
lib/ace/mode-liquid.js
lib/ace/mode-lua.js
lib/ace/mode-luapage.js
lib/ace/mode-markdown.js
lib/ace/mode-ocaml.js
lib/ace/mode-perl.js
lib/ace/mode-pgsql.js
lib/ace/mode-php.js
lib/ace/mode-powershell.js
lib/ace/mode-python.js
lib/ace/mode-ruby.js
lib/ace/mode-scad.js
lib/ace/mode-scala.js
lib/ace/mode-scss.js
lib/ace/mode-sh.js
lib/ace/mode-sql.js
lib/ace/mode-svg.js
lib/ace/mode-tcl.js
lib/ace/mode-textile.js
lib/ace/mode-text.js
lib/ace/mode-typescript.js
lib/ace/mode-xml.js
lib/ace/mode-xquery.js
lib/ace/mode-yaml.js
lib/ace/theme-ambiance.js
lib/ace/theme-chrome.js
lib/ace/theme-clouds.js
lib/ace/theme-clouds_midnight.js
lib/ace/theme-cobalt.js
lib/ace/theme-crimson_editor.js
lib/ace/theme-dawn.js
lib/ace/theme-dreamweaver.js
lib/ace/theme-eclipse.js
lib/ace/theme-github.js
lib/ace/theme-idle_fingers.js
lib/ace/theme-kr.js
lib/ace/theme-merbivore.js
```

```
lib/ace/theme-merbivore_soft.js
lib/ace/theme-mono_industrial.js
lib/ace/theme-monokai.js
lib/ace/theme-pastel_on_dark.js
lib/ace/theme-solarized_dark.js
lib/ace/theme-solarized_light.js
lib/ace/theme-textmate.js
lib/ace/theme-tomorrow.js
lib/ace/theme-tomorrow_night_blue.js
lib/ace/theme-tomorrow_night_bright.js
lib/ace/theme-tomorrow_night_eighties.js
lib/ace/theme-tomorrow_night.js
lib/ace/theme-twilight.js
lib/ace/theme-vibrant_ink.js
lib/ace/theme-xcode.js
lib/ace/worker-coffee.js
lib/ace/worker-css.js
lib/ace/worker-javascript.js
lib/ace/worker-json.js
lib/ace/worker-xquery.js
lib/backbone
lib/backbone/backbone-min.js
lib/jquery
lib/jquery-contextmenu
lib/jquery-contextmenu/images
lib/jquery-contextmenu/images/cut.png
lib/jquery-contextmenu/images/door.png
lib/jquery-contextmenu/images/page_white_add.png
lib/jquery-contextmenu/images/page_white_copy.png
lib/jquery-contextmenu/images/page_white_delete.png
lib/jquery-contextmenu/images/page_white_edit.png
lib/jquery-contextmenu/images/page_white_paste.png
lib/jquery-contextmenu/jquery.contextMenu.css
lib/jquery-contextmenu/jquery.contextMenu.js
lib/jquery-contextmenu/jquery.ui.position.js
lib/jquery-cookie
lib/jquery-cookie/jquery.cookie.js
lib/jquery/jquery-1.8.3.min.js
lib/jquery-ui
lib/jquery-ui/jquery-ui-1.8.23.min.js
lib/jquery-ui/themes
lib/jquery-ui/themes/smoothness
lib/jquery-ui/themes/smoothness/images
lib/jquery-ui/themes/smoothness/images/ui-bg_flat_0_aaaaaa_40x100.png
lib/jquery-ui/themes/smoothness/images/ui-bg_flat_75_ffffff_40x100.png
lib/jquery-ui/themes/smoothness/images/ui-bg_glass_55_fbf9ee_1x400.png
lib/jquery-ui/themes/smoothness/images/ui-bg_glass_65_ffffff_1x400.png
lib/jquery-ui/themes/smoothness/images/ui-bg_glass_75_dadada_1x400.png
lib/jquery-ui/themes/smoothness/images/ui-bg_glass_75_e6e6e6_1x400.png
lib/jquery-ui/themes/smoothness/images/ui-bg_glass_95_fef1ec_1x400.png
lib/jquery-ui/themes/smoothness/images/ui-bg_highlight-soft_75_cccccc_1x100.png
lib/jquery-ui/themes/smoothness/images/ui-icons_222222_256x240.png
lib/jquery-ui/themes/smoothness/images/ui-icons_2e83ff_256x240.png
lib/jquery-ui/themes/smoothness/images/ui-icons_454545_256x240.png
lib/jquery-ui/themes/smoothness/images/ui-icons_888888_256x240.png
lib/jquery-ui/themes/smoothness/images/ui-icons_cd0a0a_256x240.png
lib/jquery-ui/themes/smoothness/smoothness.css
lib/requirejs
lib/requirejs/require.js
lib/underscore
lib/underscore/underscore-min.js
manage
runner
runner/base.py
runner/base_test.py
runner/consts.py
runner/cxx.py
runner/cxx_test.py
runner/__init__.py
```

```
runner/misc.py
runner/php.py
runner/php_test.py
runner/python.py
runner/python_test.py
runner/server.py
runner/urls.py
runner/views.py
static
static/content
static/content/admin
static/content/css
static/content/css/common.css
static/content/css/test-detail.css
static/content/images
static/content/images/application-x-php.png
static/content/images/application-x-python.png
static/content/images/delete.png
static/content/images/exit.png
static/content/images/help.png
static/content/images/new.png
static/content/images/run.png
static/content/images/text-x-c++hdr.png
static/content/images/text-x-c++src.png
static/content/js
static/content/js/common
static/content/js/common/collections
static/content/js/common/collections/collection.js
static/content/js/common/models
static/content/js/common/models/model.js
static/content/js/common/views
static/content/js/common/views/view.js
static/content/js/config.js
static/content/js/main.js
static/content/js/test
static/content/js/test/app.js
static/content/js/test/collections
static/content/js/test/collections/file.js
static/content/js/test/collections/task.js
static/content/js/test/collections/test_task.js
static/content/js/test/models
static/content/js/test/models/file.js
static/content/js/test/models/task.js
static/content/js/test/models/test_task.js
static/content/js/test/views
static/content/js/test/views/clock.js
static/content/js/test/views/file.js
static/content/js/test/views/task.js
static/content/js/test/views/test_task.js
static/content/js/test/views/tools.js
static/content/lib
static/server.py
```

2. Tarkvara lähtekood

Bakalaureusetöös arendatud tarkvara lähtekood on saadaval tööga kaasas oleval CD-l ning aadressil http://www.tlu.ee/~karmux/src.zip

3. Näidisülesannete lahendused

Näidisülesannete lahendused programmeerimiskeeltes C++, PHP ja Python on saadaval tööga kaasas oleval CD-l ning aadressil http://www.tlu.ee/~karmux/lahendused.zip