Tartu Ülikool
Matemaatika-informaatikateaduskond

Java Unified Sharing System – J.U.S.S.

Teostajad: Henri Kuuste (FT it1)
Taimo Peelo (MT inf1)
Kaido Kalda (MT it1)
Vilen Looga (FT it1)
Olavi Püvi (FT it1)

Juhendaja: Anton Litvinenko

Tartu 2005

Sisukord

Ülesande püstitus **3**
Peamine eesmärk 3
Sihtgrupp 3
Kasutus 3
Lahenduse kirjeldus **5**
Eessõna 5
Kirjeldus 5
Tähtsamad süsteemi osad UML'is 6
Abstraktne ülevaade süsteemist 6
Ülesehitus 7
Klient 8
Server 9
Andmebaasid 10
Transpordihaldur (Transfer Manager) 11
Kasutajaliides 12
Kasutusjuhend **13**
Nõuded tarkvarale 13
Installeerimine 13
Klient 13
Server 13
Kasutamine 14
Tööprotsessi kirjeldus **15**
Grupisisene tööjaotus 15
Projekti ajagraafik 15
Hinnang projektile **17**
Kasutatud allikad **18**
Võrk 18
Andmebaasid 18
GUI 18
UML 18
XML 18
Üldine 18
Lisad **19**
Javadoc 19
Programmi kood 19
Autoriõigustest 19

Ülesande püstitus

➢ *Peamine eesmärk*

JUSS on eelkõige programm failide jagamiseks ja haldamiseks keskse serveri abil. Üldine põhimõte on natuke sarnane CVSile, kuid ette nähtud binaarsete failide jaoks ja versioonide tegemine on kasutaja otsustada (ei toimu automaatselt). Süsteem ei ole anonüümne - kõigil on kasutaja serveris ja kasutajaks saamist juhib serveri administraator. Eesmärgiks on mugav keskkond grupitöö lihtsustamiseks projektides kus suurem osa faile ei ole siiski ASCII failid. CVS ja teised sarnased süsteemid on meie kogemuste järgi selle kohapealt liiga aeglased. Tähtis on ka metadata süsteem mis võiks endas sisaldada kirjeldust, võtmesõnu ja ka kommenteerimise ja hindamise süsteemi. Nii on kergem inimestel neid huvitavaid faile üles leida.

Kasutaja peab saama korraga ühendada ennast mitme serveriga, seal faile brausida ja endale alla laadida. Samuti saab ta neid faile (ja ka katalooge) kommenteerida ja hinnata. Faili/kataloogi üles pannes on võimalik juurde panna märksõnad ja kirjeldus. Eksisteerib otsingusüsteem mis suudab kõige selle metadata põhjal faile/katalooge leida.

Serveri poolel eksisteerib siis kasutajate haldamise süsteem ja samuti ka projektide süsteem. Projekt on kasutajate ja failide kogum, mis grupeerib failid ühte loogilisse üksusesse (visuaalselt siis kataloogi) ja kus kõigil sinna kuuluvatel kasutajatel on mingid kindlad õigused. Nendel kes pole projekti liikmed pole isegi õigusi projekti sisu näha, kui just projekti kasutajaks pole ka "Everyone".

Üldiselt kohalik arvuti peegeldab kataloogi struktuuri serveris, kuid ainult niivõrd kui kasutaja on enda arvutisse ära tõmmanud. Kommentaarid ja muu metadata salvestatakse ka kohalikku (kliendi) andmebaasi, et oleks seda kiiremini võimalik brausida. Toimuvad kontrollid lokaalse ja serveri andmebaasi sünkroonis hoidmiseks. Kui kasutaja pole online, on tal ikka võimalik brausida seda osa struktuurist, mida on varem vaadanud (kuigi info ei pruugi olla kõige uuem).

➢ *Sihtgrupp*

Sihtgrupiks on eelkõige grupid ja asutused, kes vajavad eelkõige mitte-ascii failide jagamist ja nende versioonide haldamist. Näiteks siis programmeerimise projektid, kus on vaja hallata peale koodi veel suurt hulka meedia faile.

➢ *Kasutus*

Kasutaja saab teha järgmiseid asju:

- Luua kohalikke projekte, mis asuvad kõvakettal määratud kohas ja on seotud kindla serveriga
- Avada korraga mitu projekti. Need on GUI-s näha puu kujul
- Luua serveris uusi projekte, millel on kindlad liikmed ja kindel andmete kogum. Projekti loonud kasutaja saab edaspidi ka sellele projektile kasutajaid juurde lisada ja nende õigusi hallata.
- Lisada mingisse serveri poolsesse projekti faile/katalooge. Lisades on võimalik määrata ka kirjeldus ja märksõnad otsingumootorile.

- Hinnata serveris olevaid faile/katalooge.
- Kommenteerida serveris olevaid faile/katalooge.
- Tõmmata endale kõvakettale serveris olevaid faile/katalooge.
- Lisada olemasolevale failile uus versioon
- Otsida faile/katalooge märksõnade, kommentaaride, hinnete ja failinimede järgi

Lahenduse kirjeldus

➢ *Eessõna*

Programmil eksisteerib reaalne kasutajategrupp, kes hakkaks seda kasutama niipea kui asi valmis saab. Leitud on ka asjast huvitatud inimesed, kes programmi edasi arendavad, pärast seda kui see koolis esitletud ja hinnatud saab. Ajalistest piirangutest tulenevalt ei seatud seega eesmärgiks kõikide püstituses esitatud ülesannete kohest lahendamist (funktsionaalsust saab täiendada ka hiljem), vaid sellise baassüsteemi loomist mis peaks vastu töö käigus lisanduda võivatele nõuetele, ning mille edasiarendamine oleks mugav.

➢ *Kirjeldus*

Lahendusel proovisime pidada silmas süsteemi robustsust ja head töökiirust paljude kasutajate korral. Programmi võrguosa jaoks otsustasime seepärast kasutada testitud võrgutehnoloogiat – HTTP-d. Samuti otsustasime kasutada andmebaase nii serveri kui kliendis - sest nii serveris kui kliendis on vaja häid ning kiireid otsinguvõimalusi. Lisaks võimaldab kliendipoolse andmebaasi lihtsat *offline-browsingu* realiseerimist.

HTTP protokolli realiseerimiseks võtsime serveris kasutusele Tomcat veebiserveri ja Java Servlet'ide süsteemi, kliendis aga *apache-commons* paketis sisalduva *HttpClienti*. Serveri ülesseadmine võib osutuda seetõttu üsna komplitseerituks – ent vähegi oma tööd tundval võrguadmnistraatoril peaks see siiski küllalt kiirelt minema.

Andmebaasisüsteemiks sai serveris valitud *MySQL*. Kuna andmebaasiga suhtluseks on nii serveris kui kliendis kasutatud *Hibernate* ORM lahendust, pole serveripoolse andmebaasi valik tegelikult piiratud – paari konfiguratsioonirea muutmisega saab serveris kasutusele võtta mõne teise andmebaasisüsteemi. Kliendipoolselt oleks eraldi andmebaasi kasutamine olnud üleliigne ning tavakasutaja jaoks ilmselt sageli ka ülejõukäiv – seega valisime kliendipoolseks andmebaasilahenduseks programmi sisseehitatud HSQLDB.

Kasutajaliidesele esitatavad nõuded olid hea väljanägemine, sulandumine ülejäänud töökeskkonnaga, kiirus. Nende eesmärkide realiseerimise vahendiks valisime *SWT*-toolkiti.

Et süsteemi oleks võimalik mugavalt konfigureerida on kasutatud XMLil põhinevaid konfiguratsioonifaile – nende lugemist ja kirjutamist realiseerib klass *CConfig* mis kasutab XMLi manipuleerimiseks pakette *dom4j* ning *jaxen*

Tulemus ei ole kindlasti momendil optimaalne ega isegi mitte päris stabiilne. Siiski võib seda lugeda tulevase süsteemi prototüübiks, valitud komponentide efektiivsemal ärakasutamisel on võimalik süsteemi võimekust veel kõvasti tõsta.

Tegemata jäid praegu kommenteerimine/hindamine ja kasutajatehaldus. Programmi põhiliste töömeetodite demonstreerimiseks polnud need ka vajalikud.

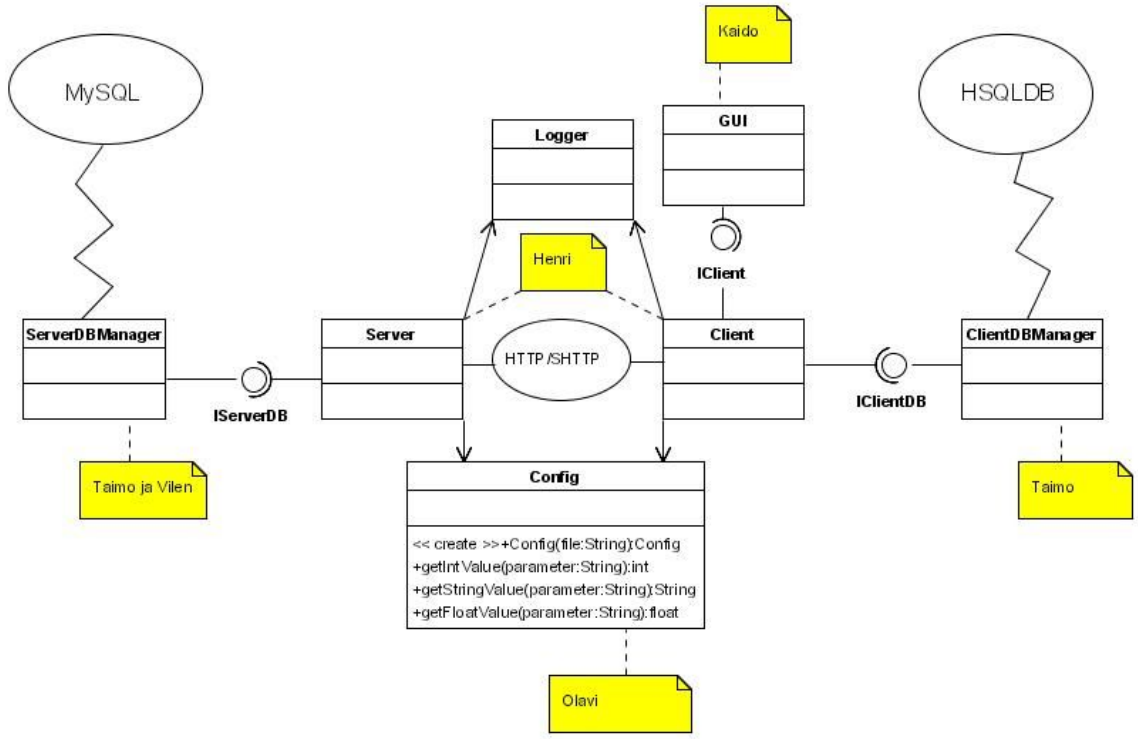
Mingeid erilisi algoritme programmis ei kasutatud. Andmestruktuuridest leidsid sagedast kasutamist erinevad *List*id ning *Map*id.

Väga oluline oli programmis lõimede kasutamine. Andmebaaside üheaegne kasutamine paljude erinevate lõimede poolt lisas süsteemi keerukust. Suurem osa turvaprobleeme ja vigasid tulenevad just sellest et paljud lõimed võivad üheaegselt kasutada sama

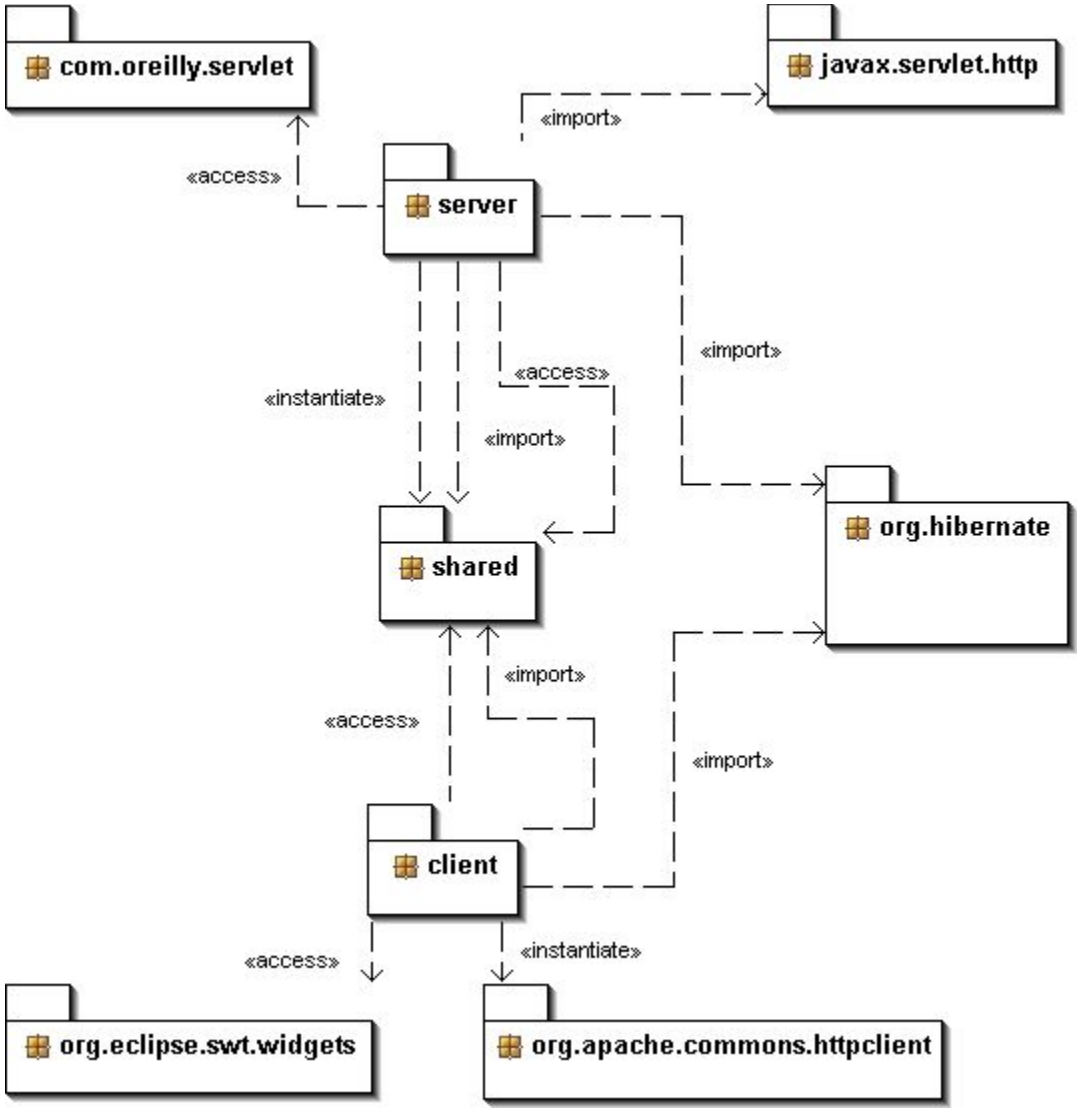
andmebaasi. Lahenduseks on hetkel see, et iga andmebaasiga suhtlev lõim kasutab oma sessiooni. Kliendis on paljud DBManager'i meetodid ka üle kaetud ja muudetud sünkroniseerituks. See langetab küll andmebaasioperatsioonide kiirust, kuid kliendipoolselt ei ole see tähtis, sest andmebaasiga tegeleb korraga vaid üks inimene. Arvatavasti peab andmebaasi samaaegse kasutamise süsteeme tulevikus veel natuke täiustama.

- ***Tähtsamad süsteemi osad UML'is***

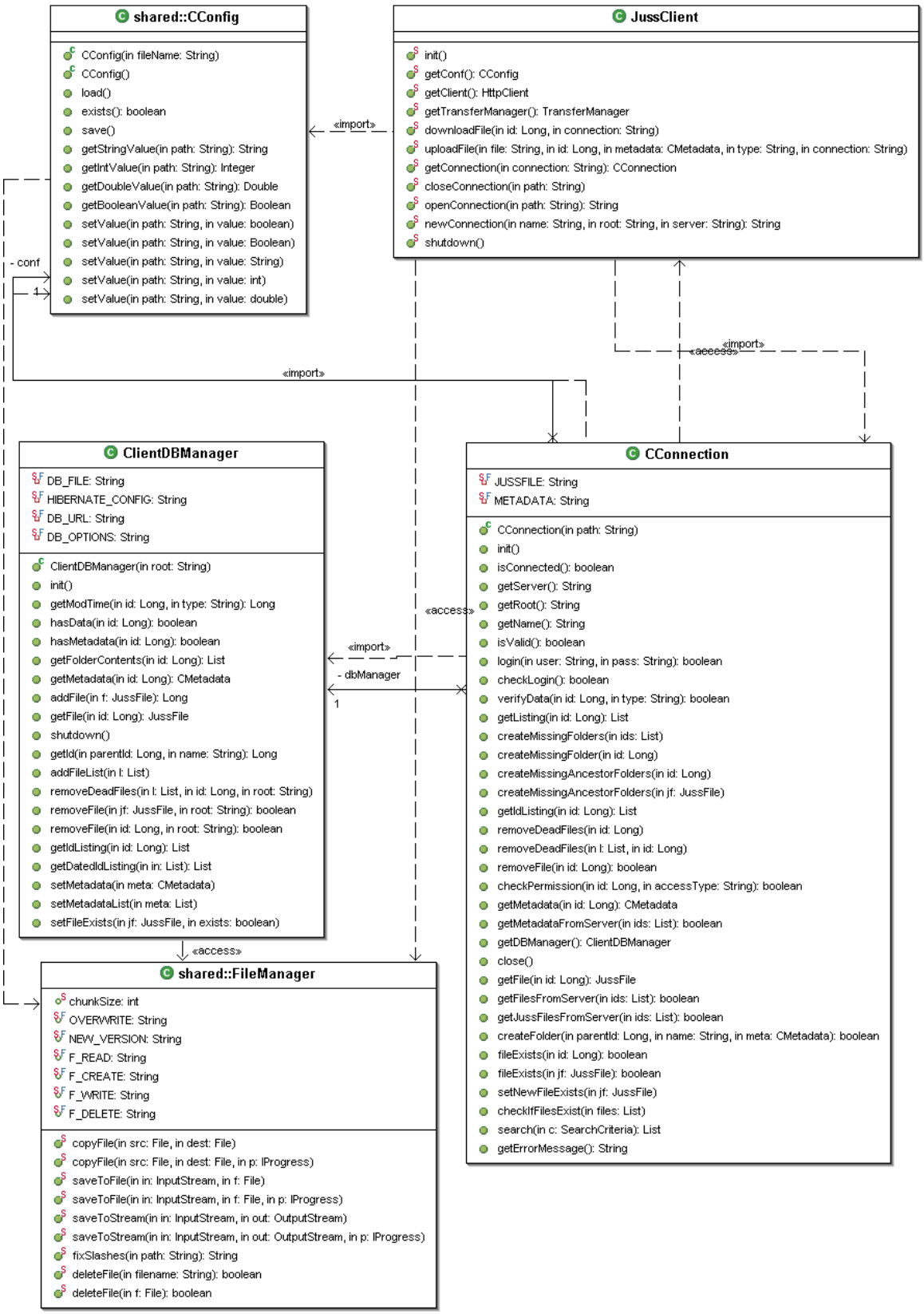
Abstraktne ülevaade süsteemist



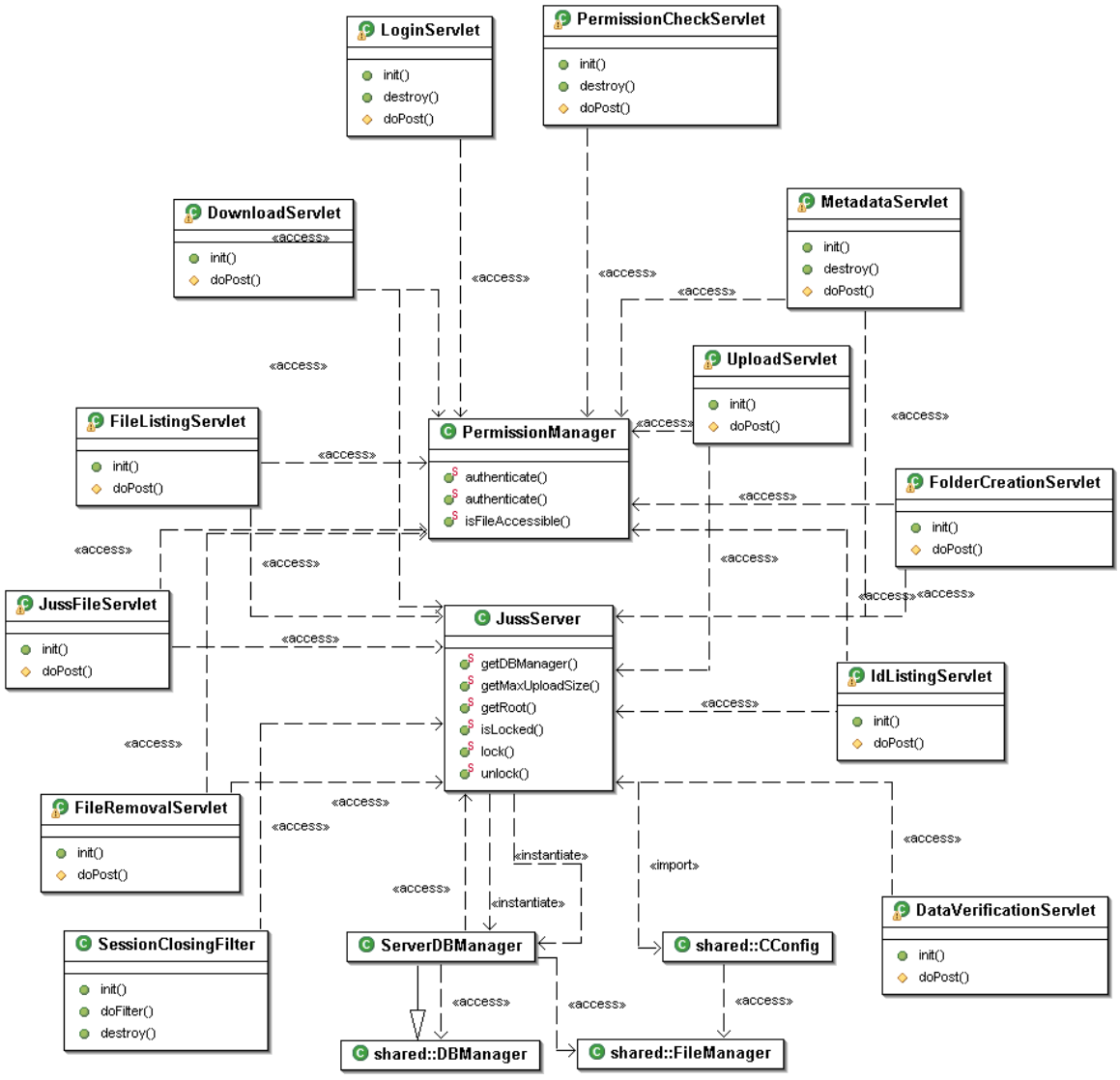
Ülesehitus


com.oreilly.servlet
javax.servlet.http
«import»
«access»
server
«import»
«instantiate»
«access»
«import»
org.hibernate
shared
«import»
«access»
«import»
client
«access»
«instantiate»
org.eclipse.swt.widgets
org.apache.commons.httpclient

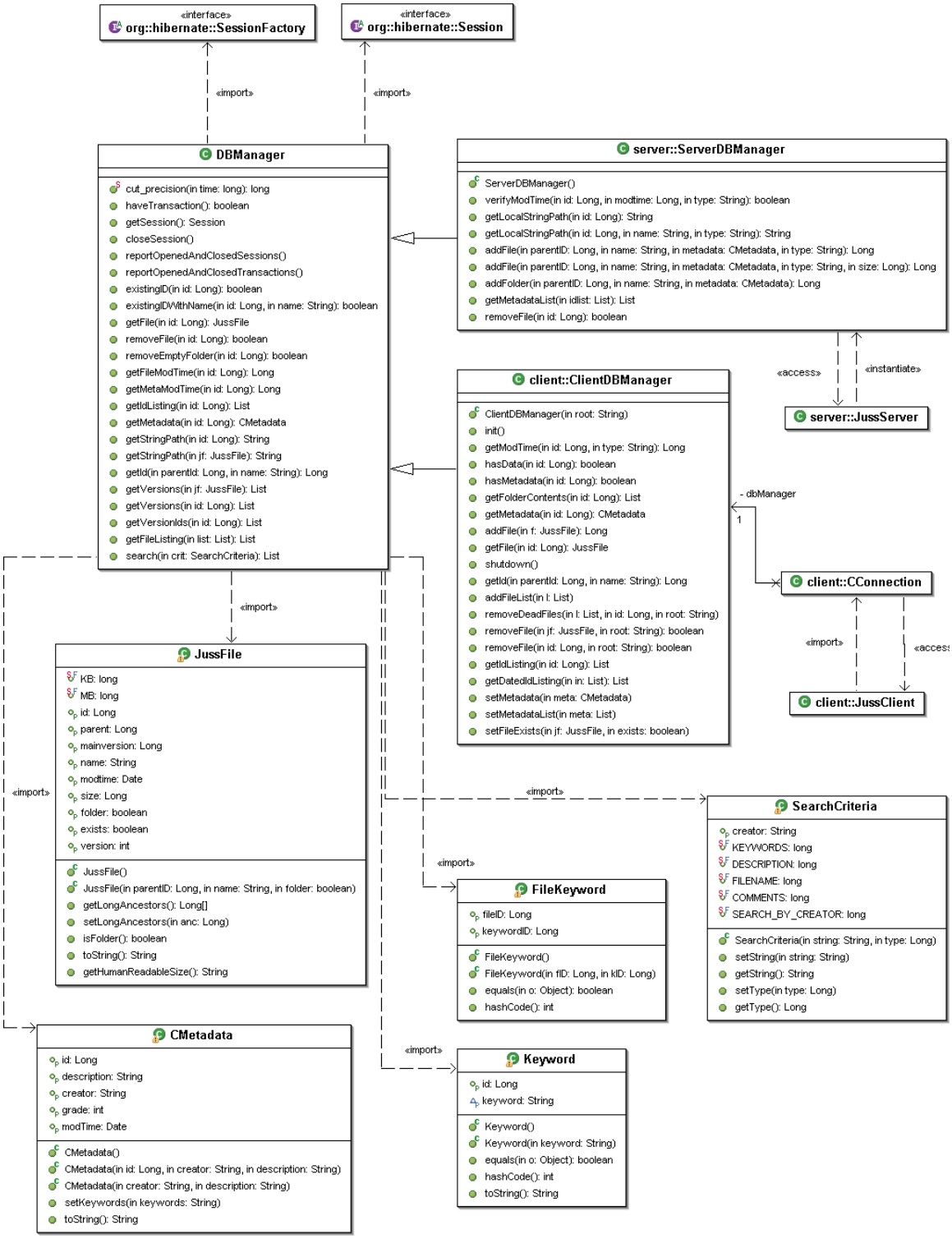
Klient


shared::CConfig
CConfig(in fileName: String)
CConfig()
load()
exists(): boolean
save()
getStringValue(in path: String): String
getIntValue(in path: String): Integer
getDoubleValue(in path: String): Double
getBooleanValue(in path: String): Boolean
setValue(in path: String, in value: boolean)
setValue(in path: String, in value: Boolean)
setValue(in path: String, in value: String)
setValue(in path: String, in value: int)
setValue(in path: String, in value: double)
JussClient
init()
getConf(): CConfig
getClient(): HttpClient
getTransferManager(): TransferManager
downloadFile(in id: Long, in connection: String)
uploadFile(in file: String, in id: Long, in metadata: CMetadata, in type: String, in connection: String)
getConnection(in connection: String): CConnection
closeConnection(in path: String)
openConnection(in path: String): String
newConnection(in name: String, in root: String, in server: String): String
shutdown()
«import»
- conf
1
«import»
«aeeeess»
ClientDBManager
DB_FILE: String
HIBERNATE_CONFIG: String
DB_URL: String
DB_OPTIONS: String
ClientDBManager(in root: String)
init()
getModTime(in id: Long, in type: String): Long
hasData(in id: Long): boolean
hasMetadata(in id: Long): boolean
getFolderContents(in id: Long): List
getMetadata(in id: Long): CMetadata
addFile(in f: JussFile): Long
getFile(in id: Long): JussFile
shutdown()
getId(in parentId: Long, in name: String): Long
addFileList(in l: List)
removeDeadFiles(in l: List, in id: Long, in root: String)
removeFile(in jf: JussFile, in root: String): boolean
removeFile(in id: Long, in root: String): boolean
getIdListing(in id: Long): List
getDatedIdListing(in in: List): List
setMetadata(in meta: CMetadata)
setMetadataList(in meta: List)
setFileExists(in jf: JussFile, in exists: boolean)
«access»
«import»
- dbManager
1
«access»
CConnection
JUSSFILE: String
METADATA: String
CConnection(in path: String)
init()
isConnected(): boolean
getServer(): String
getRoot(): String
getName(): String
isValid(): boolean
login(in user: String, in pass: String): boolean
checkLogin(): boolean
verifyData(in id: Long, in type: String): boolean
getListing(in id: Long): List
createMissingFolders(in ids: List)
createMissingFolder(in id: Long)
createMissingAncestorFolders(in id: Long)
createMissingAncestorFolders(in jf: JussFile)
getIdListing(in id: Long): List
removeDeadFiles(in id: Long)
removeDeadFiles(in l: List, in id: Long)
removeFile(in id: Long): boolean
checkPermission(in id: Long, in accessType: String): boolean
getMetadata(in id: Long): CMetadata
getMetadataFromServer(in ids: List): boolean
getDBManager(): ClientDBManager
close()
getFile(in id: Long): JussFile
getFilesFromServer(in ids: List): boolean
getJussFilesFromServer(in ids: List): boolean
createFolder(in parentId: Long, in name: String, in meta: CMetadata): boolean
fileExists(in id: Long): boolean
fileExists(in jf: JussFile): boolean
setNewFileExists(in jf: JussFile)
checkIfFilesExist(in files: List)
search(in c: SearchCriteria): List
getErrorMessage(): String
shared::FileManager
chunkSize: int
OVERWRITE: String
NEW_VERSION: String
F_READ: String
F_CREATE: String
F_WRITE: String
F_DELETE: String
copyFile(in src: File, in dest: File)
copyFile(in src: File, in dest: File, in p: IProgress)
saveToFile(in in: InputStream, in f: File)
saveToFile(in in: InputStream, in f: File, in p: IProgress)
saveToStream(in in: InputStream, in out: OutputStream)
saveToStream(in in: InputStream, in out: OutputStream, in p: IProgress)
fixSlashes(in path: String): String
deleteFile(in filename: String): boolean
deleteFile(in f: File): boolean

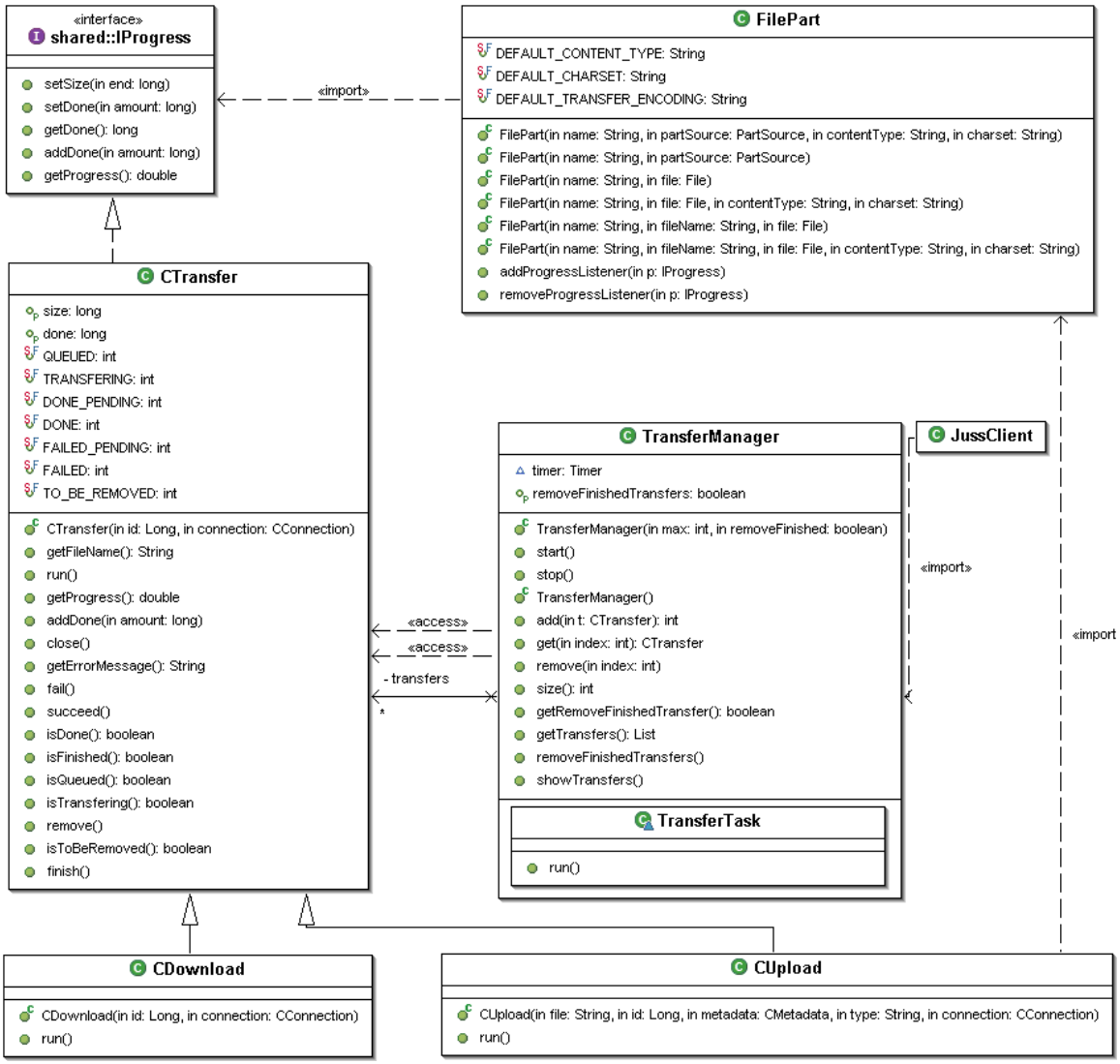
Server


LoginServlet
init()
destroy()
doPost()
PermissionCheckServlet
init()
destroy()
doPost()
DownloadServlet
«access»
init()
doPost()
MetadataServlet
init()
destroy()
doPost()
UploadServlet
init()
doPost()
FileListingServlet
init()
doPost()
PermissionManager
authenticate()
authenticate()
isFileAccessible()
FolderCreationServlet
init()
doPost()
JussFileServlet
init()
doPost()
JussServer
getDBManager()
getMaxUploadSize()
getRoot()
isLocked()
lock()
unlock()
IdListingServlet
init()
doPost()
FileRemovalServlet
init()
doPost()
DataVerificationServlet
init()
doPost()
SessionClosingFilter
init()
doFilter()
destroy()
ServerDBManager
shared::CConfig
shared::DBManager
shared::FileManager
«access»
«instantiate»
«import»

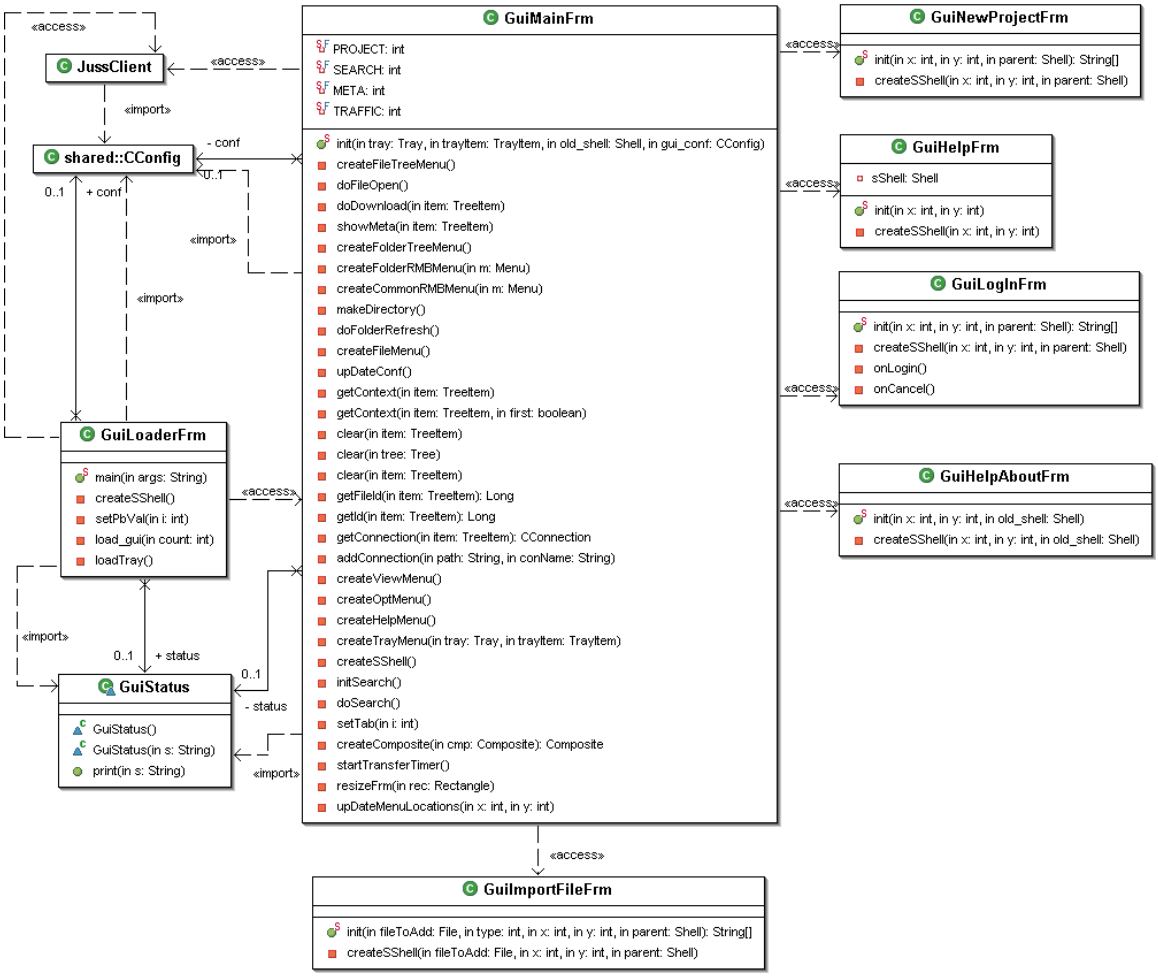
Andmebaasid



Transpordihaldur (Transfer Manager)



Kasutajaliides



Kasutusjuhend

➢ *Nõuded virtuaalmasinale*

Testitud on programmi esialgu Windowsis ja Linuxis Java versiooniga 1.4.2 ja 1.5.0. Võimalik, et programm töötab ka natuke varasemate Java versioonidega, aga meie seda hetkel kinnitada ei saa. JUSSi jooksutamiseks soovitame kas Sun'i VM'i või Blackdowni VMi. Testisime ka populaarsete VM'idega Kaffe ja SableVM, kuid avastasime, et nendega ei tööta. Põhjuseks eelkõige kliendipoolses andmebaasis kasutatava HSQLDB kõrged nõuded virtuaalmasinal. Võimalik, et server nende VM'ide peal siiski töötaks, seda veel testitud pole.

➢ *Installeerimine*

Klient

Vilen peab veel installika üles panema.

Ajutiselt on olemas JUSS'i binary:
http://ats.cs.ut.ee/oop2005/juss/releases/juss_distro.tar.gz kus asub ka klient. Käima panemiseks tuleks üldiselt kasutada run_old.bat/run_old.sh. run_new.bat/run_new.sh paneks käima uue GUI kuid seal on funktsionaalsust palju puudu ja esitlusele läheb just see vanem GUI.

Server

Serveri installeerimine on praegu üsna keerukas. Kõigepealt tuleks installeerida ise JRE (http://java.sun.com/), MySQL (http://www.mysql.com/products/mysql/) ja Tomcat (http://jakarta.apache.org/tomcat/). Edasi tuleks kas muretseda endale JUSS'i binary (http://ats.cs.ut.ee/oop2005/juss/releases/juss_distro.tar.gz) või kompileerida ise süsteem CVS'ist (vt. Lisad). Seejärel tuleks MySQL'is teha järgmised toimingud:

```
CREATE DATABASE jussdb;
```

```
GRANT ALL PRIVILEGES
ON jussdb.*
TO 'juss'@'localhost'
IDENTIFIED BY 'goodpassword';
```

Kus 'localhost' on teie serveri kohalik nimi. 'goodpassword' on parool mida juss kasutama peaks. Kui soovite kasutajanime või parooli siin muuta siis peate tegema ka vastavad muudatused WEB-INF/classes/hibernate.cfg.xml failis. Samuti kui teie MySQL port ei ole standardne või teie MySQL server ei asu samas arvutis JUSS serveriga, siis on teil ka vaja muuta just seda hibernate.cfg.xml faili.
Edasi tuleks konfigureerida Tomcat. Selleks on kaasas väike programm mille saate käivitada kasutades script'i make_conf.bat/make_conf.sh.
Selle süntaks on järgmine:
make_conf -m {Maksimaalne uploadi suurus baitides} **-t** {Tomcat'i asukoht} **-s** {kataloog kus asub JUSS'i WEB-INF} **-r** {Kataloog kuhu panna JUSS'i kasutajate poolt uploaditud andmed}

Näiteks Windowsis:
make_conf.bat -m 20000000 **-r** "C:\server_root" **-s** "C:\juss distro\Server" **-t** "C:\Tomcat 5.5"

Nüüd peaks Tomcat'i käivitudes käivituma ka JUSSi server.
Server asub aadressil http://hostname:tomcat_port/juss/

➢ *Kasutamine*

Täieliku kasutusjuhendi leiab aadressilt:

http://ats.cs.ut.ee/oop2005/juss/Juss_Manual.pdf

Tööprotsessi kirjeldus

➢ *Grupisisene tööjaotus*

- Henri Kuuste (juht)
  Võrk, UML, kliendi andmebaas, aruanne, javadoc.
- Taimo Peelo
  Andmebaasid, javadoc.
- Kaido Kalda
  GUI, javadoc, projekti hinnang.
- Vilen Looga
  XML konfiguratsioonisüsteem, installer, kasutusjuhend.
- Olavi Püvi
  Algne I/O uurimine. Algselt ka XMLil põhineva konfiguratsioonisüsteemi loomine, kuid probleemide tõttu arvutiga läks see ülesanne Vilenile. Esitluse materjalid ja esitlus.

➢ *Projekti ajagraafik*

Esialgne rühm sai loodud kohe peale projekti tegemise nõudest teada saamist. Kuid paari nädala jooksul rühma koosseis muutus ja sinna jäi ainult 2 esialgset liiget. Algne idee oli teha mingit tüüpi arvutimäng, sest rühma juhil oli sellealaseid kogemusi. Lõpuks otsustati siiski praktilise rakenduse kasuks. Otsustati kokku saada iga reedel, kui vähegi võimalik.

25.02

Selleks koosolekuks pidi igaüks välja käima vähemalt ühe korraliku idee praktilise rakenduse kohta. Koosoleku lõpus selgus et eelkõige huvitab inimesi elektroonika simulatsiooni ja modelleerimise programm. Asja tegi huvitavaks veel see, et sellekohaseid prii- või vabavaraprogramme ei teatud. Kuna osad grupiliikmed arvasid, et asi on ehk liiga keeruline ja Kaido ütles et on midagi sarnast tegelikult siiski näinud kah, siis lükati otsustamine edasi, kuni asja põhjalikuma uurimiseni.

04.03

Leidsime siiski leidnud paar tasuta elektroonikasimulatsiooni programmi ja otsustasime üritada teha teist populaarseimat ideed - failide jagamise/versioonikontrolli süsteemi. Järgmiseks korraks pidi välja mõtlema mingi nime ja uurima tehnoloogiaid mida kasutada võiks. Võrgulahenduseks oli algselt plaanis kasutada Remote Method Invocation'it (RMI'd), millega oleks kerge vajaduse tekkides lisada *peer-to-peer networking support.*

11.03

Otsustasime et projekti nimeks saab JUSS ehk Java Unified Sharing System. Hakkasime jagama ülesandeid ja saatsime avalduse saada ATS keskkonna kasutajaks. Sai loodud IRC kanal kiiremaks suhtlemiseks.

18.03

Koosolek jäi ära.

24.03

Saatsime ära rühma nimekirja.

25.03

Hakkasime planeerima GUI väljanägemist paberil. Inimesed raporteerisid oma ülesannet puudutavate tehnoloogiate kohta.

30.03

Saime lõpuks ATSi (ats.cs.ut.ee) serveris CVSi kasutamise õiguse.

01.04

Panime CVS'i ülesandepüstituse ja ka projekti üldise ülesehituse UML'i. Otsustasime kasutada HTTP tehnoloogiat võrgu jaoks, sest RMI'ga enda serveri tegemine võtaks liiga kaua aega ja ei oleks mõttekas. Parem kasutada robustset äraproovitud tehnoloogiat mis areneb meist sõltumatult. Sai loodud ka projekti kodulehekülg.

06.04-18 04

Kiired ajad projektiväliselt. Projekti kallal töö aeglane.

08.04

Algab töö serveri ja andmebaasi süsteemide disaini kallal.

10.04

Oli tekkinud mingi segadus ja lõpuks taipasime püstituse ka juhendajale saata.

20.04

Esimene backend süsteem CVS'is.

20.04-29.04

Aktiivne backend süsteemi programmeerimine.

30.04

Esimene GUI CVS'is. Algab aktiivne integreerimisprotsess.

01.04

Töötav süsteem saadetud juhendajale. Süsteem pole küll päris valmis veel aga töötab mingil määral. Algab aktiivne dokumentatsiooni (eelkõige javadoc) kirjutamine ja süsteemi viimistlemine. Otsustame hetkel ohverdada funktsionaalsuse lisamise selle nimel, et juba olemasolev süsteem paremini tööle panna.

Hinnang projektile

Kaido: „Hiline ühildamistöö ettevõtmine seadis ülesse suured kahtlused, et projekt valmib õigeaegselt. Sellest hoolimata sujus kõik suurepäraselt ning projekti võib lugeda igati õnnestunuks. Suurem osa püstituses eesmärgiks seatud ülesannetest lahendasime tänu suurepärasele grupikoostööle, kuigi tegemata jäi kliendipoolne kasutajateõiguse korraldaja. Sellest hoolimata suutsime viia kokku põhifunksionaalsuse seisukohalt püstituse ja tulemuse.

Tööjaotus oli üles seatud vastavalt võimetele ja kõik tulid oma osaga väga hästi toime. Kuigi mõnele osapoolele langes rohkem ning raskeimaid ülesandeid, sellest hoolimata keegi ei jooksnud peaga vastu seina, vaid leidsime ühisel jõul probleemile lahenduse.

Kogemus oli enneolematu, kuna kasutasime professionaalseid tarkvaraarenduse meetodeid. Tasemel programmeerijad tegid väga head koostööd ning tulemuseni jõuti üheskoos.

Sellest hoolima tekkisid ka mitmed lahkarvamused, kuigi sellest hoolimata meeskonnatöö sellepärast ei kannatanud. Hea mõistmine, ning endi seisukohtade tahaplaanile jätmine ning arvestamine kõigi arvamusega andis tulemuseks väga hea meeskonnatöö.

Projekt ise tuli väga ootuspärane, ning eeldame väga suurt kasutajaskonda. Kuigi tuleks teha veel ühildustööd erinevate platformide vahel. Seetõttu on meil kindel plaan seda projekti jätkuvalt edasi arendada ning kaasata helgeid päid ka mujalt.

Kokkuvõtteks jättis kogu projekt ja grupitöö väga positiivse mulje ning seda võib nimetada igati õnnestunuks.“

Kasutatud allikad

- ***Võrk***

http://jakarta.apache.org/commons/httpclient/index.html
http://java.sun.com/products/servlet/docs.html
http://www.servlets.com/
http://jakarta.apache.org/tomcat/
http://www.frontiernet.net/~imaging/servlets_intro.html

- ***Andmebaasid***

http://hsqldb.sourceforge.net/
http://www.hibernate.org/

- ***GUI***

http://www.eclipse.org/swt/

- ***UML***

http://www.omondo.com/product.html

- ***XML***

http://dom4j.org/

- ***Üldine***

http://java.sun.com/j2se/1.4.2/docs/api/

Lisad

- ***Javadoc***

http://ats.cs.ut.ee/oop2005/juss/javadoc/index.html

- ***Programmi kood***

http://ats.cs.ut.ee/tvp2005/viewcvs/oop/juss/

Või kui eksisteerib kasutaja ATS'is siis saab teha CVS checkout'i:

:pserver:{KASUTAJA}@krokodell.at.mt.ut.ee:2401/projects/home/tvp2005/cvsroot
CVS projekt:
oop/juss

- ***Autoriõigustest***