BACHELOROPPGAVE:

SWITCHBOARD

FORFATTER(E): TRON LØVÅS
ANDERS FLISVANG NYEN
STIG RUNE AASEN

Dato: 15. mai 2013

Sammendrag

Tittel:	Switchboard	
		Dato: 15.05.13
Deltaker(e):	Tron Løvås	
	Anders Flisvang Nyen	
	Stig Rune Aasen	
Veileder(e):	Tom Røise	
Oppdragsgiver:	Jon Langseth	
Kontaktperson:	Tron Løvås	
Stikkord:	Java, LDAP, CalDAV, Sentralbord	

Antall sider: 65	Antall vedlegg: 10	Tilgjengelighet: Åpen

Kort beskrivelse av bacheloroppgaven:

Switchboard-applikasjonen bygger på en eksisterende applikasjon utviklet av UNINETT. Applikasjonen brukes til å viderekoble innkommende samtaler til ønsket person via en Asterisk og Kamailio SIP server. Gjennom prosjektperioden har vi arbeidet for å endre måten applikasjonen henter kontakt- og kalenderinformasjon fra å bruke MySQL og PostgreSQL, til protokollene LDAP og CalDAV. Vi greide dessverre ikke å fullføre CalDAV modulen før tidsfristen.

Summary

Title:	Switchboard	
		Date: 15.05.13
Participants:	Tron Løvås	
	Anders Flisvang Nyen	
	Stig Rune Aasen	
Supervisor:	Tom Røise	
Employer:	Jon Langseth	
Kontaktperson:	Tron Løvås	
Keywords:	Java, LDAP, CalDAV, Switchboard	

Number of pages: 65	Number of appendix: 10	Availability: open

Short description of the bachelor thesis:

The switchboard application is based on an existing application developed by UNINETT. The application is used to forward incoming calls to the desired person by an Asterisk and Kamailio SIP server. Throughout the project we've worked on modifying the part of the application that fetches contact- and calendar-information, from using MySQL and PostgreSQL, to using the LDAP and CalDAV protocols. We were unable to finish the CalDAV module before the deadline.

Forord

Denne rapporten er utarbeidet som et bachelorprosjekt ved Høgskolen i Gjøvik, og er avslutningen på et treårig bachelorstudium i ingeniørfag – data.

Da prosjektoppgavene ble lagt ut ved HiG november 2012, begynte vi å danne en prosjektgruppe med felles interesse og ambisjon. Etter samtale med Hilde Bakke fikk vi høre om en prosjektoppgave som enda ikke var utgitt, laget av IT-tjenesten ved HiG. Kontaktperson for oppgaven var Jon Langseth.

Oppgaven tok for seg flere temaer som programmeringsspråkene Java og SQL, IP-telefoni og utveksling av data via protokollene LDAP og CalDAV, samt at oppgaven tok utgangspunkt i en eksisterende applikasjon. Gruppemedlemmene hadde lite eller ingen erfaring med disse temaene, og vi følte det var en spennende og utfordrende prosjektoppgave.

Vi ønsker å rette en stor takk til veileder gjennom prosjektet, Tom Røise. Tom har gjennom hele prosjektperioden gitt veldig gode tilbakemeldinger, råd og veiledning.

Vi vil også rette en takk til oppdragsgiver, Jon Langseth. Gjennom prosjektet har Jon vært lett tilgjengelig for spørsmål, kommet med tips til arbeidet og raskt tilskaffet servere som var nødvendig for å gjennomføre oppgaven.

Table of Contents

Table of Figures

Kapittel 1 Introduksjon

1.1 Formål med prosjektoppgaven

Høgskolen i Gjøvik er allerede i gang med å fornye sitt interne telefonsystem, fra tradisjonell telefoni (*PBX*) til en ny struktur (*VoIP*), basert på IP-telefoni. Med disse endringene følger det et behov for en ny switchboard-applikasjon.

HiG ønsket ved prosjektoppgaven å anskaffe en åpen switchboard-applikasjon, tilpasset deres interne telefoni-system. Som utgangspunkt for prosjektoppgaven fikk oppdragsgiver tilgang til en løsning utviklet av UNINETT AS. Denne løsningen ble utviklet for intern bruk ved UNINETT, og er derfor tilpasset deres system.

1.2 Prosjektmål

1.2.1 Effektmål

Med dette prosjektet ønsker Høgskolen i Gjøvik å kunne fullstendig erstatte den gamle PBX løsningen til fordel for en mer moderne VoIP løsning. Som årsak av dette vil Høgskolen spare tilnærmet 200 000,- årlig.

1.2.2 Resultatmål

Hovetmålet med prosjektet var å lage en åpen, fungerende switchboard-applikasjon som inneholder all nødvendig funksjonalitet, men er åpen for utvidelser og/eller modifikasjoner.

1.2.3 Læringsmål

Etter å ha jobbet med dette prosjektet skal vi ha innarbeidet oss kunnskaper i java, hvordan switchboard-applikasjoner er bygd opp, ip-telefoni med fokus på hvordan *Asterisk* og *Kamailio* fungerer i praksis, og hvordan man kan hente data fra en ekstern server ved bruk av LDAP og CalDAV. Vi skal også ha lært hvordan man går frem for å videreutvikle en ferdig applikasjon.

1.3 Problemområde

Når man hører ordet 'sentralbord' dannes det gjerne et svart-hvitt bilde av en lang rekke voksne damer som manuelt kobler sammen samtaler ved å flytte plugger rundt omkring på en stor tavle. I dag er tavlene og pluggene erstattet med en sammensetning av kodelinjer, og behovet for bemanning langt mindre.

Det kan sies at dagens switchboard-applikasjoner er selve knutepunktet for moderne bedrifter. Operatørene tar imot alle innkommende samtaler, og viderefører samtalene til rett person ved hjelp av applikasjonens grensesnitt. Dagens sentralbord-applikasjoner kan inneholde mye forskjellig funksjonalitet.

Den mest sentrale funksjonaliteten i en switchboard-applikasjon er håndtering av samtaler. En operatør må effektivt kunne ta imot samtaler, finne frem til rett person, kontrollere at gitt person er tilgjengelig og deretter videreføre samtalen.

Det finnes i dag en rekke forskjellige sentralbord-løsninger. De fleste løsningene er utviklet av diverse firmaer som arbeider med utvikling av programvare, og bruk av løsningene medfører ofte et abonnement eller en lisens. Samtidig er disse løsningene ofte tett knyttet mot et spesielt databasesystem. Dette har ført til ønsker om en open-source switchboard-applikasjon, som kan konfigureres til å passe for forskjellige databasesystem.

1.4 Oppgavedefinisjon

Hensikten med prosjektet var å gjøre endringer på UNINETT's interne switchboard-applikasjon. Endringene skulle gjøre applikasjonen kompatibel med HiG's telefoni-system, og tilby en åpen sentralbord-løsning, som er lett å tilpasse for andre organisasjoner som bruker SIP telefoni.

Hoveddelen av oppgaven bestod av å bytte ut den eksisterende løsningen for å hente kalender- og kontakt-informasjon. UNINETT's løsning hentet både kalender- og kontakt-informasjon fra en postgreSQL server, hvor databasen var satt opp etter deres behov, og følger ingen standardformat.

HiG benytter seg av en exchange server, hvor kontakt-informasjon for de ansatte er lagret i *AD* (*Active Directory*), og kalender-informasjonen lagres sammen med brukernes e-postkonto. Kalenderne lagret her følger standardformatet til iCaldendar. For å gjøre applikasjonen kompatibel med HiG's system måtte delene av applikasjonen som samhandler med postgreSQL databasen endres til å samhandle med HiG's exchange server.

Kommunikasjonen mellom applikasjonen og exchange serveren foregår over protokollene LDAP og CalDAV, for henholdsvis kontakt-informasjon og kalender-informasjon. Beslutningen å bruke disse protokollene ble tatt av oppdragsgiver i forkant av prosjektet.

For å gjøre videreutvikling av applikasjonen lettere, har vi utarbeidet javadocs for applikasjonens klasser og funksjoner og både ny og gammel kildekode måtte kommenteres.

Oppdragsgiver ønsket at den ferdige løsningen skulle være mulig å legge ut som open source. Det vil si at alle biblioteker som ble brukt i utviklingen måtte være lisensiert for å tillate dette.

For å kunne gjennomføre endringene i oppgaven var det nødvendig å sette seg godt inn i, og forstå kildekoden fra UNINETT. Dette vil være essensielt for våre endringer, slik at de utvikles etter samme struktur som de resterende delene av koden.

1.5 Avgrensning

For å kunne legge ut applikasjonen som open source, må alle biblioteker og programvare som brukes i utvikling av applikasjonen skal være uten GPL(GNU General Public License).

1.6 Målgruppe

Oppgaven ble primært utarbeidet for å utvikle en switchboard-applikasjon for bruk ved HiG. Men siden oppdragsgiver ønsket åpen kildekode for applikasjonen, vil også andre organisasjoner eller privatpersoner med behov for en åpen switchboard-applikasjon være en del av målgruppen.

Rapportens målgruppe vil hovedsakelig være sensor og oppdragsgiver, men rapporten vil også være aktuell for studenter som vil lære mer om hvordan man skal angripe det å endre på en allerede fungerende applikasjon, og/eller ønsker et innblikk i hvordan en switchboard-applikasjon er oppbygd.

1.7 Bakgrunn og kompetanse

Prosjektgruppen bestod av tre studenter som studerte bachelor i ingeniørfag – data, ved Høgskolen i Gjøvik. Tron Løvås, Anders Flisvang Nyen og Stig Rune Aasen. Alle gruppens medlemmer hadde tidligere erfaring med programmering i C++, men lite erfaring med java, da Stig Rune var det eneste gruppemedlemmet som har gjennomført kurs i java.
Gruppen hadde også lite erfaring med å arbeide på større prosjekter.

1.8 Rammer

Prosjektet hadde fastsatt tidfrist for innlevering av prosjektrapport den 15. mai 2013.

Utviklingen av UNINETT's applikasjon har foregått i NetBeans. Oppdragsgiver ønsket at vi fortsetter å bruke samme utviklingsverktøy. Programmeringsspråket brukt i applikasjonen var hovedsakelig Java inkludert *API* (bibliotek) for MySQL, PostgreSQL, JDBC og Asterisk. Protokollene for å hente kalender- og kontaktinformasjon var bestemt av oppdragsgiver på forhånd av prosjektet.

For å kjøre test-miljø stilte oppdragsgiver med virtuell pc, som kjørte linux(debian) og standard installasjon av *Asterisk* server, samt MySQL og PostgreSQL.

Oppdragsgiver stilte også med en virtuell pc som kjørte windows exchange server 2007, Service Pack 1. Formålet med denne server pc-en var å lage korrekte spørringer for LDAP og CalDAV, uten å arbeide opp mot HiG's egne databaser.

1.9 Øvrige roller

1.9.1 Oppdragsgiver

Jon Langseth, Avdelingsingeniør ved IT-avdelingen HiG.
Telefon: 611 35 156
E-post: jon.langseth@hig.no

1.9.2 Veileder

Tom Røise, Høgskolelektor ved Høgskolen i Gjøvik
Telefon: 611 35 192
E-post: tom.roise@hig.no

1.10 Organisering av rapporten

1.10.1 Terminologi

Liste med utdypelse av ikke-trivielle ord og uttrykk finnes i vedlegg A. Ordene i listen er skrevet i *kursiv* første gang de nevnes i rapporten.

1.10.2 Inndeling av rapporten

Kapittel 1 – Introduksjon
Her finner du formål og mål med prosjektoppgaven, problemområde, oppgavedefinisjon, avgrensning, målgruppe, bakgrunn og kompetanse samt rammer og øvrige roller.

Kapittel 2 – Kravspesifikasjon
Her finner du informasjon om utgangspunktet vi hadde, om de forskjellige kravene og beskrivelse av systemet.

Kapittel 3 – Arkitektur
Her finner du informasjon om den overordna arkitekturen og filstrukturen til applikasjonen.

Kapittel 4 – Design
Her finner du informasjon om databaser og protokoller som blir brukt. Kommunikasjonen mellom applikasjonen og databasen, valg av API og en beskrivelse av GUI til applikasjonen.

Kapittel 5 – Koding og Implementasjon
Her finner du informasjon om de utviklingsverktøyene vi brukte, hva som ble gjort med LDAP og CalDAV

Kapittel 6 – Testing
Her finner du informasjon om hva som ble brukt til testmiljøet og om testingen av LDAP

Kapittel 7 – Avslutning
Her finner du evaluering av oppgaven, våre problemer og utfordringer, videre arbeid, evaluering av gruppens arbeid samt en kort konklusjon.

Kapittel 8 – Litteraturliste

Her vil du finne liste av alle kilder som ble brukt til dette prosjektet.

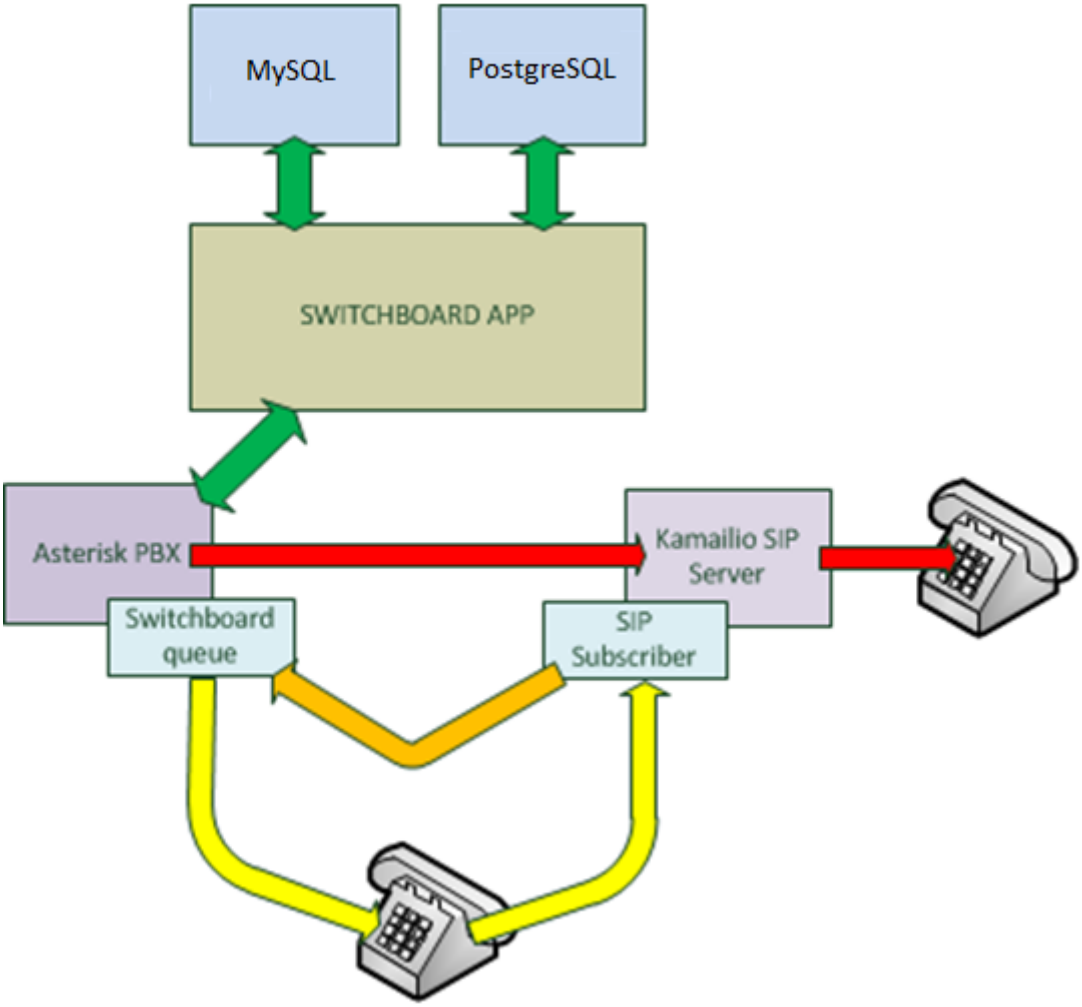
Kapittel 9 – Vedlegg

Her kommer vedlegg som er aktuelle for rapporten

Kapittel 2 Kravspesifikasjon

2.1 Utgangspunkt

Den eksisterende programvaren består av flere deler som utfører ulike oppgaver i systemet, som illustrert på figuren under.



Figur 1: Overblikk over systemet. [1]

Alle inn- og utgående samtaler blir håndtert av *Kamailio*. Dette er en åpen SIP server som kan håndtere tusenvis av samtaler per sekund. Når inngående samtaler er håndtert av Kamailio, blir de videresendt til Asterisk. Asterisk programvaren gjør om en vanlig pc til en kommunikasjonsserver. Her kan administrator definere samtale-køer for å holde styr på samtalene som mottas fra Kamailio.

I selve applikasjonen (‘Switchboard app’ på figuren over) kan det gjøres spørringer mot databasene for å finne kontakt og/eller kalenderinformasjon om personene som er lagret der. Denne informasjonen kan Asterisk bruke for å viderekoble samtaler som er i en kø, til de aktuelle personene fra databasen, samt brukes av operatøren for å sjekke om en person er tilgjengelig eller ikke.

Alle endringer som gjøres iløpet av prosjektperioden vil skje på «switchboard app» delen av systemet, da de andre delene skal være kompatible som de er.

2.2 Brukerbeskrivelser

2.2.1 Systemets omgivelser

Systemet vil bestå av flere separate deler.

Databasen for kontakt- og kalenderinformasjon administreres av IT-avdelingen ved HiG. Dette er en eksisterende database som allerede er i bruk ved HiG og vil bli vedlikeholdt av IT-avdelingen. De vil ha egne interne rutiner for oppdatering og endringer av databaseinholdet(dette med tanke på nye ansatte, oppsigelser m.m.). Applikasjonen vår vil kjøre spørringer opp mot databasen for å hente ut dataene den trenger.

Selve switchboard-applikasjonen skal være installert på pc-en som sentralbord-operatøren ved HiG arbeider på.

2.2.2 Systemets brukere

Applikasjonen utvikles hovedsakelig for å brukes av sentralbord-operatører ansatt ved HiG. Disse operatørene har varierende bakgrunnskunnskaper. Derfor må applikasjonen være enkel å forstå og være lett å bruke for både nye, og for mer erfarne brukere.

Administrator for systemet vil være ansatte ved HiGs IT-avdeling. Deres hovedansvar er å ajourføre databasene, konfigurere systemet og innstallere systemet på operatørenes PC-er.

Open source brukere
Siden oppdragsgiver ønsker å la programvaren være open source finnes det mange potensielle brukere, også med varierende bakgrunnskunnskaper. Disse brukerne må selv administrere databasene og konfigurere programvaren. For å bruke applikasjonen stilles det krav til open source brukerne at de selv har nok kunnskaper til å gjennomføre disse oppgavene.

2.3 Systemkrav

Systemkrav for alle maskiner som skal kjøre switchboard-applikasjonen:

- JRE (Java Runtime Environment) må være installert.
- Internet tilkobling for tilgang til databaseserverne.

I og med at applikasjonen i seg selv er så enkel, vil det kreves minimalt av operatørens PC for å kjøre programmet.

2.4 Eksterne krav

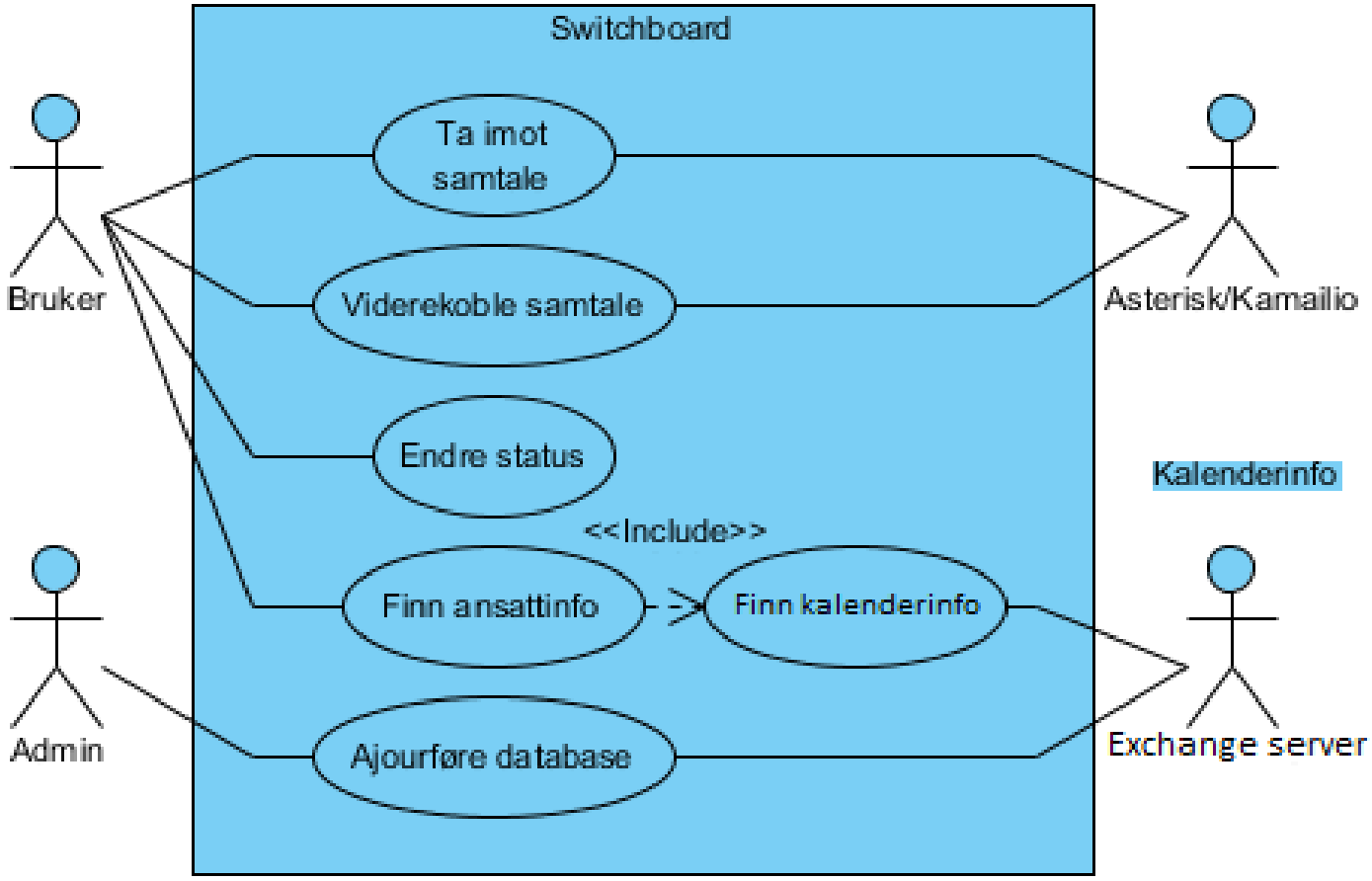
Eksterne krav for bruk av switchboard-applikasjonen:

- Server(e) for kontakt- og kalender-informasjon (LDAP og CalDAV).
- Asterisk server.

2.5 Systemets funksjonelle egenskaper

2.5.1 Use Case

Diagram



Figur 2: Use Case Diagram

Høy-nivå use case

Use Case:	**Ta imot samtale**
Aktør:	Bruker
Mål:	Omdirigere samtalen til riktig person
Beskrivelse:	Bruker tar imot en innkommende samtale. Deretter kan brukeren benytte seg av applikasjonens søkefunksjon (beskrevet i senere use case) for å finne informasjon om personen som ønskes å nås. Herfra kan brukeren enten viderekoble samtalen til gitt person (Beskrevet i senere use case) eller bryte forbindelsen.

Use Case:	**Viderekoble samtale**
Aktør:	Bruker
Mål:	Omdirigere innkommende samtaler
Beskrivelse:	Brukeren har mulighet til å omdirigere en aktiv samtale. Målet for omdirigeringen kan enten spesifiseres ved å taste nummeret, eller ved å benytte seg av søkefunksjonen, og velge aktuell person fra listen den gir.

Use Case:	**Endre Status**
Aktør:	Bruker
Mål:	Informere om brukerens status
Beskrivelse:	Gir brukeren muligheten til å sette sin status til «open», «lunch», «closed» og «custom». Endringer i brukerens status loggføres.

Use Case:	**Finn ansattinfo**
Aktør:	Bruker
Mål:	Finne informasjon om en gitt person i databasen.
Beskrivelse:	Brukeren kan søke på personer i databasen ved å oppgi riktig kriterier. Personene kan søkes opp via f.eks. navn, e-post kontor- eller mobilnummer. Etter søk blir aktuelle treff listet opp med sin informasjon, og herfra kan brukeren initsiere funksjonen for å hente kalenderinformasjon.

Use Case:	**Finn kalenderinfo**
Aktør:	Bruker
Mål:	Finne kalenderinformasjonen for en gitt person i databasen.
Beskrivelse:	Brukeren velger en gitt person som vises i applikasjonens GUI etter et personsøk har blitt gjennomført. Heretter blir personens kalender vist i applikasjonen

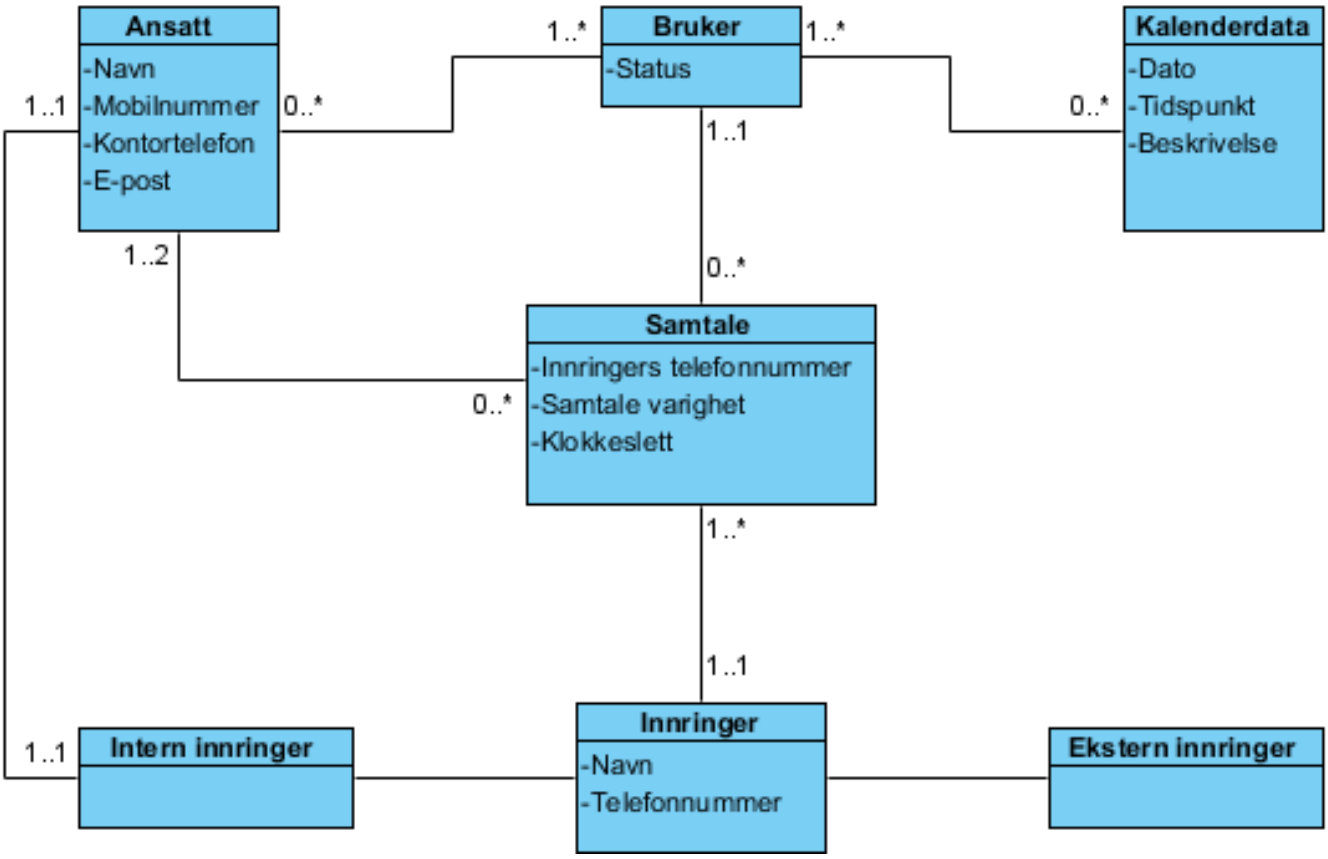
Use Case:	**Ajourføre database**
Aktør:	Administrator
Mål:	Holde databasen oppdatert
Beskrivelse:	Administrator kan slette, redigere og legge til data i databasen for å holde den oppdatert.

Detaljert use case

Use Case:	Viderekoble samtale
Aktør:	Bruker
Mål:	Omdirigere innkommende samtaler
Beskrivelse:	Brukeren har mulighet til å omdirigere en aktiv samtale. Målet for omdirigeringen kan enten spesifiseres ved å taste nummeret, eller ved å benytte seg av søkefunksjonen, og velge aktuell person fra listen den gir.
Betingelser:	1. En operatør må være tilgjengelig 2. Må være tilkoblet et nettverk som har tilgang til databasen
Detaljert hendelsesforløp:	1. Operatøren tar imot en samtale 2. Får informasjon om hvor kunden ønskes å bli viderekoblet 3. Skriver inn navn på den ansatte i søkefeltet 4. Velger hvilket nummer kunden skal viderekobles til 5. Trykker på viderekoble-knappen
Feilsituasjoner:	3. Ansatt ikke tilgjengelig i databasen 4. Ansatt ikke tilgjengelig på numrene i databasen
Alternativer:	3. Må si i fra til administrator, som legger inn ansatt i databasen 4. Skrive inn et nummer direkte i applikasjonen

Use Case:	**Ajourføre database**
Aktør:	Administrator
Mål:	Holde databasen oppdatert
Beskrivelse:	Administrator kan slette, redigere og legge til data i databasen for å holde den oppdatert.
Betingelser:	1. Må være administrator til stedet 2. Må være tilkoblet et nettverk som har tilgang til databasen
Detaljert hendelsesforløp:	1. Koble seg opp mot Exchange serveren 2. Logge inn på Exchange serveren 3. Gå inn i AD Ldap program 4. Velger gren man skal legge til, gren «Ansatte» 5. Trykke på «Add User» 6. Legge til navn, e-post og telefonnumre 7. Trykker «OK» 8. Høyreklikker på ny bruker. 9. Trykker på egenskaper 10. Legger til resterende ønsket data 11. Trykke «OK»
Feilsituasjoner:	4. Velger feil gren og lager ny bruker 6. Legger til feil navn, e-post eller telefonnumre 10. Legger til feil resterende data
Alternativer:	4. Sletter brukeren som ble opprettet og velger rett tre 6. Oppretter bruker på nytt 10. Går på egenskaper på nytt og setter inn rett data

2.5.2 Konseptuelt klassediagram



Figur 3: Konseptuelt Klassediagram

Kommentarer

I det konseptuelle klassediagrammet kan man se klassene som systemet vil bestå av, deres attributter og hvordan de henger sammen.

Klassen «samtale» representerer en samtale som kommer inn på systemet, fra enten en intern eller ekstern innringer. Dens 1 til 2 relasjon med klassen «Ansatte» viser at minst 1 ansatt må delta i en samtale. Dersom samtalen har en intern innringer vil 2 ansatte delta i samtalen.

«Ansatte» klassen representerer den delen av databasen hvor informasjon om ansatte lagres. Klassen «Kalenderdata» representerer databasen hvor data om de ansattes kalendere lagres. Relasjonene mellom klassen «Bruker», og «Ansatte» / «Kalenderdata» er for å illustrere at en bruker i systemet har tilgang til dataene lagret i disse databasene, gjennom applikasjonens GUI.

2.6 Systembeskrivelse

Applikasjonen bygger på et eksisterende, fungerende program som er i bruk ved UNINETT. Applikasjonens hovedfunksjon er å rute samtaler til riktig personer innad i en bedrift, gjennom en SIP server ved bruk av asterisk manager.

Denne prosessen foregår ved at det kommer en samtale inn i systemet, som blir besvart av en tilgjengelig switchboard operatør. Operatøren vil da ha muligheten til å viderekoble samtalen til riktig person ved hjelp av applikasjonen. Dette gjøres ved at operatøren taster inn et søkekriterie, f.eks. personens navn, e-post eller telefonnummer. Applikasjonen gjør da en spørring til databasen, og viser kontaktinformasjonen for treffene i dens GUI. Operatøren vil da få muligheten til å markere en av treffene som vises. Når en person er markert gjør applikasjonen en ny spørring til databasen. Denne spørringen henter kalenderinformasjon for den markerte personen, og viser disse dataene i applikasjonens GUI. Operatøren vil da ha oversikt over om personen sitter i møte, er hjemmeværende eller om den har noe planlagt. Dersom personen er ledig kan operatøren velge å overføre samtalen enten til personens mobiltelefon, kontortelefon eller ett spesifisert telefonnummer.

Operatørene vil også ha tilgang til å se aktive samtaler. Dette kan være nyttig dersom en innringer ønsker å overføres til en person som allerede er i en telefonsamtale, slik at operatøren kan videreføre denne informasjonen.

Applikasjonen lagrer også informasjon om x-antall nylige samtaler. Dette antallet blir spesifisert i filen switchboard.config.

2.6.1 Data spesifikasjon

Data input

Den eneste input'en som applikasjonen trenger er et søkekriterie for å søke opp personer i databasen. Her kan brukeren søke i alle attributtene lagret hos en bruker, som f.eks. personens navn, e-post, kontor- eller mobilnummer, ansatt ID eller hvilken etasje vedkommende hører til.

Data output

Den eksisterende applikasjonen gir output til skjermen i form av:

- Kontaktinformasjon
  - Navn, kategori/tittel, kontortelefon, mobiltelefon, e-post, avdeling og etasje.
- Kalenderinformasjon
  - Tid, type, status, sted, oppsummering, beskrivelse, besøkende og deltakere.
- Samtaleinformasjon
  - Navn, telefonnummer, lengde og tid.

Disse output dataene kan gjennomgå endringer etter oppdragsgivers ønske.

Kapittel 3 Arkitektur

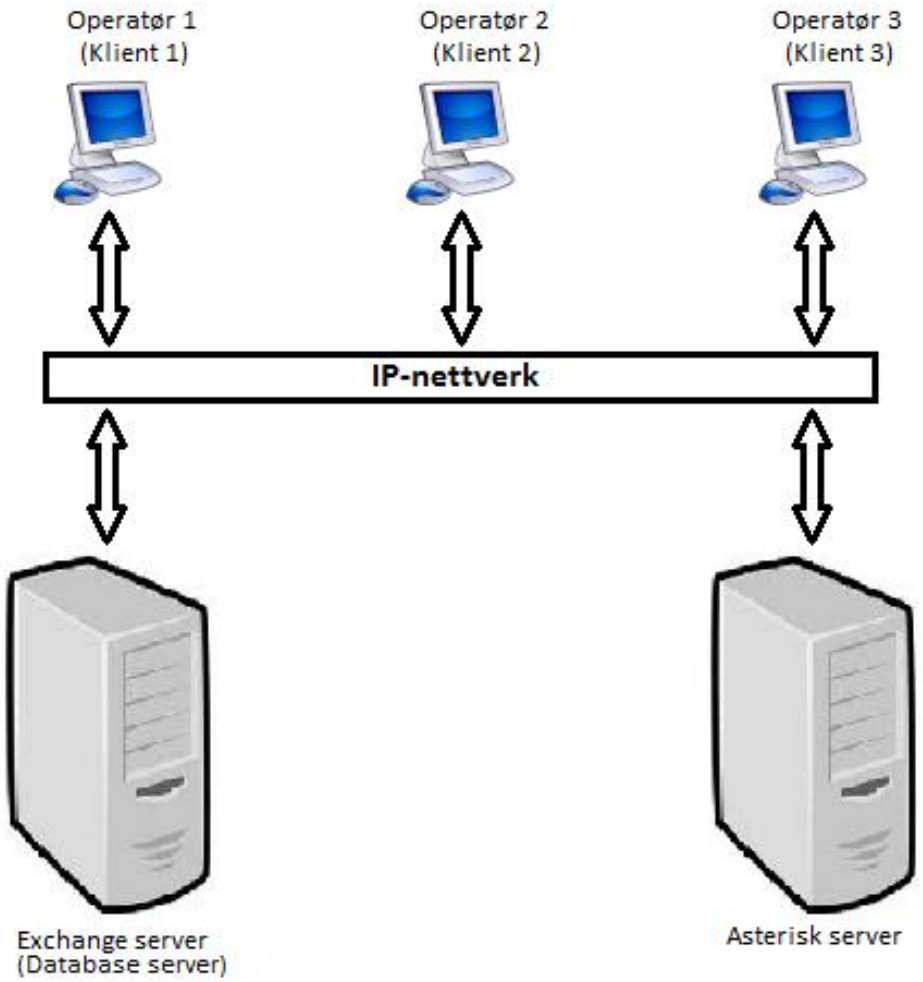
3.1 Overordnet arkitektur

Switchboard-applikasjonen fra UNINETT er oppbygd etter klient-tjener modellen. Det finnes to vanlige modeller for hvordan en klient-tjener løsning kan lages. Man kan enten lage en løsning med tynn klient og tykk tjener, eller en løsning med tykk klient og tynn tjener.

Ved tynn klient og tykk tjener menes at mesteparten av applikasjonen ligger på server-siden. Ved å lage en løsning etter denne modellen vil det ikke stilles høye maskinkrav til klientene, men heller større krav til serveren (tjeneren). En stor fordel med å utvikle applikasjoner etter denne modellen kommer frem når det er behov for oppdateringer. Det er mye enklere å oppdatere en server, enn å oppdatere alle klientene separat.

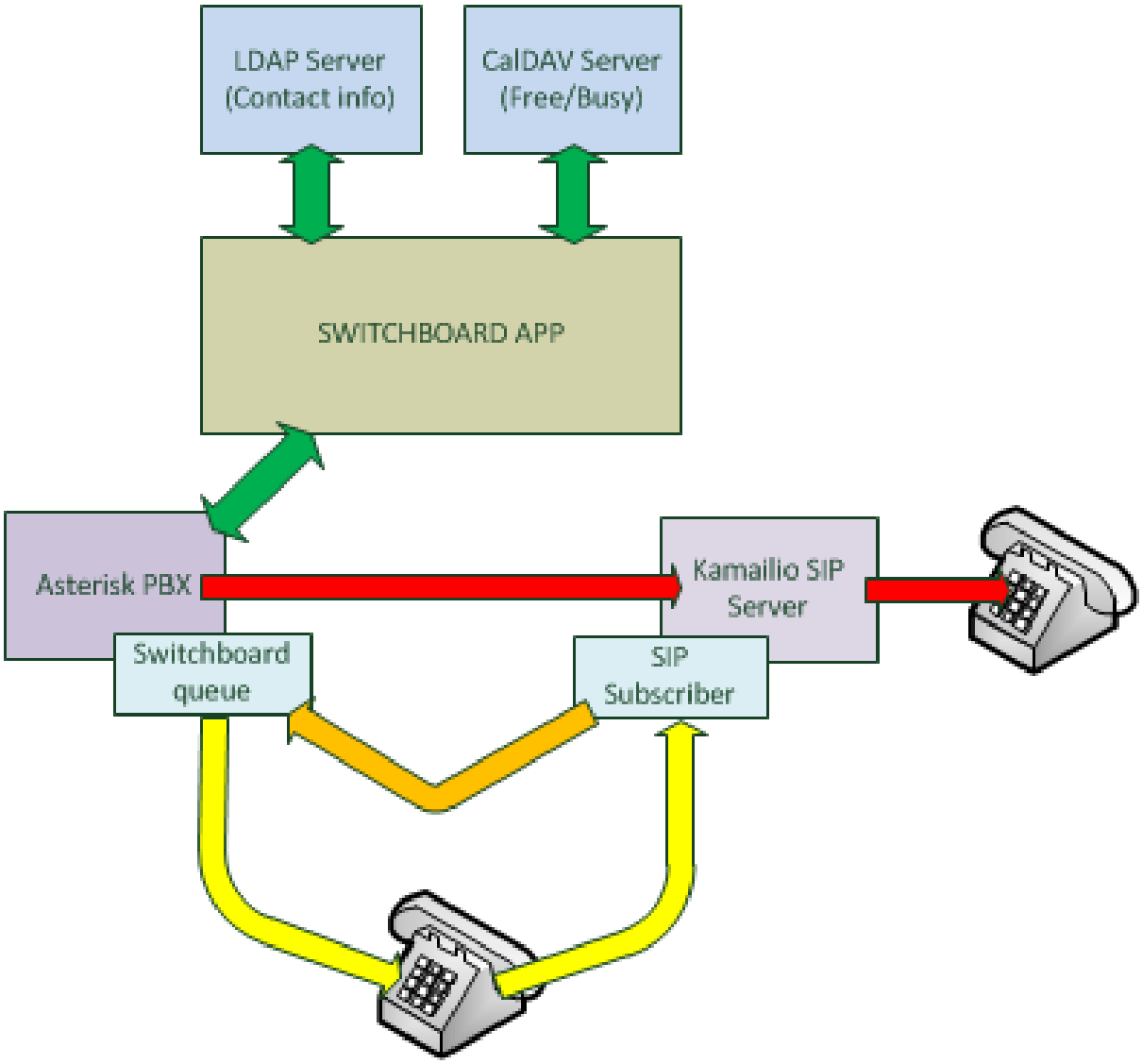
Applikasjoner bygget med tykk klient og tynn server stiller høyere krav til brukerpc-ene, og lavere krav til server pc-en. Det er fordelaktig å bruke denne modellen dersom applikasjonen krever mye prosessering. Ved å bruke tykke klienter kan man øke antallet brukere en server kan støtte, fordi arbeidet gjøres hos klienten.

UNINETTs sentralbord-løsning er bygd med tykke klienter og tynne tjenere som kommuniserer over et ip-nettverk. All prosessering foregår på klient-siden, mens serverene i systemet kun svarer på requests fra klientene. Se figur på neste side.



Figur 4: Illustrasjon av programvarearkitektur

De endringene vi hadde tenkt å gjøre var å endre MySQL og PostgreSQL til LDAP og CalDAV. Figuren under viser hvordan både vi og oppdragsgiver ønsket det. Den viser et overblikk av tilkoblingen av hele systemet sånn det skulle blitt.



Figur 5: Overblikk over det nye systemet. [1]

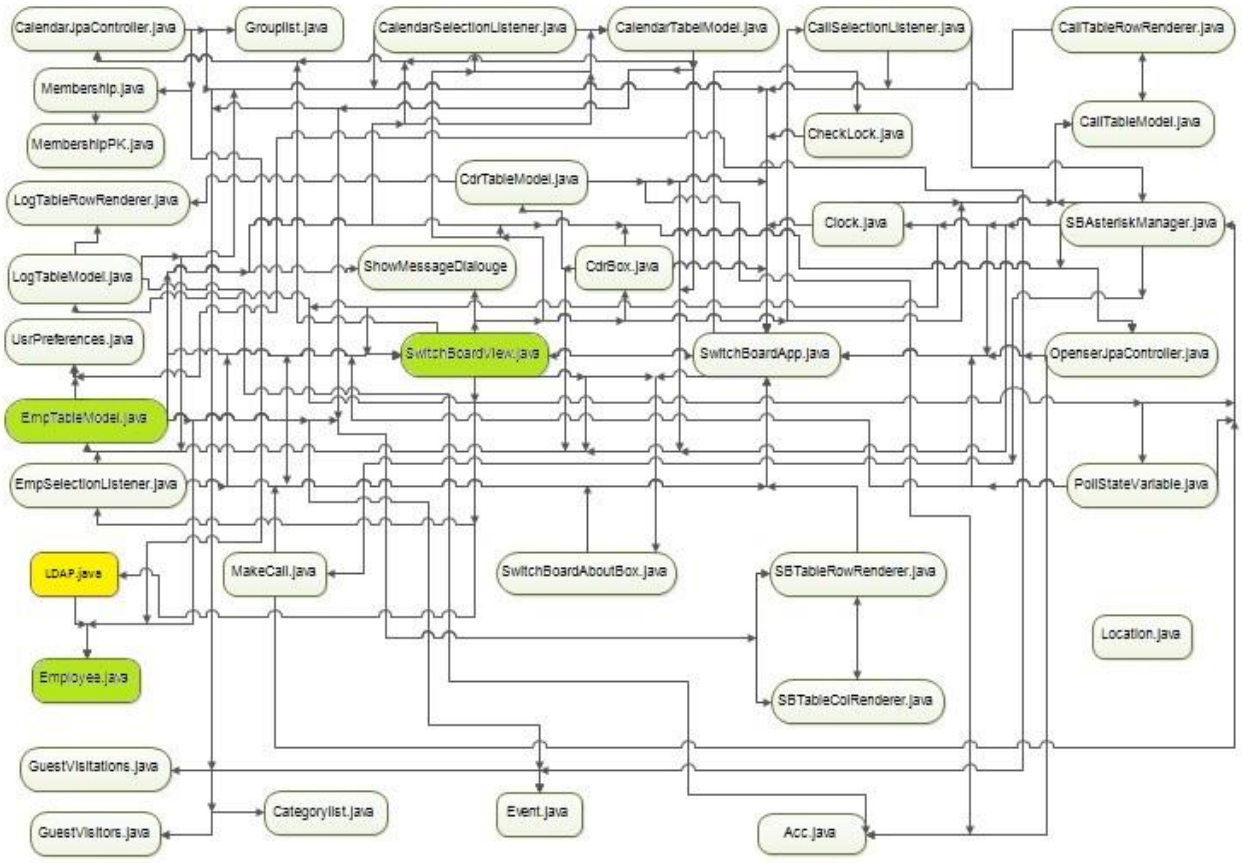
.

3.2 Filstrukturen av Javafilene

Dette er kun strukturen til javafilene.

I Switchboard-applikasjonen er filene veldig tett knyttet hverandre og aktivt brukt igjennom hele applikasjonen. Den mest sentrale filen er «SwitchBoardView.java». Det er hovedsaklig denne filen all informasjonen går igjennom. Enten om den blir tilkalt til «SwitchBoardView.java», eller sendt videre via «SwitchBoardView.java». Andre sentrale filer er «EmpTableModel.java» og CalendarTabelModel.java». Dette er filene som ordner tabellene for de ansatte og deres kalendere.

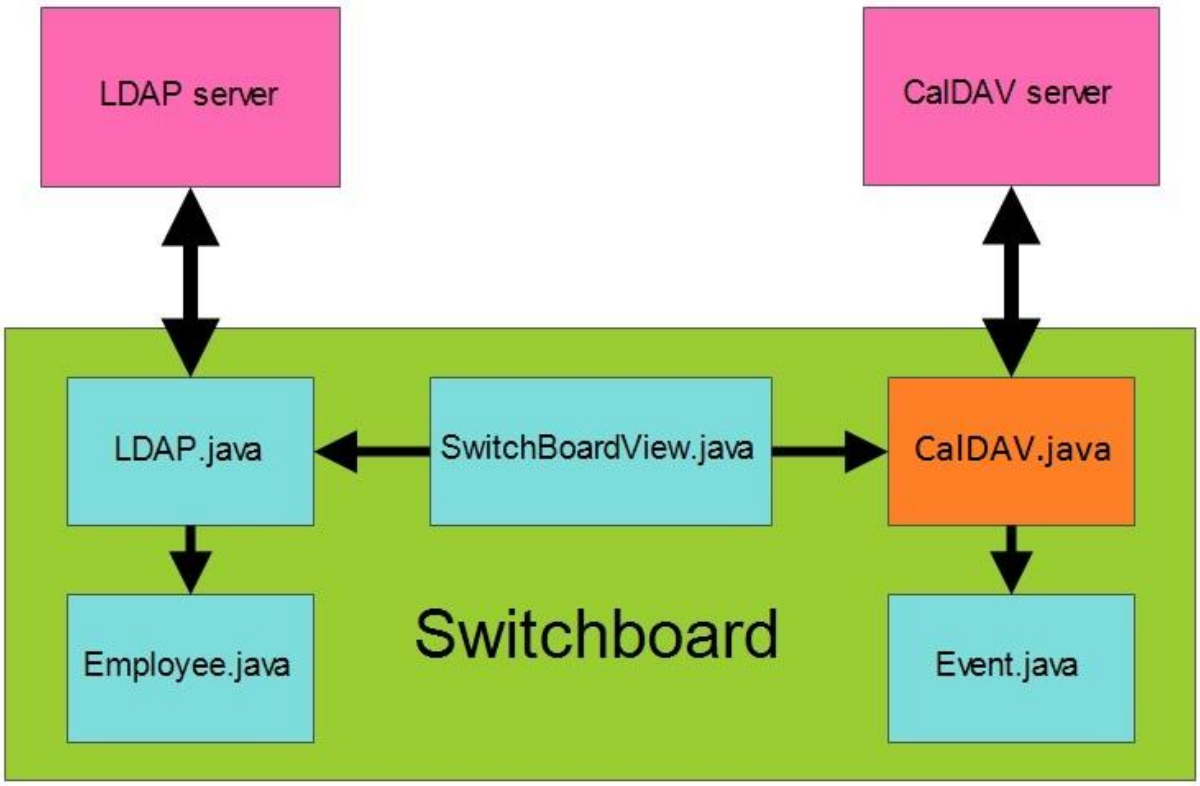
Det er en fil, «Location.java», som ikke blir brukt av noen av de andre java-filene. «Location.java» er en av mange filer som blir koblet opp mot en XML fil (Se vedlegg J).



Figur 6: Visualisering av filstruktur.

Figuren på forrige side gir en oversikt over hvilke filer som bruker de forskjellige filene i Switchboard-applikasjonen. Dit pilene peker, den filen blir brukt av den filen hvor pilen kom fra. Skriftlig liste til figuren er i vedlegg J. Gul representerer filene vi har laget. Grønn representerer filene vi har modifisert. Hvis repesenterer filene vi ikke har rørt.

I Switchboard-applikasjonens arkitektur, er det gjort sånn at hver server blir kontaktet i egne moduler. LDAP-modulen er uavhengig av CalDAV-modulen, men CalDAV-modulen er avhengig av LDAP-modulen. Figuren nedenfor viser ikke visuelt at CalDAV er avhengig av LDAP. Grunnen for dette er at kalenderinformasjonen blir hentet utifra e-posten til den spesifikke ansatte via applikasjonen. Switchboard-applikasjonen henter inn kontaktinformasjon om de ansatte, legger de i objekter og i en liste, som derretter blir sendt videre i switchboard-applikasjonen. Kalenderinformasjonen blir hentet inn i en annen modul, som så blir koblet opp mot en e-post til den kontaktpersonen operatøren trykker på i gui'n.



Figur 7: Tilknytning mellom LDAP/CalDAV modulene og hovedapplikasjonen.

Figuren figuren på forrige side viser visuelt at de to serverne har hver sin modul som er tilknyttet via «SwitchBoardView.java» . Dette er bare et lite utdrag fra Switchboard-applikasjonen. Resten av filene ligger i Switchboard, men var irelenvant å ta med i denne visualiseringen av LDAP og CalDAV modulene. Oransje representerer filen som ikke er med i programmet, men sånn det er tenkt at den skal være tilkoblet Switchboard-applikasjonen og CalDAV-serveren.

Kapittel 4 Design

4.1 Database

Databaser blir brukt for å lagre informasjon. Dataene som lagres følger ofte en formell og standarisert struktur. I dagens samfunn brukes databaser i nesten alle bedrifter. For eksempel på et bibliotek. Biblioteket bruker en egen database for å holde styr på hvilke bøker som er tilgjengelige og hvilke som er lånt ut, samt en egen database for registrerte kunder.

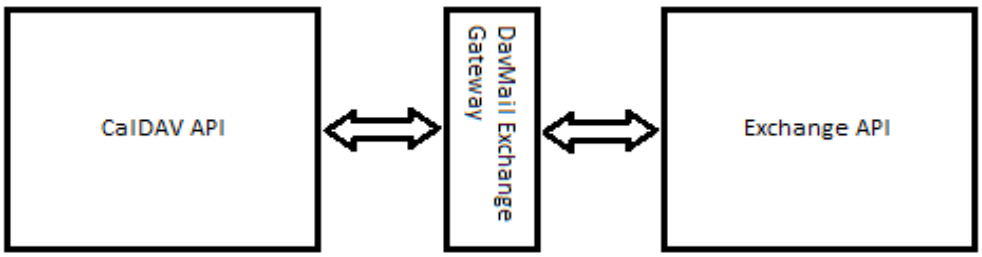
4.1.1 Server

Som server for kalender- og kontakt-informasjon bruker HiG en Microsoft exchange server 2007. Dette er en mail server med integrert støtte for kalenderdata og kontaktinformasjon. I vår applikasjon skulle vi primært bruke funksjonaliteten for kalender- og kontaktinformasjon.

Microsoft's exchange server støtter ikke protokollen vi ønsker å bruke for kalenderinformasjon. Derfor må vi benytte oss av ett 'mellomlegg'. Stian Husemoen ved IT-tjenesten anbefalte oss å bruke DavMail Exchange Gateway. Etter å ha lett etter andre alternativer uten å finne noen en annen exchange gateway med støtte for CalDAV, bestemte vi å benytte oss av DavMail Exchange Gateway.

DavMail Exchange Gateway

DavMail Exchange Gateway er en programvare som gir støtte for flere standard protokoller mot exchange servere. Programvaren innkluderer blant annet en LDAP 'gateway' for å utveksle kontaktinformasjon og full kalenderstøtte med free/busy display.



Figur 8: Illustrasjon av kommunikasjon via DavMail.

DavMail brukes som en gateway mellom CalDAV API'et og exchange serverens API, og gjør det mulig å gjøre CalDAV spørringer mot en exchange server. DavMail er veldig godt dokumentert på deres hjemmesider, med blant annet dokumentasjon for å sette opp forskjellige mail klienter mot exchange server og hvordan konfigurere DavMail til å kjøre i server modus.

For bruk til å utvikle vår applikasjon installerte vi DavMail på server pc-en, og konfigurerte det for å kjøre i server modus. [2]

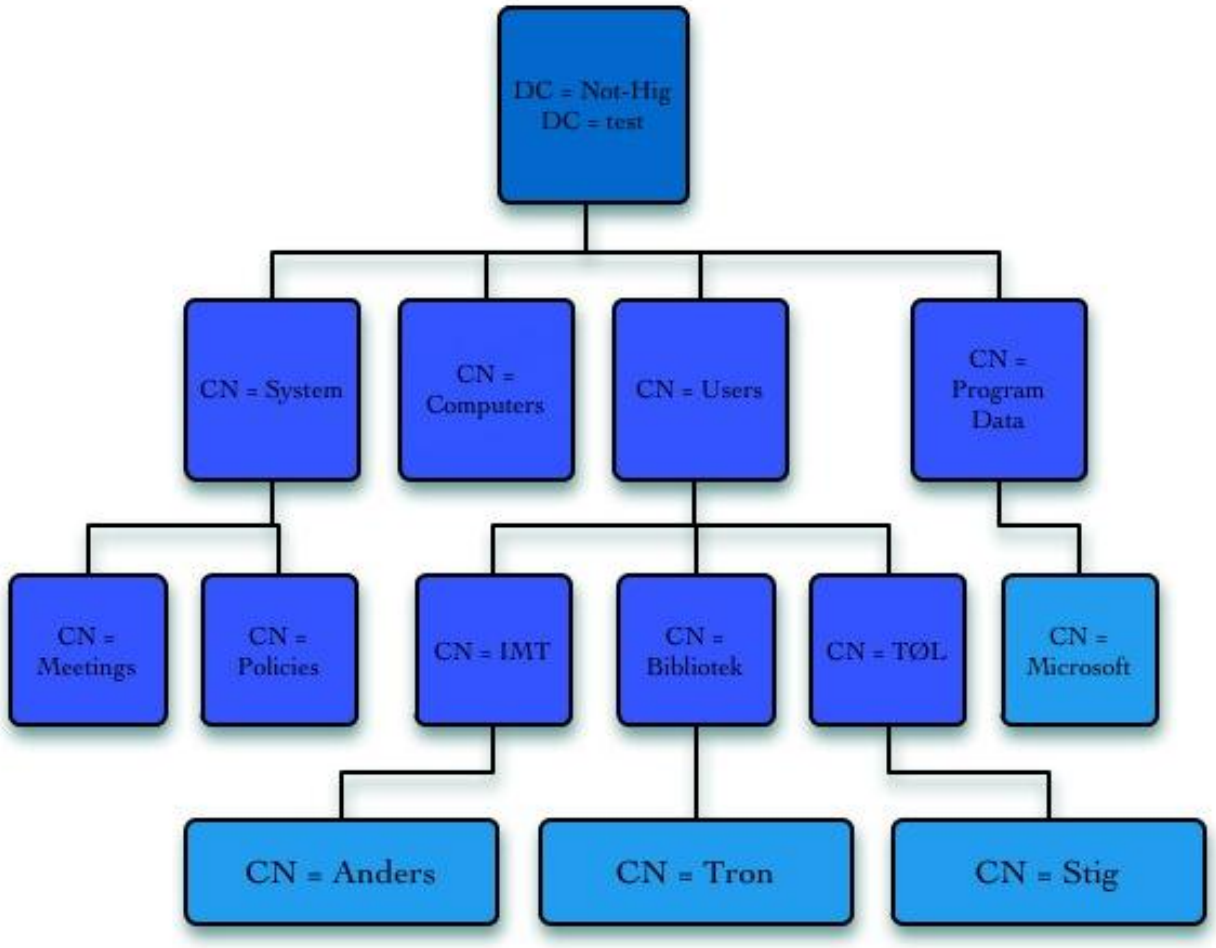
4.1.2 Protokoller

For å kommunisere med serveren har vi brukt protokollene LDAP og CalDAV.

LDAP

LDAP er en protokoll som brukes for oppslag i en katalogtjeneste på en server over et IP-nettverk. Den kan brukes både til å hente ut data via spørringer, og til å gjøre endringer på en server. Vi vil bruke denne protokollen til å hente ut informasjon om de ansatte som ligger på serveren når sentralbordoperatøren ønsker det.

Protokollen er bygd opp i en trestruktur, hvor man har en «path» man må følge for å finne den informasjonen man er ute etter. LDAP-søkingen fungerer på den måten at man må søke fra ett bestemt sted i treet, og deretter kan man velge om man vil kun søke i nivået under ditt søkepunkt, eller søke i alle utspring fra søkepunktet som ble satt. Se figur nedenfor.



Figur 9: Eksempel på LDAP's trestruktur

Med dette i bakhodet, og at alle ansatte her på Høgskolen er plassert under en «gren» i treet, klarte vi å redusere søketiden for uthentingen av informasjon. Vi satte søket vårt utifra den grenen i Høgskolens database som kun inneholdt de ansatte, så søket unngikk de grenene hvor det ikke fantes ansatte i. Se figur 4 ovenfor.

Fra oppdragsgiver fikk vi en retningslinje / ønske om hvordan strukturen skulle være. Og det var at det skulle være enkelt å endre databasetype. Som for eksempel fra ldap til mysql. Grunnen for dette, var at programmet kan bli lagt ut for Open Source om UNINETT og IT-tjenesten blir enige om det. Sånn statusen er nå, har de ikke hatt diskusjonen om det enda, men diskusjonen er irrelevant for vår oppgave. Vår oppgave angående dette emnet er å gjøre det kompatibelt for det. IT-tjenesten kunne ønske seg å ha det som Open Source, så det står på skuldrene til UNINETT om det blir lagt ut for Open Source eller ikke. Uansett utfallet, vil det bli lettere for IT-tjeneste og andre bedrifter å endre databasen til sitt eget bruk med strukturen vi planla.
Det hadde vært mye mer arbeid hvis ikke applikasjonen allerede hadde meste av den oppbygningen den måtte ha for at det skulle være mulig på så kort tid. Hvis det ikke hadde vært funksjonalitet for dette, måtte endringer på hele programmet blitt gjort.

Sånn vi valgte å løse problemet, var å lage en egen modul til LDAP i applikasjonen. Modulen vår skulle kun hente ut informasjonen som var relevant til sluttbruker. Informasjonen skulle hentes ut fra databasen å legge all informasjonen om alle ansatt-objektene i en liste. Deretter skulle vi sende listen med en funksjon i Switchboard-applikasjonen som håndterte søking på navn og utskriving til skjerm. Grunnen til at vi la objektene i en liste, var at vår tolkning av kildekoden tilsa at funksjonen skulle motta en liste. Dette viste seg å være både rett og feil, noe som vi kommer nærmere innpå i kapittel 6.2, LDAP Testing.

CalDAV

CalDAV er en utvidelse av WebDAV, en http baserte protokoll for data manipulasjon. CalDAV kan brukes for å manipulere, eller hente data på en ekstern server. Protokollen bruker standardformatet iCalendar. iCalendar formatet ble designet for å være protokoll uavhengig, noe som gjør at formatet kan sendes over flere medier. Videre følger et eksempel på et iCalendar objekt.

```
BEGIN:VCALENDAR                // Begynner et calendar objekt
PRODID:-//Example Corp.//CalDAV Client//EN
VERSION:2.0
BEGIN:VEVENT                   // starten på en event(VEVENT)
UID:1@example.com              // Bruker ID som denne event tilhører
SUMMARY:One-off Meeting        // En liten beksrivelse om eventen
DTSTAMP:20041210T183904Z
DTSTART:20041207T120000Z       // Dato og tid når eventen starter og
DTEND:20041207T130000Z         // slutter
END:VEVENT                     // Avslutter gjeldene event
END:VCALENDAR
```

I vår applikasjon vil vi bruke denne protokollen for å hente ut kalenderinformasjon fra serveren når sentralbordoperatøren velger en gitt person.

4.1.3 Kommunikasjon

All kommunikasjon mellom applikasjonen og databaseserveren blir initiert av brukeren.

Applikasjon/bruker – databaseserver (kontaktinformasjon)
Serveren lytter hele tiden etter requests fra klientene. Når en bruker søker opp en person i applikasjonens grensesnitt sendes det et request til serveren om å hente alle informasjonen om alle personer i databasen. Serveren sender sine treff tilbake applikasjonen, som går igjennom resultatene en etter en, og sammenligner hver persons data med søkekriteriet fra brukeren. Personer som matcher søkekriteriet blir lagt i en liste, og tilslutt fremstilt for brukeren i applikasjonens grensesnitt.

Applikasjon/bruker – databaseserver (kalenderinformasjon)
Etter brukeren har søkt opp personer i databasen kan han/hun også finne kalenderinformasjonen til en gitt person. Brukeren velger gitt person ut fra listen i applikasjonens grensesnitt, og det gjøres en ny spørring mot databaseserveren. Applikasjonen henter navnet til personen brukeren har valgt, og dagens dato. Disse attributtene brukes i spørringen, og serveren returnerer alle kalenderoppføringer som samsvarer med kriteriene.

4.2 Valg av API

4.2.1 LDAP

I valget av LDAP API legger vi hovedsaklig vekt på at API'et ikke er utdatert, at det er utviklet av pålitelig kilder og at det ikke er GPL lisensiert. Etter å ha valgt et API som følger våre kriterier vil vi gjennomføre tester av API'et før vi tar et endelig valg.

Apache Directory LDAP API:
Apache Directory LDAP API er kontinuerlig under arbeid for å skape et forbedret LDAP API for java, som erstatning for JNDI og andre LDAP API'er (jLdap og Mozilla LDAP API). [3]

Fordeler:

- Kontinuerlig arbeid tilsier at API'et ikke er utdatert.
- Apache 2.0 lisens tillater gratis open source bruk.
- Utviklet av apache.
- Anbefalt av tidligere brukere.
- God dokumentasjon.

Ulemper:

- Kilder forteller om problemer med tidligere versjoner av dette API'et. (versjon 1.0.0-M3) [4]

jLDAP:
jLDAP ble utviklet for å tilby et kraftig, men fortsatt enkelt API for LDAP tjenester I java. [5]

Fordeler:

- OpenLDAP Public License.
- Utviklet av Novell.
- Anbefalt av tidligere brukere.

Ulemper:

- Repository som inneholder API'et ble sist endret 6. juli 2009

UnboundID LDAP SDK:

UnboundID LDAP SDK for java er et raskt, kraftig og bruker vennlig java API for kommunikasjon med en LDAP server. [6]

Fordeler:

- Dokumentert som det raskeste java LDAP API'et i 2011. [4]
- God dokumentasjon.
- Gratis å bruke og redistribuere som open source.
- Anbefalt av tidligere brukere.

Ulemper:

- Nyeste versjon ble sluppet tidlig i 2012.

Konklusjon:

Vi valgte å benytte oss av Apache Directory LDAP API. Hovedgrunnen til denne konklusjonen er at både jLDAP og UnboundID LDAP SDK har gått lenge uten noen oppdateringer, og kan være utdatert, mens Apache Directory LDAP API'et derimot er et pågående arbeid, som sist ble oppdatert 3. mars 2013.

Apache Directory LDAP API'et tilfredsstiller også kravet fra oppdragsgiver angående lisensiering, og det finnes mye god dokumentasjon på apache sine nettsider, som f.eks javadocs og en omfattende brukerveiledning. Dette API'et er også varmt anbefalt av tidligere brukere.

Etter å ha gjort noen enkle funksjonstester med API'et er vi godt fornøyd med vårt valg. Vi hadde ingen problemer med å koble opp mot exchange serveren vi programmerer opp mot, og fant de riktige dataene uten store problemer.

4.2.2 CalDAV

I valget av API for CalDAV har vi hovedsakelig sett etter ett API som støtter all funksjonalitet vi trenger, at det er utviklet av pålitelige kilder, at det ikke er GPL lisensiert og at det ikke er utdatert. Etter å ha funnet et API som passer våre kriterier vil vi gjennomføre enkle funksjonstester, før vi tar et endelig valg.

CalDAV4j:
CalDAV4j er et java skapt for å implementere CalDav protokollen i java, hvor de største bidragsyterne er BABEL og OSA (Open Source Application) Foundation. Biblioteket skal gjøre det lett å sende komplekse spørringer til en CalDav server. [7]

Fordeler:

- Apache 2.0 lisens.
- Støtte for «free-busy».
- Seriøse bidragsytere.
- Anbefalt av tidligere brukere.

Ulemper:

- Sist oppdatert i 2012.

iCal4j:
iCal4j er et utvidelses-bibliotek for java som tilbyr støtte for å koble til «back-end» kalender og vCard servere, inkludert CalDav og CardDav servere. [8]

Fordeler:

- Anbefalt av tidligere brukere.

Ulemper:

- Lite lett tilgjengelig informasjon.
- Lite lett tilgjengelig dokumentasjon.
- Fant ingen dokumentasjon angående støtte av «free-busy».
- Sist oppdatert sent i 2012.

Milton:
Milton API støtter kalender- og kontakt-informasjon via Webdav, CalDav, Carddav og LDAP, og er kompatibelt for alle større operativsystemer. [9]

Fordeler:

- Siste oppdatering i mars 2013.
- Ble i 2012 integrert i Google Contacts, som benyttes i Mac og iPhone/iPad.
- God dokumentasjon

Ulemper:

- Kun basis funksjonalitet tilgjengelig for open source bruk

Konklusjon:
Vi valgte å benytte oss av CalDAV4j. iCal4j luket vi tidlig ut, da vi hadde problemer med å finne god dokumentasjon av funksjonalitet og beskrivelser.

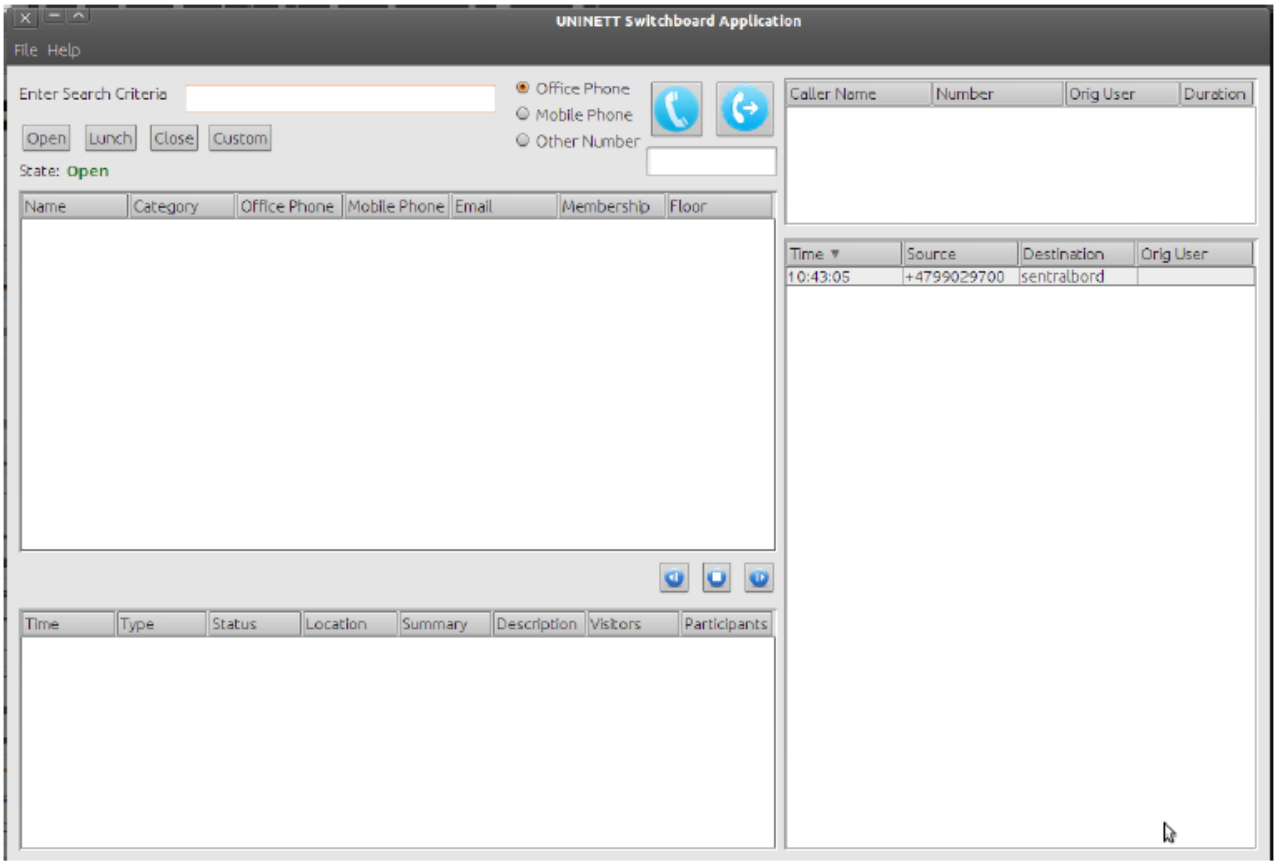
Milton derimot virket som en lovende kandidat, siden API'et inkluderer støtte for både CalDAV og LDAP, og var godt dokumentert på deres nettsider. Det eneste ulempen med dette API'et er at man kun får tilgang til basis funksjonalitet, uten å betale for lisens. Denne basis funksjonaliteten inneholder ikke støtte for CalDAV.

CalDAV4j kommer varmt anbefalt av tidligere brukere, samtidig som det innfrir oppdragsgivers krav for lisensiering. API'et er relativt dårlig dokumentert på deres hjemmesider, men man kan få tilgang til kildekoden ved å bruke mavens utvidelse til Netbeans IDE og hente kildekoden fra mavens repositories. Det finnes ingen gode guider, eller kode eksempler tilgjengelig.

Den dårlige dokumentasjonen gjorde det vanskelig å gjennomføre tester av API'et. Etter en del prøving og feiling, greide vi å koble opp mot serveren, og forsatte å arbeide med CalDAV4j.

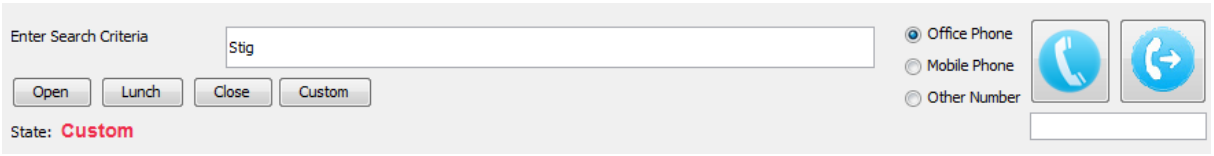
4.3 Beskrivelse av applikasjonens GUI

Applikasjonens GUI består av et hovedvindu, som er inndelt i 4 seksjoner:



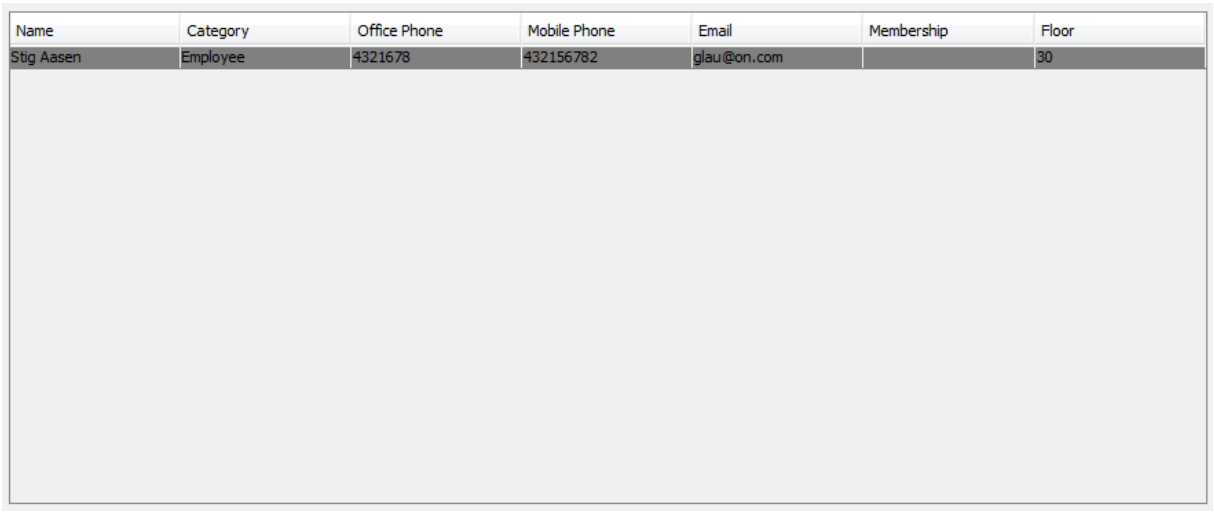
Figur 10: Bilde av applikasjonens GUI

1. Den første seksjonen inneholder et søkefelt (søkrekriterier er beskrevet i kap. 2.6.1, 'data input'), muligheten for å endre status for operatør og muligheten for å viderekoble samtaler til et egendefinert nummer eller et nummer som er lagret i databasen. Viderekoblingen kan skje enten ved å velge et telefonnummer registert på en person som finnes i databasen, eller ved å taste det inn manuelt.



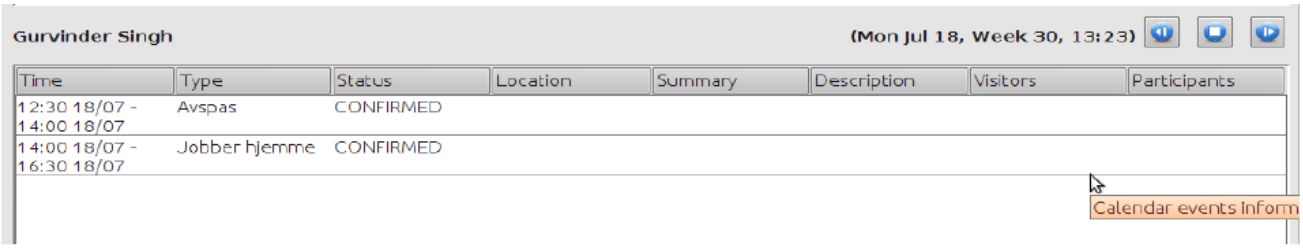
Figur 11: Eksempelbilde av seksjon 1

2. Denne seksjonen benyttes for å vise frem informasjonen som finnes i databasen om personene som søkes opp av brukeren. Dataene som blir vist i dette vinduet er beskrevet under punktet ‘kontaktinformasjon’, i 2.6.1 ‘Data output’.



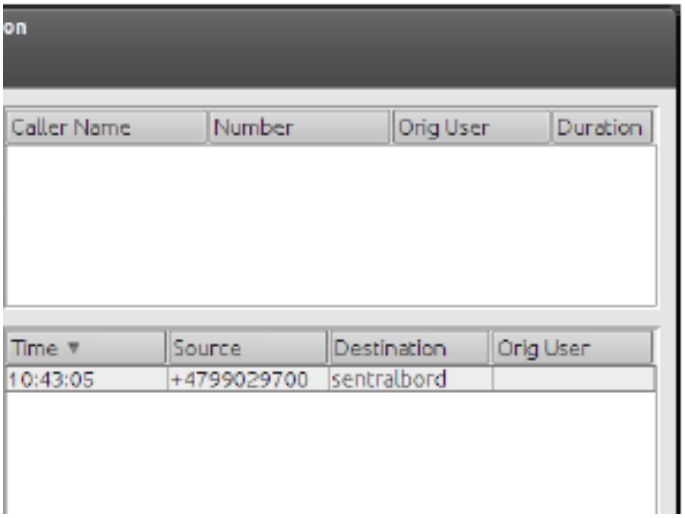
Figur 12: Eksempelbilde av seksjon 2

3. Dette vinduet brukes til å vise kalender-informasjon for en person valgt i seksjon 2. Det vises automatisk dagens avtaler, men brukeren har også muligheten til å navigere mellom flere datoer via GUI’et. Dataene som blir vist i dette vinduet er beskrevet under punktet ‘Kalenderinformasjon’, i 2.6.1 ‘Data output’.



Figur 13: Eksempelbilde av seksjon 3

4. Denne seksjonen inneholder informasjon om pågående og nylige samtaler. Dataene som blir vist i dette vinduet er beskerevet under punktet ‘Samtaleinformasjon’, i 2.6.1 ‘Data output’.



Figur 14: Eksempelbilde av seksjon 4

Kapittel 5 Koding og Implementasjon

5.1 Utviklingsverktøy

Oppdragsgiver ønsket at vi skulle fortsette å bruke samme miljø og språk som UNINETT allerede hadde benyttet i sin eksisterende applikasjon. UNINETT har benyttet seg av Java i sin applikasjon, med Netbeans som arbeidsmiljø.

5.1.1 Java

Java er en av de mest brukte men også et av de nyere programmeringsspråkene. Java har noen sterke sider som har gjort at den har trukket til seg programmerere, det at den kan kjøres på alle plattformer er noe å ta i betraktning. For å nevne noen til så er Java for lett å lære seg, dette sammenlignet med andre programmeringsspråk, det er lett å kompilere, debugge og skrive koden og det er innebygd automatisk minne håndtering. Det er et åpent programmering språk noe som gjør at dem slipper å tenke på store lisens utgifter årlig.

5.1.2 Netbeans

Netbeans er et enkelt og ryddig IDE (Integrated Development Environment) med støtte for flere programmeringsspråk som Java, HTML5, PHP og C++. Netbeans inneholder masse funksjonalitet for å hjelpe utviklere. Funksonaliteten innebærer for eksempel kode tips og sammenligning, og markering av ord.

5.1.3 TortoiseGIT

For å få hente ut kildekoden fra UNINETT's repository, brukte vi TortoiseGIT.

TortoiseGIT er en GIT versjonskontroll klient for windows. En GIT versjonskontroll klient er programvare som brukes i forbindelse med programvareutvikling. Programvaren skal hjelpe til med å holde styr på endringer som gjøres av flere brukere i et prosjekt, slik at de kan jobbe individuelt, men alikevel unngå at data blir overskrevet.

5.1.4 Wireshark

Wireshark er programvare for å analysere pakker som sendes over et nettverk. En slik programvare forsøker å fange opp nettverkspakker og vise pakkedataene så detaljert som mulig. I dag er Wireshark en av de beste open source pakke analysatorene på markedet.

5.2 LDAP

5.2.1 Prossessen rundt ldap koden

Først måtte vi forstå den opprinnelige koden. Og ettersom det var null kommentarer i selve koden, ble det brukt mye tid til debugging av applikasjonen. Det som ble sett etter under debugging var struktur på liste og objekter. Hvordan programmet hang sammen, samt hvordan informasjon ble sendt rundt i programmet.

Da strukturen og koden var undersøkt og forstått, begynte planleggingen av vår egen kode. Det var nå vi forstod at vi kunne lage en modul som ordnet opp i LDAP. Vår modul skulle bestå av å kontakte databasen, søke igjennom og hente ut informasjonen sluttbruker er ute etter. Hvordan vår modul ble bygd opp, tok vi steg for steg. Først måtte vi ordne at modulen fikk kontakt med databasen. Det var flere alternativer på akkurat dette punktet, men noen av dem var i et annet API enn det vi hadde valgt.

I det man tar kontakt med serveren, sier man også hvordan man vil motta informasjonen. Og hvilken måte man ønsker å motta informasjonen på, kommer an på hvilken port du tar kontakt opp mot. Man kan velge mellom om man vil ha informasjonen tilsendt kryptert eller i klartekst. Hvis man ønsker klartekst, velger man port 389. Og hvis man ønsker kryptert, velger man port 636. Velger man kryptert, må man sende med en tredje input da man tar kontakt, som er «true».

Alt av innhold på LDAP-databasen er helt vanlig kontaktinformasjon. Så vi anså ikke dette som sensitiv invformasjon, og med det valgte vi å hente ut informasjonen i klartekst.

5.2.2 Om ldapkoden

Først måtte vi få kontakt med databasen fra modulen. LDAP er litt spesielt på akkurat å ta kontakt. Man må først ta kontakt, også spørre om man komme inn. Spørre om lov gjør man med «Bind».

```
LdapConnection connection = new LdapNetworkConnection( "128.39.142.36", 389 );
connection.bind( "CN=Anders Tester,CN=Users,DC=not-hig,DC=test", "passord" );
connection.setTimeOut(0);
```

Dette er kode som er hentet fra LDAP Apache sin brukermanual! [10, 11] Første linjen på koden over tar kontakt med serveren, hvor det oppgis IP i første input og port i andre input. På neste linje prøver modulen å autentisere brukeren med brukernavn og passord som input. Siste linjen sier at kontakten med databasen ikke skal bryte automatisk etter 30 sekunder. Vi valgte å sette inn «connection.setTimeOut(0)» i tilfelle noe uventet skulle skje med Switchboard-applikasjonen i oppstartsfasen som gjorde at oppstarten tok mer enn 30 sekunder, vil fortsatt Switchboard-applikasjonen kunne hente ut informasjon fra Ldap.

Sånn det ble programmert av oss, er konfigurasjonesinformasjon som IP, port, brukernavn og passord hardkodet i Switchboard-applikasjonen. Dette skulle være en midertidig løsning som var planlagt å endre senere da resten var på plass. Det viktigste for oss var å få på plass funksjonaliteten rundt LDAP. I kapittel 7.x tar vi for oss hvordan vi hadde tenkt å endre konfigurasjonsinformasjon fra hardkodet til å legge det i en konfigurasjonsfil.

Etter koden som tok kontakt med databasen funksjonabelt, var søket det neste som skulle ordnes. Søket bestod av en «path», en spørring, hvordan søket skulle gå og hvilke attributter som skulle hentes ut. Alt som hadde med kontakt opp mot databasen og uthenting av informasjon skjedde med API-kall.

Søkingen var tatt som utgangspunkt ut fra brukermanual til apache på deres hjemmesider, med litt modifikasjoner fra vår side. Under ser du koden vår på hvordan søket i databasen er skrevet.

```
SearchRequest searchRequest = new SearchRequestImpl();
searchRequest.setBase(new Dn("CN=Users, DC=not-hig, DC=test"));
searchRequest.setFilter("(&(objectclass=organizationalPerson)(givenName=*))");
searchRequest.setScope(SearchScope.SUBTREE);
String [] attributes ={"mail", "telephoneNumber", "otherTelephone", "sn", "givenName", distinguishedName"};
searchRequest.addAttributes(attributes);
```

Dette er kode som er hentet fra LDAP Apache sin brukermanual [12], som vi har tilpasset vårt bruk. Det som foregår i koden over er at det blir opprettet et søk. Vi har satt søkebasen til grenen «CN=Users» under roten «DC=not-hig, DC=test». Filteret er spørringen som skal filtrere ut de objektene som ikke skal hentes. Det vår spørring sier er alle objekter med «*objectclass = organizationalPerson*» og har noe innhold i «*givenName*». Grunnen for akkurat disse to, er på grunn av at alle ansatte er av objektklassen «*organizationalPerson*». Men det også andre objekter som er av denne objektklassen. Derfor har vi tatt med «*givenName*», for alle objekter som ikke er ansatte har ikke noe innhold i denne attributten.

Det er to forskjellig søk i ldap, det er «SUBTREE» og «ONELEVEL». «ONELEVEL» sier at søket skal kun gjøres på nivået under søkebasen, mens «SUBTREE» gjør søket på alle nivåene under søkebasen. Vi valgte «SUBTREE» på grunn av at det var det beste alternativet utifra hvordan databasen til Høgskolen er bygget opp. Attributtene er den informasjonen i objektene vi var ute etter fra databasen. Attributtene vi var ute etter var e-post, telefonnumer og navn. Vi tok også med en unik tekst som kan bli brukt som brukernavn i videre utvikling.

Manglene som stod igjen av kode for at vår modul var å overføre innholdet av objektene i databasen over til en liste med objekter i programmet. Med et LDAP API-kall kunne vi gå fra objekt til objekt. Mens for å se igjennom og traversere innholdet i hvert objekt, måtte vi bruke «iterator». Dette er en javafunksjon som deler opp objektet så vi kunne hente ut informasjonen og bruke det. For å kunne gjøre dette, måtte vi ha to while-løkker, en for objektene og en for innholdet i objektene. While-løkkene var så enkle og elegante som at hvis det var en neste, så fortsatte løkken. Den indre løkken gikk igjennom innholdet i objektet. Attributten ble lagt i en midertidig string. Stringen ble sjekket opp en annen string, og hvis den fant en string som samsvarte, ble den lagt inn i sin respektive datakapsel. Hvis det var en neste attributt i objektet, fortsatte while-løkken. Akkurat i denne if-løkka blir det i tillegg til å legge attributt i sin datakapsel, også lagt til en id til objektet.

```
if ( "givenN".equals(temp.substring(0, 6)))    {
  firstname = temp.substring(11, temp.length() -1);
        id++;
  }
```

Den ytre while-løkken gikk igjennom objektene. Det som foregikk i den ytre while-løkken, var at man opprettet et objekt med sine respektive data og ble lagt i en liste.

```
Employee arbeider = new Employee(username,firstname,lastname,mail,cellphone,otherTelephone,id);
  emplist.add(arbeider);
```

All koden som ble laget av oss ligger i filen «ldap.java», men det ble gjort noen endringer på noen andre filer. Det kommer vi tilbake til i neste delkapittel.

5.2.3 Integreringen av applikasjonen og ldapmodulen

Integreringen av LDAP modulen og den eksisterende applikasjonen gikk ikke uten noen endringer i applikasjonen. Vi lagde klassen på objektene på samme måte som den allerede var på i applikasjonen, bare med andre stringnavn. Så da vi integrerte de to systemene måtte vi endre på de private stringene så de samsvarte med hverandre. Dette gjaldt også i funksjonene som tilkalte de forskjellige stringene. Endringene av navnene ville ikke det ha noen effekt på funksjonene som allerede var opprettet i klassen, «Employee.java». Vi la ikke til noen andre funksjoner, men vi fjernet funksjoner som aldri ble brukt i den eksisterende applikasjonen. Det var rett og slett bare overflødig kode.

En annen fil vi måtte gjøre endringer på, var «EmpTableModel.java». Spesifikk funksjonene «public List searchData(string searchText)». Denne funksjonen går igjennom alle ansatte og igjennom alle dens data og sjekker om de inneholder søketeksten som sluttbruker har skrevet i søkefeltet sitt. I denne funksjonen er det en for-løkke som hadde feil interval. Vi endret intervalet til å stemme med det antallet på størrelsen av array'en med data den skulle sjekke, som var en større enn opprinnelig;

Opprinnelig:
```
for (int i = 0; i < values.length - 1; i++)
```
Nå:
```
for (int i = 0; i < values.length; i++)
```

Den andre funksjonen som ble endret på var «public void fillSearchData(List empList, List grplList, List memList)». Denne funksjonen fikk opprinnelig tre lister inn. Hvor to av listene ikke ble brukt til noe av det vi kunne se, så vi fjernet det og anså det som overflødig kode. «SwitchBoardView.java» var den siste filen som det ble gjort endringer på. Denne filen tilkalte «fillSearchData(List empList, List grpList, List memList)» funksjonen, og ettersom vi hadde gjort endringer på hva den tok i mot, måtte vi også endre tilkallingen av funksjonen. Nedenfor ser du endringen;

Opprinnelig:
```
empModel.fillSearchData(calDB.findEmployeeEntities(),
        calDB.findGroupList("PUBLIC"), calDB.findMembershipEntities());
```
Nå:
```
empModel.fillSearchData(emp.emplist);
```

Resten av funksjonen brukte den siste listen, empList, som var listen av de ansatte. Den går igjennom alle ansatte og legger de i en midertidig array, en array per ansatt med sine respektive data, som blir sendt videre i applikasjonen. Vi fjernet en del unødvendig kode som vi ikke så behoved for som omhandlet de ansatte. Index'ene ble endret på, og vi reduserte størrelsen på array'en ned til 5;

Opprinnelig: `String[] temp_emp = new String[9];`

Nå: `String[] temp_emp = new String[5];`

5.3 CalDAV

Denne modulen ble ikke ferdigstilt innen innleveringsfristen, og er derfor ikke inkludert i vår løsning. Videre i dette delkapittelet følger en detaljert beskrivelse av vårt arbeid med CalDAV.

5.3.1 Prosessen rundt CalDAV

Før vi kunne gå i gang med utviklingen av CalDAV modulen, måtte vi først sette oss inn i hvordan den eksisterende applikasjonen fungerte og identifisere hvilke filer det var nødvendig å endre for implementere den nye modulen. Vi lærte mye om applikasjonen gjennom debugging. Her fulgte vi gangen i programmet, og fant raskt frem til hvilke filer og funksjoner som samhandlet for å hente, og vise kalenderinformasjonen.

Vi kom frem til at de følgende filene og funksjonene var involvert for å hente kalenderinformasjonen:

- Switchboardview.java
    - toDayActionPerformed()
        - Kalles når brukeren velger en person i listen over personer som matcher brukerens søkekriterie.
        - Kaller på funksjonen showTodayEvents() i filen EmpTableModel.java.
    - nextDayActionPerformed()
        - Kalles når brukeren trykker knappen i GUI for å vise neste dag av kalenderen for valgt person.
        - Kaller på funksjonen showNextDayEvents() i filen EmpTableModel.java.
    - prevDayActionPerformed()
        - Kalles når brukeren trykker knappen i GUI for å vise forrige dag av kalenderen for valgt person.
        - Kaller på funksjonen showPrevDayEvents() i filen EmpTableModel.java.

- EmpTableModel.java
    - showTodayEvents(), showNextDayEvents(), showPrevDayEvents()
        - Henter dagens/morgendagens/gårsdagens dato og kaller på funksjonen findEvents() i filen CalendarJpaController.java. Når findEvents() kalles sendes med aktuell persons navn, og start- og sluttdato for intervallet som det hentes kalenderinformasjon fra. I dette tilfellet er start- og sluttdato den samme. Til slutt returnerer funksjonen en liste med arrangementer.
        - Kaller på funksjonen showCalEnteries() i filen CalendarTabelModel.java. Når denne funksjonen kalles sendes det med listen over arrangementer som hentes av findEvents(), og aktuell persons ansattnummer.
- CalendarJpaController.java
    - findEvents()
        - Setter opp en forhåndsdefinert spørring for å hente arrangementer og legger resultatene fra spørringene i en liste.
- CalendarTabelModel.java
    - showCalEnteries()
        - Viser frem resultatene i listen som sendes med når funksjonen kalles i applikasjonens GUI.

Etter å ha identifisert de nødvendige filer og funksjoner begynte arbeidet med å planlegge modulen. Vi innså at kildekoden for kalenderinformasjon var strukturert likt som kontaktinformasjon, som vi allerede hadde hatt suksess med å gjenbruke delene av koden som tok seg av visning. Og vi valgte derfor å prøve samme løsning for kalenderinformasjon.

For å implementere CalDAV spørringen måtte vi byttet ut funksjonen findEvents() med en egen funksjon som hentet aktuell kalender fra exchange serveren. Deretter måtte funksjonen sortert de aktuelle arrangementene og returnert dataene som en liste. Listen måtte bli videresendt til funksjonen showCalEnteries(), og informasjonen vist for brukeren i applikasjonens GUI.

5.3.2 Arbeidet med CalDAV

For å bytte ut den gamle modulen for å hente kalenderinformasjon, valgte vi å definere en ny klasse kalt 'BaseCaldavClient'. Vi definerte den nye klassen som en utvidelse av klassen 'HttpClient'Klassen 'HttpClient' brukes av CalDAV4j API'et for å koble opp mot en server. BaseCaldavClient klassen inneholdt all data nødvendig for å koble til en server, autentisere en bruker og finne frem til rett kalender.

Før vi fortsatte å arbeide med BaseCaldavClient ønsket vi å verifisere at oppkoblingen mot serveren funket. CalDAV4j inneholdt en egen funksjon for nettopp dette, funksjonen testConnection() i klassen CalDAVCollection. Vi opprettet et object av klassen CalDAVCollection, og kalte på funksjonen testConnection() med et object av BaseCaldavClient som argument.

Her møtte vi på en rekke feilmeldinger. Mer detaljer om problemene og de mest sentrale feilmeldingene med CalDAV finnes i kapittel 7.2.2 'CalDAV'.

Kapittel 6 Testing

6.1 Testmiljø

For å kunne kjøre applikasjonen i dens opprinnelige form, var vi avhengige av å sette opp ett testmiljø. Dette test miljøet måtte inneholde en asterisk server, mysql server og postgresql server. Oppdragsgiver stilte med en virtuell pc som kjørte linux (debian) for å kjøre serverne til testmiljøet.

«Dummy serveren» beskrevet i delkapittel 5.1.4 ble satt opp for å teste og implementere LDAP og CalDAV spørringer.

6.1.1 Asterisk server

For å gjøre konfigurasjonen av asterisk serveren lettere installerte vi et grafisk web grensesnitt for «Asterisk Manager Interface» (AMI). Gjennom dette grensesnittet ble det mulig å definere brukere, sette opp køer og holde oversikt over hele asterisk systemet. Installasjonen av grensesnittet ble gjort etter en guide publisert på asteriskguru.com.

AMI må være aktivert for at applikasjonen skal fungere, siden den bruker AMI'et for å koble opp og kommunisere med asterisk.

Aktivering av AMI og GUI

- /usr/src/gui
    - Laster ned filer for det grafiske web grensesnittet.
    - Installerte grensesnittet med kommandoen 'made install'.
- /etc/asterisk/manager.d/mymanager.conf
    - Opprettet filen 'mymanager.conf' og definerte brukerne som skulle tillates tilgang til AMI.
- /etc/asterisk/manager.conf
    - Inkluderte følgende kodelinjer:
        - Enabled = yes
        - Webenabled = yes
        - #include manager.d/mymanager.conf
- /etc/asterisk/http.conf
    - Inkluderte følgende kodelinjer:
        - Enabled = yes
        - Enablestatic = yes
        - Bindaddr = 0.0.0.0
- /var/lib/asterisk/static-http og /var/lib/asterisk/gui_backups
    - Endret rettigheter på disse mappene slik at AMI'et fikk tilgang til å endre nødvendige filer. [13]

Konfigurasjon av asterisk server

For å konfigurere asterisk serveren korrekt kontaktet vi Gurvinder Singh via mail, og spurte om hva som trengtes for å sette opp testmiljøet. Gurvinder svarte på mailen dagen etter, og vi fikk beskjed om at vi måtte gjøre følgende:

- Opprette en kø i asterisk.
- Opprette 3 brukere i asterisk.
- Definere en bruker i asterisk som medlem av køen.

6.1.2 MySQL

Undersøkte litt angående fjernstyring av en MySQL database. Det gikk litt på sikkerhet og hvordan man ga tilgang til en server til en annen datamaskin. Tok en kort diskusjon med oppdragsgiver om hvordan han ville ha sikkerheten rundt det, og sikkerhetsmessig er det dårlig. Men ettersom dette produktet er under utvikling, hadde ikke sikkerheten rundt tilgang til MySQL databasen mye å si.
Måtte inn i en fil kalt «my.cnf» som er config fila til MySQL. Der gjorde vi endringer under headingen [mysqld] på IP, hvor vi endra IP fra *127.0.0.1* til *128.39.142.236*. Søkte etter en linje som jeg enten måtte fjerne eller kommentere bort for å kunne gi fjernstyrt adgang. Den kodelinja var allerede fjernet da vi først gikk inn i config fila. Kodelinja var «*skip-networking*».
Etter endringen i config fila, restarta vi serveren og da var alt satt opp for at vi kunne gi adgang til hvem vi ville til databasen. Kode for å restarte serveren er: /etc/init.d/mysql restart
Logget inn på mysql serveren med kode i denne rekkefølgen:

- mysql –u root –p mysql

CREATE DATABASE testswb;

GRANT ALL ON testwsb.* TO root@'%' IDENTIFIED BY 'MRr6mPSp'

% betyr alle ip'er på det nettverket MySQL serveren er på. [14, 15]

6.1.3 PostgreSQL

Under arbeidet med å få eksisterende kildekode kjørbart så trengte vi en Postgresql server å gjøre spørringene mot. Vi fikk tilgitt en server av IT-tjenesten der det var installerte en postgresql server. Vi hadde ikke fått vite datastrukturen på databasen, vi tok dermed kontakt med arbeidsgiver angående dette. Det endte opp med at arbeidsgiver tok kontakt med Gurvinder og fikk vite at tabellene og attributtene var beskrevet i koden. Siden dette var bare et midlertidig test miljø så hadde vi ikke behov for å sette primærnøkler og fremmednøkler i databasen. Det viktigste var å få riktig navn på attributtene og tabellene. Vi brukte pgAdmin3 til å konfigurer databasen. For å kunne teste programmet la vi inn litt data om fiktive personer i databasen noe som ga oss innblikk i eksisterende kode. Dette ga oss innblikk i hvordan spørringene henger sammen i applikasjonen, og gjorde det enklere å vite hvor vi skulle bytte til Ldap og CalDAV. [16]

6.1.4 Dummy server

Istedenfor å arbeide direkte på HiGs exchange server, satte oppdragsgiver opp en virtuell pc som kjørte windows server 2008. På denne serveren kunne vi trygt arbeide med å implementere LDAP og CalDAV uten å risikere HiGs databaser.

Oppsett av server
Da vi fikk oppgitt serverens informasjon fra oppdragsgiver,var kun windows server 2008 installert og remote desktop aktivert. Herfra logget vi inn, og kontrollerte pc-en fra vårt eget skrivebord via «Microsoft Terminal Services Client» (mstsc.exe).

Det første vi gjorde med serveren var å installere exchange server 2007 Service Pack 1 og sette opp vårt eget domene, som vi valgte å kalle not-hig.test. Deretter lagde vi en rekke med testbrukere med diverse forskjellige attributter som f.eks mobilnummer, slik at vi kunne prøve å hente ut data fra AD via LDAP.

For å kunne bruke CalDAV mot exchange serveren brukte vi DavMail Exchange Gateway for å oversette mellom CalDAV API-et og exchange API-et.
Vi lastet ned DavMail fra deres sider på sourceforge.net, og fulgte instruksjonene for installasjon som var godt dokumentert på nettsiden.

DavMail kan kjøres både på klient-siden, og på server-siden. For å teste og utvikle CalDAV modulen valgte vi å kjøre DavMail i server modus. Konfigurasjonen for å sette opp DavMail i server modus var også godt dokumentert på nettet.

For å konfigurere DavMail, måtte vi endre på de relevante verdiene i davmail.properties. Ved oppstart av programmet leses verdiene i davmail.properties filen, og brukes til å sette forskjellige variable i DavMail.

davmail.properties

Følgende endringer måtte gjøres i davmail.properties for å kjøre DavMail i server modus: [17]

davmail.url=http://«server-ip»/owa – URL'en til Outlook Web Access for exchange serveren.
davmail.enableEws=true – Aktiverer Exchange Web Services for exchange serveren.
davmail.caldavPort=1080 – Spesifiserer en port DavMail lytter etter CalDAV spørringer.
davmail.server=true – Spesifiserer at DavMail skal kjøre i server modus.
davmail.allowRemote=true – Tillater eksterne tilkoblinger mot DavMail.

6.2 LDAP Testing

Testingen vi gjorde er det vi kaller «*blackbox testing*». Det vil si at vi ser om programmet oppfører seg som det skal, sender informasjonen til rett sted til rett tid. [18]
Vi ville ha ferdig vårt interne system før vi i det hele tatt ville prøve å integrere det inn i selve applikasjonen. Det vi først fikk feil på, var at attributtene la seg på feil plass. Ettersom koden vår var laget sånn at for eks hvis «telefonnr» var den første attributten vi fikk tak i, la vi «telefonnr» først for innsetting. Denne løsningen fungerte helt fint fram til vi oppdaget et nytt problem. Og det problemet var at hvis det ikke var en attributt av den informasjonen vi var ute etter, så kræsjet programmet. Dette gjorde at vi måtte endre koden som omhandlet innlegging av objektene.

Med en del tenkning på en aktuell løsning til dette. Kom vi fram til at vi kunne legge string'n vi fikk inn i en midlertidig string som vi igjen sjekket opp mot enkelte ord for at den skulle bli lagt riktig. Og hvis den ikke hadde fått noen plass, ble den ikke brukt.

```
String lastname = "";
//kode i mellom som er irrelevant til eksempelet..
if( "sn:".equals(temp.substring(0,3)))  {
        lastname = temp.substring(4, temp.length() -1);
}
```

Initsierer string'n til å være tom i tilfelle den ikke har noe informasjon. Hvis den har noe data, legges det informasjonen inn.

Det som er med ldap databasen, er at alle attributter har sin egen måte å lagre innhold. Stringen inneholder ikke bare attributten, men den inneholder også navnet på attributten. Her kommer et eksempel:
La oss si vi har en e-post, som er «rudolf@not-hig.test», som vi skal ha ut. I en hvilken som helst database, vil den string'n inneholde «rudolf@not-hig.test». Men i LDAP vil den stringen inneholde «mail: rudolf@not-hig.test».

Så for å skille hvilke attributter som skal hvor. Sjekket vi det første ordet opp mot det ordet vi har satt i koden. Men for å unngå at vi sender med disse attributt-navnene videre, brukte vi den midlertidige string'n. Så kopierte vi stringen fra «temp» til den aktuelle lagringsplassen den skal ha. I tillegg til å luke ut attributt-navnene, måtte vi utelukke det siste tegnet i string'n. For det viste seg å være et linjeskift.
Vi brukte javafunksjonen «substring(X ,Y)» for å få dette til. «X» er starten på string'n, som har lengden til attributtnavnet pluss 2. Og «Y» er slutten, som er total-lengden på string'n minus 1.

```
lastname = temp.substring(4, temp.length() -1);
```

Nå var vårt interne system feilfritt. Det tok inn de attributtene vi ønsket, og de ble lagt på rett plass i hvert objekt som senere ble lagt inn i ei liste.

Integreringen av vårt interne system og selve applikasjonen stod igjen. Det viste seg å bli verre enn vi hadde trodd. Applikasjonen kræsjet hver gang vi kom til det punktet applikasjonen skulle bruke listen vi hadde sendt med, og vår tolkning av koden tilsa at den skulle ha inn en liste. Og vi visste at hvert objekt var satt opp på samme måte som fra den forrige databasen den fikk innhentet informasjon fra. Det ble mange timer med debugging, kodesjekking og research på nettet uten hell.

På vei til et ukentlig møte med vår veileder, smatt tanken inn om at kanskje applikasjonen ikke ville ha den tradisjonelle liste-typen, men en annen type av en liste. Så før herr Røise kom inn på kråkereiret, startet vi å debugge. Det viste seg at applikasjonen ville ha inn en liste av typen «Vector», så vi endret listetypen i vårt interne system så den lagde en «Vector»-liste i stedet for den tradisjonelle liste-typen.

Men funksjonen «fillSearchData(List empList)» som ikke var helt som den skulle, som gjorde at søkefunksjonen på navn ikke fungerte i det hele tatt. Dette var et problem vi hadde en aning om hvorfor skjedde. Vi løste problemet på den enkle måten at vi reduserte lengden på array'en og endret hvilke data som skulle hvor. Dette løste problemet med søking i selve applikasjonen.

Kapittel 7 Avslutning

7.1 Evaluering av oppgaven

Når vi ser tilbake på arbeidet med oppgaven, føler vi det er lite vi kunne gjort annerledes.

Våre begrensede erfaringer med større applikasjoner og bruk av uavhengige biblioteker viste seg å være en større faktor enn tidligere antatt, spesielt under arbeidet med å gjøre UNINETT's løsning kjørbar (jfr. kapittel 7.2.1 'Hovedapplikasjon'). For å løse problemene måtte vi bruke mange flere arbeidstimer enn planlagt, som igjen førte til forsinkelser i videre arbeid.

Oppsettet av testmiljøet ble gjennomført uten noen større problemer. Testmiljøet bestod av en asterisk server, mysql server og postgresql server, og var nødvendig for å sette seg inn i applikasjonens prosesser. For mer utdypende informasjon rundt oppsett av testmiljøet, se kapittelene 6.1.1 'Asterisk server', 6.1.2 'MySQL' og 6.1.3 'PostgreSQL.

Siden applikasjonen skulle skreddersys til HiG's bruk, antok vi at integrasjonen av LDAP og CalDAV skulle foregå opp mot deres databaser. Dette viste seg å være noe oppdragsgiver ikke ønsket, og vi endte derfor opp med å måtte sette opp en egen windows exchange server for å integrere de nye databaseløsningene. Arbeidet med å sette opp serveren gikk relativt smertefritt, men siden dette arbeidet var uforutsett, førte dette til ytterlige forsinkelser i videre arbeid. På dette tidspunktet i prosjektperioden lå vi relativt langt etter planlagt fremdrift, og vi økte antallet ukentlige arbeidstimer for å gjøre opp for tapt tid.

Etter at exchange serveren var på plass begynte vi straks på integrasjonen av LDAP. Arbeidet med LDAP gikk veldig bra, og vi fullførte modulen på kortere tid enn planlagt.

Vi begynte arbeidet med å integrere CalDAV noen dager etter exchange serveren var på plass. Her møtte vi på en rekke problemer. Dette gjorde at arbeidet med CalDAV ble for tidkrevende, og under møte med oppdragsgiver den 30. april, ble vi anbefalt å avslutte arbeidet med CalDAV og heller fokusere videre arbeid på hovedrapporten.

7.2 Problemer og utfordringer

7.2.1 Hovedapplikasjon

Vi fikk tilgang til UNINETT's switchboard-applikasjon gjennom deres git repository, og lastet den ned ved hjelp av TortoiseGIT. Når vi planla fremdriften i prosjektet hadde vi ikke kjennskap til at de nødvendige bibliotekene ikke var med i samme repository. Dette skapte problemer når vi senere begynte arbeidet med den eksisterende applikasjon, siden vi ikke hadde forutsett å måtte gjøre denne jobben selv.

Arbeidet startet med å debugge applikasjonen i Netbeans. Resultatet av dette var en rekke feilmeldinger, hvor de fleste oppstod på grunn av manglende biblioteker. Vi forsøkte å rette opp i problemene ved å finne og inkludere nødvendige biblioteker, men fortsatte å få diverse feilmeldinger. Etter å ha stått fast på dette problemet en stund, uten å finne en løsning, tok vi kontakt med oppdragsgiver. Jon sa seg villig til å hjelpe oss finne en løsning, og noen arbeidsdager senere var alle bibliotekene på plass.

Etter at de nødvendige bibliotekene var inkludert i applikasjonen, møtte vi på nye feilmeldinger. Disse feilmeldingere dukket opp fordi applikasjonen ikke hadde noen MySQL og PostgreSQL servere å koble opp mot. Detaljer rundt oppsettet av database serverene kan finnes i kapittel 6.1.2 'MySQL' og 6.1.3 '.

Selve tabellene i databasene måtte vi opprette ut i fra hvordan de var definert i kildekoden. Tabellene var greit forklart i java filene, som f.eks. 'Employee.java', som inneholdt tabellen for ansatte.

7.2.2 CalDAV

Det første problemet vi møtte på under arbeidet med CalDAV, viste seg allerede under valget av API. Ingen av open source API'ene var godt dokumentert, og de manglet guider/instruksjoner. Den dårlige dokumentasjonen gjorde funksjonstestingen av API'et vanskeligere, og mer tidkrevende enn tidligere antatt. Dette førte til at vi måtte bruke mer ressurser enn vi hadde planlagt på denne modulen.

Etter hvert som vi arbeidet med å koble til og hente ut informasjon fra serveren, støtte vi på flere forskjellige problemer. Men for hvert problem vi fikset, fulgte det en ny feilmelding.

Feilmelding: Connection timed out

Første gang vi forsøkte å kjøre funksjonen testConnection() for å teste oppkoblingen mellom serveren og applikasjonen, endte vi opp med følgende feilmelding:

[2013-04-26 16:55:38] INFO [main] HttpMethodDirector – I / O exception (java.net.ConnectException) caugh when processing request: Connection Timed out: Connect
[2013-04-26 16:55:38] INFO [main] HttpMethodDirector – Retrying request
[2013-04-26 16:55:59] INFO [main] HttpMethodDirector – I / O exception (java.net.ConnectException) caugh when processing request: Connection Timed out: Connect
[2013-04-26 16:55:59] INFO [main] HttpMethodDirector – Retrying request
[2013-04-26 16:56:20] INFO [main] HttpMethodDirector – I / O exception (java.net.ConnectException) caugh when processing request: Connection Timed out: Connect
[2013-04-26 16:56:20] INFO [main] HttpMethodDirector – Retrying request
Exception in thread "main" org.osaf.caldav4j.exceptions.CalDAV4JException: Connection timed out: connect

Applikasjonen forsøkte å kontakte serveren 3 ganger, men hver gang ble oppkoblingen tidsavbrutt uten å få noe svar. For å finne ut av hva som var galt, begynte vi å debugge applikasjonen. Alle attributter så korrekte ut, og applikasjonen kjørte helt fint frem til kodelinjen hvor oppkoblingen skulle skje. Her frøs applikasjonen i ca. 30 sekunder, før den kom med beskjed om at koblingen hadde blitt tidsavbrutt.

Etter dette mistenkte vi at feilen lå i DavMail konfigurasjonen. Vi valgte å kontrollere verdiene i davmail.properties filen. Alle attributtene så ut til å ha rette verdier i henhold til konfigurasjonen på deres nettsider og vår server, og vi flyttet fokuset fra DavMail over til exchange serveren.

Vi begynte med å analysere trafikken på nettverket via Wireshark. Her kunne vi se 3 requests bli sendt fra utviklings pc-en, til den virtuelle server pc-en, men ingen svar fra server pc-en. Dette fikk oss til å tenke at serveren kunne være beskyttet av en brannmur.

For å finne ut om brannmuren var problemet, aksesserte vi server pc-en via eksternt skrivebord og deaktiverte brannmuren. Etter at brannmuren ble deaktivert, kom vi endelig et steg videre med oppkoblingen.

Feilmelding: Unrecognized SSL message, plaintext connection?

Etter å ha fikset opp i feilmeldingen som følge av brannmuren, møtte vi straks på en ny feilmelding.

[2013-04-26 16:55:38] INFO [main] HttpMethodDirector – I / O exception (javax.net.ssl.SSLException) caught when processing request: Unrecognized SSL message, plaintext connection?
[2013-04-26 16:55:38] INFO [main] HttpMethodDirector – Retrying request
[2013-04-26 16:55:38] INFO [main] HttpMethodDirector – I / O exception (javax.net.ssl.SSLException) caught when processing request: Unrecognized SSL message, plaintext connection?
[2013-04-26 16:55:38] INFO [main] HttpMethodDirector – Retrying request
[2013-04-26 16:55:38] INFO [main] HttpMethodDirector – I / O exception (javax.net.ssl.SSLException) caught when processing request: Unrecognized SSL message, plaintext connection?
[2013-04-26 16:55:38] INFO [main] HttpMethodDirector – Retrying request
Exception in thread “main” org.osaf.caldav4j.exceptions.CalDAV4JException: Unrecognized SSL message, plaintext connection?

Vi begynte feilsøkingen med å søke etter lignende feil på nettet [19]. Her fant vi forumposter om lignende problemer, og lærte at årsaken til problemet kunne være at serveren forventet en sikret tilkobling, mens klienten forsøkte å koble opp usikret.

For å løse problemet endret vi innstillingene på serveren, slik at den godtok usikrede oppkoblinger. Vi endret også serverprotokollen i applikasonen fra https til http, og greide til slutt å løse problemet.

DavMail

Siden vi brukte DavMail Exchange Gateway for å oversette fra CalDAV API'et slik at exchange API'et forstod, var det vanskelig å identifisere hvilket ledd i kommunikasjonen mellom applikasjonen og serveren hvor ting gikk galt (jfr. Figur 8: 'Illustrasjon av kommunikasjon via DavMail'). Ettersom vi ikke hadde erfaring med DavMail og ikke fant konkrete bevis på at oppsettet var korrekt, måtte vi ta høyde for dette i feilsøkingen. Dette førte til at videre feilsøking også innebar å endre konfigurasjonen av DavMail, frem til vi fant bevis for at konfigurasjonen var rett.

Etter å ha prøvd og feilet med forskjellige konfigurasjoner av DavMail, greide vi til slutt å koble opp mot exchange serveren via Mac OSX sin kalender klient. Her fikk vi tilgang til, og hentet ut kalenderen til en gitt bruker, og kunne konkludere at DavMail konfigurasjonen var korrekt.

7.3 Videre Arbeid

7.3.1 Hovedapplikasjonen

Hovedapplikasjonen inneholder fortsatt funksjonalitet som ikke er i bruk eller ikke skal brukes. Dette gjelder blant annet oppkobling og spørringer mot MySQL og PostgreSQL serverne, og overflødige funksjoner laget av UNINETT, som aldri ble fjernet.

MySQL og PostgreSQL fjernet vi ikke på grunn av det må tas gradvis ettersom man erstatter databasene. Tar man det gradvis, reduserer det risikoen for at arbeidet tar lengre tid. Man får god oversikt på hvordan man ligger an, og vet hvor man har gjort en feil hvis det skulle oppstå.

Videre kan det være hensiktsmessig å utvide applikasjonens snarveier for brukerne. Eventuelle utvidelser bør bestemmes i samarbeid med en eller flere av operatørene, slik at de kan beskrive sine behov for å effektivisere deres arbeid.

7.3.2 CalDAV

Siden arbeidet med CalDAV ble avsluttet før det var fungerende, mangler applikasjonene hele denne modulen.

7.3.3 LDAP

Det eneste arbeidet som gjenstår for LDAP modulen er å modifisere attributtene som skal hentes fra Active Directory'et etter informasjonen man er ute etter.

For å gjøre det lettere å bytte hvilken server applikasjonen henter attributtene fra, kan man istedenfor å hardkode inn serverens data, hente nødvendige data fra en konfigurasjonsfil. På denne måten trenger man kun å endre data i konfigurasjonsfilen, istedenfor å gå inn i kildekoden og endre hardkodede data her. For å realisere dette, må man ha parametre i konfigurasjonsfilen med serverens data. Disse parametrene blir så tilkalt inn i Switchboard-applikasjonen ved hjelp av en javaklasse, «java.util.Properties». Se vedlegg J for konfigurasjonsoppsett.

7.4 Evaluering av gruppens arbeid

Gruppearbeidet har fungert veldig bra, og alle har jobbet tilnærmet like mye med oppgaven. Alle gruppens medlemmer kjente hverandre godt i forkant av prosjektet, både sosialt og i skole-sammenheng. Vi mener dette har bidratt til god arbeidsmoral, godt samarbeid og aktiv deltakelse i diskusjoner og beslutninger.

Gruppen består av 3 gruppemedlemmer, hvor alle studerer ingeniørfag - data. Siden alle gruppemedlemene fulgte samme studieløp var basiskunnskapene før prosjektet relativt like innad i gruppa, og innebar lite av den nødvendige kunnskapen for å gjennomføre prosjektet. Det har derfor blitt brukt et stort antall arbeidstimer på å tilegne seg nødvendige kunnskaper, slik som hvordan bruke LDAP og CalDAV.

Gjennom prosjektperioden har prosjektgruppen hatt en nokså uformell organisering. Tron Løvås har fungert som offisiell prosjektleder, men i realiteten har vi fulgt en flat organiseringsstruktur, hvor alle gruppemedlemmene har vært likeverdige i alle faser av prosjektet.

Arbeidet med prosjektet har vekslet mellom individuelt arbeid og gruppearbeid etter behov, og vi føler at denne arbeidsmetoden har fungert bra. All arbeidsfordeling har skjedd løpende under arbeidet med prosjektet. Alle gruppens medlemmer deltok aktivt i fordelingen av arbeidet, noe som førte til at alle gruppemedlemmene hadde tilnærmet like mye ansvar. Kommunikasjon med oppdragsgiver og andre nødvendige kontakter har foregått for det meste over mail eller personlig oppmøte. Mesteparten av kommunikasjonen har vært mindre spørsmål og avklaringer fra vår side. Ved formelle møter med enten oppdragsgiver eller veileder ble det tatt notater, og skrevet møtereferat. Møtereferatene er lagt ved rapporten i vedlegg B.

Selv om det til tider har vært frustrasjon på grunn av problemer/utfordringer underveis, tidsfrister og gjenstående arbeid, er vi meget fornøyd gruppens innsats.

7.5 Konklusjon

Som vår avslutningsoppgave ved Høgskolen i Gjøvik, har vi arbeidet med å tilpasse en eksisterende switchboard-applikasjonen for HiG's nye telefonisystem. Gjennom dette arbeidet har vi tilegnet oss masse nye kunnskaper innenfor emner som java programmering, LDAP og CalDAV. Samtidig har vi lært oss å arbeide ut i fra en eksisterende applikasjon, og hvordan strukturere og legge opp arbeid i en prosjektgruppe.

Når prosjektet nå er ferdig, føler vi oss fornøyde med gruppens innsats, men noe misfornøyd med resultatet. Vår begrensede erfaring innenfor flere av temaene prosjektoppgaven tok for seg, krevde at vi måtte bruke mye tid på å tilegne oss den nødvendige kunnskapen.

Prosessen med å sette seg inn i den eksisterende switchboard-applikasjonen var også meget tidskrevende. Applikasjonen begynte kun som en test av Asterisk, men på grunn av UNINETT's behov for en switchboard-applikasjon, utviklet testen seg for å dekke dette behovet. Av denne årsaken er koden mer eller mindre uten kommentarer.

Kombinasjonen av vår begrensede erfaring, kompleksiteten i koden og det uforutsette arbeidet med å gjøre kildekoden kjørbar, krevde mer tid enn antatt. Derfor strakk ikke tiden til for å fullføre CalDAV modulen.

Ved prosjektets slutt er det enighet i gruppen om at prosjektoppgaven har vært både spennende, utfordrende og lærerik.

Kapittel 8 Litteraturliste

1. Figur 1: Oversikt over systemet. Hentet fra introduksjon til bacheloroppgaven 'Switchboard' av Jon Langseth, ved IT-tjenesten, HiG.
2. DavMail. DavMail POP/IMAP/SMTP/Caldav/Carddav/LDAP Exchange Gateway [Internettside]. Utgivelsessted ukjent: DavMail; [Sist oppdatert ukjent , Brukt 4. april 2013]. Tilgjengelig fra: http://davmail.sourceforge.net/
3. The Apache Software Foundation. Apache Directory LDAP API [Internettside]. Utgivelsessted ukjent: Utgivers navn ukjent; [Sist oppdatert 2. Mai 2013, Brukt 10.Februar 2013]. Tilgjengelig fra: http://directory.apache.org/api/
4. Neil A. Wilson. Comparing Java LDAP SDK Performance [Internettside]. Austin, TX, USA: Utgivers navn ukjent; [oppdatert 31. Mai 2011] Tilgjengelig fra: http://www.dirmgr.com/blog/2011/5/31/comparing-java-ldap-sdk-performance.html
5. Novell. Java LDAP [Internettside]. Utgivelsessted ukjent: OpenLDAP; [Sist oppdatert ukjent, Brukt 10. Februar 2013]. Tilgjengelig fra: http://www.openldap.org/jldap/
6. UnboundID. LDAP SDK for Java [Internettside]. Utgivelsessted Austin, TX, USA: Utgivers navn ukjent; [oppdatert ukjent]. Tilgjengelig fra: https://www.unboundid.com/products/ldap-sdk/
7. BABEL & OSA Foundation. CalDAV4j, A CalDAV client-library for Java [Internettside]. Utgivelsessted ukjent: Utgivers navn ukjent; [oppdatert ukjent] Tilgjengelig fra: https://code.google.com/p/caldav4j/
8. Micronode. iCal4j Connector[Internettside]. Utgivelsessted ukjent: Sourceforge; [oppdatert 31. Januar 2012] Tilgjengelig fra: http://build.mnode.org/projects/ical4j-connector/project-summary.html
9. Milton. Milton Webdav [Internettside]. Utgivelsessted ukjent: Utgivers navn ukjent; [oppdatert ukjent] Tilgjengelig fra: http://milton.io
10. The Apache Software Foundation. Connection and disconnection [Internettside]. Utgivelsessted ukjent: Apache; [Sist oppdatert ukjent, Brukt 8. April 2013]. Tilgjengelig fra: http://directory.apache.org/api/user-guide/2.1-connection-disconnection.html
11. The Apache Software Foundation. Binding and unbinding (...) [Internettside]. Utgivelsessted ukjent: Apache; [Sist oppdatert ukjent, Brukt 12. April 2013]. Tilgjengelig fra: http://directory.apache.org/api/user-guide/2.2-binding-unbinding.html
12. The Apache Software Foundation. Searching (...) [Internettside]. Utgivelsessted ukjent: Apache; [Sist oppdatert ukjent, Brukt 18. April 2013]. Tilgjengelig fra: http://directory.apache.org/api/user-guide/2.3-searching.html
13. AsteriskGuru. Installation of Asterisk GUI [Internettside]. Utgivelsessted ukjent: Utgivers navn ukjent; [oppdatert ukjent] tilgjengelig fra: http://www.asteriskguru.com/tutorials/asterisk_gui.html

14. NIXCRAFT. How Do I Enable Remote Access To MySQL Database Server? [Internettside]. Utgiverssted ukjent:NIXCRAFT [Sist oppdatert 29. Juni 2012; Brukt 6. Mars 2013] Tilgjengelig fra: http://www.cyberciti.biz/tips/how-do-i-enable-remote-access-to-mysql-database-server.html

15. Clayton Mcllrath. How to Enable Remote Access to MySQL [Internettside]. Utgiverssted ukjent: Endpoint; [Sist oppdatert 30. Oktober 2012, Brukt 6. Mars 2013]. Tilgjengelig fra: http://endpoint.co/technology/enable-remote-access-mysql

16. Christoph Berg. PostgreSql Debian wiki[Internettside] Utgivers sted ukjent:PostgreSql [Sist oppdatert:1.Februar 2013, Brukt: 6.Mars 2013] http://wiki.debian.org/PostgreSql

17. DavMail. DavMail Setup as a standalone server [Internettside] Utgivelsessted ukjent: DavMail;[Sist oppdatert ukjent, Brukt 5. april 2013] Tilgjengelig fra: http://davmail.sourceforge.net/serversetup.html

18. LinkedIn. Black-Box Testing [Internettside]. 2029 Stierlin CourtMountain View, CA 94043USA: LinkedIn; [Sist oppdatert ukjent, Sitert 29. April 2013]. Tilgjengelig fra: http://www.linkedin.com/skills/skill/Black_Box_Testing

19. Karthi. Unrecognized SSL message, plaintext connection? – How to solve? [Internettside]. Utgivelsessted ukjent: Stackoverflow; [oppdatert 30. juni 2011, Brukt 20. april 2013]. Tilgjengelig fra: http://stackoverflow.com/questions/6532273/unrecognized-ssl-message-plaintext-connection-how-to-solve

Kapittel 9 Vedlegg

Vedlegg A Ordliste

Kamailio:

En åpen SIP server som kan ta imot tusenvis av samtaler per sekund. Kan også bli brukt for meldinger, men for vår del er det VoIP delen som er relevant. Det er denne serveren som håndterer alle inkommende og utgående samtaler og ruter dette videre til et kø system.

LDAP:

Dette er en applikasjons protocol for å få tilgang og vedlikeholde et oppslagsverk over IP(Internet Protocol). I våres samenheng vil dette bli brukt som en telefonbok. Der programmer går gjennom LDAP for så å hente ut informasjon som tilhører de enkelte personene.

Asterisk:

Asterisk forandrer en vanlig datamaskin til en kommunikasjons server. Dette brukes i små bedrifter,store bedrifter, kundesenter m.m. Det er denne serveren som applikasjonen snakker med(dette blir å hente samtale fra kø, svare, vidreføre og avslutte samtaler).

GPL:

Dette er en form for lisensiering, ofte i kombinasjon med gratis programvare som også inneholder kildekode. Du kan ikke modifsere og i etterkant distribuere dette programmet for inntekt.

PBX:

Den lager koblinger mellom det interne telefonnetverket ofte innen en bedrift og kobler disse ut til det ofentlige tele nettet. Hoved fordelen med dette var redusert kostnader på telefoni innad I bedriften

Voice over Internet Protocol (VoIP):

Dette er kort forklart telefoni over internett. Den bruker IP(Internet Protocol) til å overføre talekomunikasjon og multimedia, dette er alt fra standar telefon funksjon til videokonferanser og sende multimedia meldinger.

Session Initiation Protocol (SIP):
Dette er en protokoll som brukes for VoIP signalisering. Den er designet for å håntere tradisjonel telefoni, men er også laget for generell bruk til multimedia komunikasjon, ikke bare begrenset til taleprogrammer.

CalDAV:
Dette er en kalender som er standardisert som gir en klient adgang til planleggingsinformasjon til en bestemt person på en ekstern server.

UNINETT AS:
Er en bedrift som utvikler og opprettholder det norske forskningsnettet, som forbinder mer enn 200 norske utdannings- og forskningsinstitusjoner med over 300 000 brukere, og knytter dem opp mot internasjonale forskningsnett.

Switchboard:
Er en enhet som kobler en telefon til en annen eller til en virtuell telefon (eks skype) som igjen gjør at de telefonene som ble koblet sammen kan ha en samtale. Du kan se på det som en enhet som blir ringt opp til, hvor man får svar. Du sier hvor du skal, og switchboardet ruter deg videre til rett mottaker.

Java Database Connectivity(JDBC):
Dette er en API som definere hvordan en klient kan få tilgang til databasen. Den inneholder metoder for spørringer og opdateringer til databasen

POSTGRESQL:
Dette en av de kraftige database systemer. Det har sterke rykter angående pålitelighet, dataintegritet og korrekthet.

MySQL:
En veldig populært databasetjener, den er en vesentlig del av LAMP-systemer. De fleste programeringspråk kan koble til MySQL-databaser, den er mest populær for webapplikasjoner.

Structured Query Language(SQL):
Dette er et programeringsspråk for spesielle formål for å behandle data i relasjonsdatabase styringssystemer. Dette er det mest brukte språket for databaser.

Application Programming Interface(API):
Dette er en protokoll for å muligjøre kommunikasjon mellom forskjellige programvarekomponenter, det kan se på som et grensesnitt.

DavMail Exchange Gateway:
Denne applikasjonen gjør det mulig for alle e-post klienter å kommunisere med en exchange server.

Exchange server:
Dette er en e-posttjener fra Microsoft. De sentrale oppgavene til denne tjeneren er e-post, calendar og kontakter.

Icalender:
Dette er et fil format som muligjører det å sende møte eller andre arangement forespørsel over internet. Dette kan skje over e-post eller ved å lagre den som en fil med .ics

X.500:
Dette er en gammel protokol. LDAP bygger på DAP some r en del av X.500. X.500 kan nå bruke TCP/IP men LDAP er fortsatt den mest populære protokollen.

Active Directory:
Dette blir brukt til å lagre data i en struktur, dette gjør at vi kan lett finne tilbake til informasjonen vi trenger. Vi vil bruke LDAP for å søke igjennom etter kontakt info som er lagret på serveren.

Outlook Web Access (OWA):
Dette er en web løsning for å få tilgang til exchange serveren. Den kan brukes til å få tilgang til e-post, calendar, kontakter, gjøremål og dokumenter

Exchange Web Services (EWS):
EWS er et bibliotek som gir klient applikasoner funksjonalitet for å kommunisere med exchange serveren. Biblioteket gir tilgang til mye av de samme dataene som er tilgjengelige via Microsoft Office Outlook(microsofts mailklient).

Java Runtime Environment (JRE):
Java krever dette for å bli kjørt på en maskin. Den inneholder støtte filer til java programmer, samt bibliotek som er standar.

Graphical User Interface (GUI):
Forkortelse for Graphical User Interface, dette er det brukeren ser når programmet kjøres.

Extensible Markup Language (XML):
Dette er et filformat der data er organisert i en herarkisk struktur. Denne kan leses av bade mensker og applikasjoner, da det er strukturert på en ryddig måte

Microsoft Terminal Services Client (MSTSC):
Applikasjon til å koble seg opp mot et eksternt skrivebord. Som f.eks å styre en server du ikke har fysisk tilgang til.

Vedlegg B Møtereferater

Fra møter med veileder

24/01

I forkant av dagens møte hadde vi sendt Tom en kopi av prosjektplanen vår, så mesteparten av møtet gikk med på å få tilbakemeldinger fra han. Han mente vi var på rett vei i arbeidet, men vi manglet noe dybde/forklaringer generelt gjennom prosjektplanen. Vi fikk også tilbakemelding på at noen punkter(SU-modell, risikoanalyse, WBS) var litt for lite spesifikke for vårt prosjekt.

31/01

Siden vi hadde innleveringsfrist for prosjektplanen den 28. januar, ble store deler av møtet brukt på tilbakemeldinger på innleveringen. Vi fikk høre at prosjektplanen vår var bra, men kunne trenge noe finpuss. Utover dette diskuterte vi litt hvordan arbeidet bør planlegges og struktureres videre.

07/02

Til dette møtet hadde vi arbeidet med, og nesten ferdigstilt nettsiden til prosjektet. Derfor ble mesteparten av møtet brukt på tilbakemeldinger fra Tom, og litt tips til hva mer som kunne vært aktuelt å ha med på nettsiden. Resten av møtet gikk med på statusoppdateringer for arbeidet med resten av prosjektet.

14/02

Dagens møte innebar kun statusoppdateringer fra vår side. På dette tidspunktet var fremdriften i prosjektarbeidet like langt som planlagt i forprosjektet.

28/02

Dagens møte med Tom ble brukt som statusmøte. Her presenterte vi hva vi hadde gjort, hvilke problemer vi hadde møtt på under arbeidet hittil og diverse annen informasjon rundt prosjektets status. Etter vi var ferdige med å presentere statusen, fikk vi en del innspill fra Tom angående videre fremdrift og rapportskriving.

07/03

Til dagens møte med Tom, hadde vi fått applikasjonen til å kjøre, men manglet å sette opp database de nødvendige database serverene og deres innhold. Vi brukte litt tid på å oppdatere Tom på vår status, og vi fikk en del tips til å lage krav-spec, samt innhold i rapporten.

14/03

Dagens møte ble kortfattet. Vi oppdaterte veileder på status for arbeidet vårt, om våre fremtidige planer og om vår kontakt med Gurvinder og oppdragsgiver.

04/04

Frem mot dagens møte hadde vi gjort ferdig første utkast til kravspesifikasjonen. Store deler av møtet ble derfor brukt på tilbakemeldinger på dette arbeidet. Videre fikk vi noen tips til neste del av rapporten, og oppdaterte Tom på den nåværende statusen for prosjektet.

11/04

I dag var det duket for ett nytt statusmøte med veileder. Mesteparten av møtet gikk med til å presentere og forklare prosjektets nåværende status. Etter dette kom Tom med innspill angående kapittel 1, «Introduksjon» i rapporten.
Både oppdragsgiver og veileder var tilstede på dagens statusmøte.

18/04

Dagens møte ble kortvarig. Siden vi ikke ønsket noen tilbakemeldinger denne uken, brukte vi kun tid på å oppdatere veileder på vår status.

25/04

I forkant av dagens møte hadde vi skrevet første utkast til kapittel 3, «Design» i rapporten. Mesteparten av møtet ble brukt på tilbakemeldinger på denne biten, og vi oppdaterte Tom på hvordan vi lå ann med de andre delene av prosjektet.

13/05

Dette ble prosjektets siste møte med veileder. Tom kom med noen siste tilbakemeldinger til kapittel 7, ‘Avslutning’ i rapporten, og ellers konkluderte vi møtet med litt prat rundt arbeidet med prosjektet.

Fra møter med oppdragsgiver

Gjennom hele prosjektet har vi kun hatt ett formelt møte med oppdragsgiver. Grunnen til dette er at vi gjennom prosjektet har hatt muntlig eller mailkontakt med oppdragsgiver, som hele tiden har vært tilgjengelig ved IT-tjenesten ved HiG.

24/01

På dagens møte ble det for det meste prat om prosjektplanen. Vi hadde en del spørsmål til Jon angående bakgrunn og avgrensninger for oppgaven. Jon oppdaterte oss også på status angående oppsett av test-miljø, og gav oss litt mer innføring i LDAP og CalDav. Vi ble også anbefalt å bruke GIT repository eller subversion som versjonskontroll.

Vedlegg C Statusrapporter

Statusrapport 1

Innholdsfortegnelse

1 Status for

1.1 Planlegging

Etter planen vi hadde satt for oss ligger vi ca. en uke etter planlagt fremdrift. Hovedgrunnen til at vi ligger etter skjema er problemer vi støtte på for å få den eksisterende applikasjonen til å kjøre, da vi måtte selv finne de biblioteker som var brukt under utviklingen.

1.2 Organisering av gruppens arbeid og ansvarsområder

Organisering av arbeid og ansvar har skjedd løpende under arbeidet. Vi har fordelt oppgaver etter behov, og jobbet i fellesskap der dette har vært lønnsomt. Det meste av arbeidet foregår mens gruppen er samlet, som regel i skolens lokaler. Ved å arbeide på denne måten har vi alltid muligheten til å diskutere, og spørre hverandre om innspill for å løse problemer vi møter underveis.

1.3 Klargjøring av problemstilling

Det har foreløpig ikke vært noe behov for videre klargjøring av problemstilling, da oppgaveteksten og innledningen til oppgaven vi fikk av oppdragsgiver var veldig grei og presis.

1.4 Rapportskriving

Så langt har vi ikke lagt mye fokus på selve rapporten, men vi har dokumentert deler av prosessen så langt og enkelte temaer vi har diskutert.

2 Oppsummering

Selvom vi for øyeblikket ligger litt etter planlagt fremdrift, føler vi at fortsatt ligger godt ann i forhold til tidsfristen, da vi ligger godt ann med research til videre arbeid.

3 Problemer

Det første problemet vi støtte på var når vi arbeidet med å sette opp http GUI for asterisk serveren. Dette på grunn av at det var flere forskjellige versjoner av GUI'en, som virket til forskjellige versjoner av asterisk, og forskjellige fremgangsmåter for å settes opp.
Vi søtte også på problemer med å finne riktige biblioteker til kildekoden vi bruker som utgangspunkt.
Et mulig fremtidig problem kan være at vi har for lite erfaring med java programmering og Netbeans IDE.

4 Ferdige oppgaver

Nettsiden for prosjektet er ferdig designet (med forbehold for mindre endringer). Innhold vil bli oppdatert løpende dersom det blir aktuelt.
Asterisk Manager Interface (AMI) er konfigurert slik at det er klart for applikasjonen.
Asterisk http GUI er konfigurert. Her kan vi administrere køer og brukere på serveren.
Vi har funnet en java API løsning til LDAP og CalDav. Begge API'ene er lisensiert under Apache 2.0 lisens, og har fått gode anmeldelser fra tidligere brukere.

5 Oppgaver under arbeid

For øyeblikket arbeider vi med å få gjøre kildekoden kjørbar, slik at vi kan starte med våre egne endringer. Vi jobber også med å sette oss inn i java programmering, og API'ene for LDAP og CalDAV for å gjøre arbeidet lettere når vi applikasjonen er kjørbar.

6 Tidsfrister

Vi ligger som sagt en uke etter skjema. Dette har grunnlag I at vi har hatt problemer med kildekoden samt at vi har hatt noe fravær på grunn av personlige grunner. Det at vi ligger litt bak skjema er ikke kritisk men det er selvsagt ikke noe vi ønsker. Vi må bare holde fokuset oppe på det viktigste, og arbeide for å være klar for neste punkt på lista når den tid kommer.

7 Motivasjon

Motivasjonen i gruppa har vært bra! Vi føler at vi samarbeider godt sammen, og at vi klarer å fordele oppgavene på en rettferdig og effektiv måte.

8 Veilederkontakt

Vi føler at vi har fått bra tilbakemelding, Tom er alltid lett å få tak I og han er alltid godt forberedt til ukentlig møter. Vi føler at han gjør det som kan kreves av veilederen. Det er også positivt at han har liten erfaring med teknologien rundt switchboard applikasjoner og IP-telefoni, slik at han setter spørsmål til det vi skriver I raporten og får oss til å forklare begrepene mer detaljert.

Statusrapport 2

Innholdsfortegnelse

1 Status for

1.1 Planlegging

For øyeblikket ligger vi mellom 1-2 uker etter planlagt fremdrift. Dette kommer av at vi brukte lengre tid enn planlagt for å få applikasjonen til å kjøre. Vi har derfor lagt om arbeidet litt, og divergert fra prosessen vi planla i forprosjektet, ved å arbeide med kravspec og design for CalDAV samtidig som vi arbeider med LDAP programmering.

1.2 Organisering av gruppens arbeid og ansvarsområder

Organisering av arbeid og ansvar har skjedd løpende under arbeidet. Vi har fordelt oppgaver etter behov, og jobbet i fellesskap der dette har vært lønnsomt. Det meste av arbeidet foregår mens gruppen er samlet, som regel i skolens lokaler. Ved å arbeide på denne måten har vi alltid muligheten til å diskutere, og spørre hverandre om innspill for å løse problemer vi møter underveis.

1.3 Klargjøring av problemstilling

Det har foreløpig ikke vært noe behov for videre klargjøring av problemstilling, da oppgaveteksten og innledningen til oppgaven vi fikk av oppdragsgiver var veldig grei og presis.

1.4 Rapportskriving

Vi har laget oppsett for mesteparten av rapporten, men mangler fortsatt å samle en del (ferdig) dokumentasjon. I skrivende stund er vi ferdige med første revisjon av kapittel 1 og 2, og er i gang med første revisjon til kapittel 3.

2 Oppsummering

For øyeblikket ligger vi litt etter planlagt fremdrift på enkelte punkter, men litt i forkant på andre. Det mangler en del arbeid både med rapporten og selve programmeringen, men alt annet er nå på plass.

3 Problemer

Det har til tider vært vanskelig å forstå sammenhengen i kildekoden, siden kommenteringen er veldig tynn. Ellers har fremdriften gått relativt greit, uten å møte på noen større problemer.

4 Ferdige oppgaver

- Exchange server for test miljø er satt opp.
- Kapitell 1 + 2 i rapporten
- Kildekoden er Kjørbar
    - MySQL og PostgreSQL servere er oppe

5 Oppgaver under arbeid

- Ldap programering
- Komentering av koden for videre utvikling
- Kravspekeskinke
- Bestemelse av calendar protokoll

6 Tidsfrister

Da vi ligger ei uke etter skjema har vi planlagt å legge inn litt ekstra timer og fordele oppgavene sånn at alle har noe dem skal fokusere på, dette for å prøve å effektivisere arbeidet

7 Motivasjon

Motivasjonen i gruppa har vært bra! Vi føler at vi samarbeider godt sammen, og at vi klarer å fordele oppgavene på en rettferdig og effektiv måte.

8 Veilederkontakt

Vi føler at vi har fått bra tilbakemelding, Tom er alltid lett å få tak I og han er alltid godt forberedt til ukentlig møter. Vi føler at han gjør det som kan kreves av veilederen. Det er også positivt at han har liten erfaring med teknologien rundt switchboard applikasjoner og IP-telefoni, slik at han setter spørsmål til det vi skriver I raporten og får oss til å forklare begrepene mer detaljert.

Vedlegg D Arbeidslogg

Uke 2:

Arbeidstimer: ~8 timer pr. pers.	Stikkord: Research, Tidligere rapporter
Vi startet det nye semesteret og arbeidet med bacheloroppgaven denne uken. Fokuset her lå på å gjøre seg kjent med tidligere rapporter, for å klargjøre oss for arbeidet med forprosjektet de neste ukene.	

Uke 3:

Arbeidstimer: ~17 timer pr. pers.	Stikkord: Forprosjekt, Planlegging
Denne uka startet vi arbeidet med forprosjektet. Her fordelte vi ansvar for de forskjellige delene av forprosjektet, og arbeidet for det meste individuelt, men møttes for diskusjon der dette var gunstig. Ved slutten av denne uka var vi godt i gang med forprosjektet.	

Uke 4:

Arbeidstimer:~10 timer pr. pers.	Stikkord: Forprosjekt, Planlegging
I uke 4 fortsatte vi arbeidet med forprosjektet. Siden innleveringsfristen for forprosjektene var satt til mandag 28/01, måtte vi arbeide for å ferdigstille forprosjektet denne uken. Vi fortsatte å arbeide etter samme rutine som vi brukte forrige uke.	

Uke 5:

Arbeidstimer: ~16 timer pr. pers.	Stikkord: Forprosjekt, Planlegging, Nettside
Etter å ha levert forprosjektrapporten var det fortsatt rom for forbedring. Derfor benyttet vi denne uken til å forbedre prosjektrapporten etter tilbakemeldinger fra veileder både uke 4 og uke 5. Vi begynte også å se på alternativer for hvordan vi skulle lage nettsiden til prosjektet.	

Uke 6

Arbeidstimer: ~9.5 timer pr. pers.	Stikkord: Nettside, Wordpress, API, LDAP, CalDAV
Vi fikk tildelt webområde i forkant av denne uken. Derfor la vi mye arbeid i nettsiden denne uken, og ble ferdig med designet (med forbehold om mindre endringer). Samtidig arbeidet vi med siste finpuss på forprosjektet, og begynte å lete etter API'er for CalDAV og LDAP.	

Uke 7:

Arbeidstimer: ~16 timer pr. pers.	Stikkord: Testmiljø, LDAP, CalDAV, Asterisk,
I uke 7 fikk vi tilgang til den virtuelle pc-en for å kjøre servern på. Vi begynte arbeidet med å konfigurere asterisk serveren denne uken, samtidig som vi fortsatte å lete etter de beste API'ene for CalDAV og LDAP, og jobbet med å sette oss inn i kildekoden. Vi møtte på noen problemer når det gjaldt å konfigurere asterisk serveren. Disse problemene var rundt å sette opp http GUI'en for å definere brukere og køer.	

Uke 8:

Arbeidstimer: ~18.5 timer pr. pers.	Stikkord: Testmiljø, Kildekode, Biblioteker
Denne uken jobbet fortsatte vi arbeidet med å konfigurere asterisk serveren, og ble ferdig med dette. Vi fortsatte også å sette oss inn i kildekoden, samt å få applikasjonen til å kjøre. Her støtte vi på mange problemer, da vi måtte finne hvilke biblioteker (og riktig versjon av disse) som var brukt i kildekoden, samt ett par andre småfeil.	

Uke 9:

Arbeidstimer: ~16 timer pr. pers.	Stikkord: Statusmøte, Kildekode
I uke 9 arbeidet vi hovedsakelig med å gjøre oss klare for statusmøtet senere i uka. Samtidig arbeidet vi med å løse problemene vi hadde med å få kjørt kildekoden, og fortsatte å sette oss mer inn i selve koden.	

Uke 10:

Arbeidstimer: ~15 timer	Stikkord: Kildekode, MySQL, PostgreSQL, Dokumentasjon
Etter å ha støtt på mange problemer med å få kildekoden til å kjøre, fikk vi endelig dette på plass, takket være god hjelp fra oppdragsgiver. Utover dette har vi arbeidet med å sette opp postgreSQL og MySQL databaser for testmiljøet, samt arbeidet med å ta igjen noe dokumentasjon.	

Uke 11:

Arbeidstimer: ~16 timer pr. pers.	Stikkord: MySQL, PostgreSQL, Kravspesifikason, Database
Denne ukens arbeid har vært for det meste individuelt. Anders fikk ansvaret for å konfigurere MySQL serveren, og Stig for å konfigurere PostgreSQL serveren, slik at vi kunne administrere databasene fra en ekstern pc. Disse oppgavene ble gjennomført uten å møte på noen spesielle problemer. Videre arbeidet vi med å skrive kravspesifikasjon, reverse engineering av databasene (ut i fra kildekoden) og hvordan programmere med LDAP.	

Uke 12:

Arbeidstimer: ~22 timer pr. pers.	Stikkord: MySQL, PostgreSQL, Kravspesifikason
I uke 12 arbeidet vi videre med databasene til applikasjonen, og fikk tilslutt satt de opp slik at applikasjonen var kjørbar. Vi møtte kun på problemer når applikasjonen forsøkte å hente ut informasjon om en «event» til en ansatt fra databasen. Utover dette har vi arbeidet med å lage kravspec, og sette sammen andre deler av rapporten.	

Uke 13:
Påskeferie

Uke 14:

Arbeidstimer: ~30.5 timer pr. pers.	Stikkord: Exchange server, LDAP, Kildekode
Denne uken har vi arbeidet med mange temaer. Vi har blant annet satt opp en windows exchange server for å hjelpe oss programmere LDAP og arbeidet for å bedre forståelsen for applikasjonen, samt legge inn en del av våre egne kommentarer. Utover dette gjorde vi ferdig første utkast av kravspec til hovedrapporten.	

Uke 15:

Arbeidstimer: ~25 timer pr. pers.	Stikkord: LDAP, Statusmøte, Design, Oppkobling
Denne uken har vi hovedsakelig arbeidet med å implementere LDAP i applikasjonen, forbredelse til statusmøte og kapittel 4, «Design» i hovedrapporten. Vi klarte å fikse oppkobling og verifisering av bruker opp mot exchange serveren, samt å lage spørringer for å hente ut informasjon om ansatte.	

Uke 16:

Arbeidstimer: ~36.5 timer pr. pers.	Stikkord: LDAP, CalDAV, Integrering
I uke 16 arbeidet vi parallelt med LDAP- og CalDAV-programmering. Ved slutten av arbeidsuken manglet vi kun integrering for LDAP delen, mens for CalDAV var enda i startfasen, pga. problemer med de tilgjengelige API-ene.	

Uke 17:

Arbeidstimer: ~47 timer pr. pers.	Stikkord: LDAP, CalDAV, Design, Integrering
Denne ukens arbeid har vært fokusert på å ferdigstille LDAP programmeringen, fortsette med programmeringen av CalDAV og kapittel 4, «Design» i rapporten. Vi fortsatte å møte på problemer med CalDAV programmeringen denne uken, men ellers har arbeidet hatt en god flyt.	

Uke 18:

Arbeidstimer: ~58 timer pr. pers.	Stikkord: CalDAV, Hovedrapport, Koding, Testing
Etter møte med oppdragsgiver tidlig i uke 18, ble det bestemt å avslutte arbeidet med å integrere CalDAV. Resten av arbeidet denne uken ble fokusert på hovedrapporten, spesielt kapittel 5 «Koding og implementasjon» og 6 «Testing».	

Uke 19:

Arbeidstimer: ~40 timer pr. pers.	Stikkord: Arkitektur, Avslutning,
I uke 19 arbeidet vi hovedsakelig med kapittel 4 «Arkitektur» og kapittel 7 «Avslutning». Vi hadde ikke planlagt noen møter denne uken.	

Uke 20:

Arbeidstimer: ~22 timer pr. pers.	Stikkord: Avslutning, Finpuss, Innlevering
Innleveringsfristen for bacheloroppgaven var på onsdag denne uken. Vi begynte arbeidsuken med et møte med Tom, som kom med noen siste innspill for forbedring. De siste 2 dagene ble kun brukt på finpuss og ferdigstilling av rapporten.	

HØGSKOLEN I GJØVIK

Forprosjekt

Switchboard

Tron Løvås

Stig Rune Aasen

Anders Flisvang Nyen

Innhold

1 Mål og rammer

1.1 Bakgrunn

Høgskolen i Gjøvik er allerede i gang med å fornye sitt interne telefonsystem, fra tradisjonell telefoni (*PBX*) til en *VoIP* infrastruktur basert på *SIP*. Med den nye infrastrukturen er det behov for en ny *switchboard applikasjon*, da den gamle løsningen ikke kan overføres til det nye systemet, og de eksisterende løsningene er enten ikke gode nok eller for kommersielle. Høgskolen har fått tilgang på en løsning utviklet av *UNINETT AS* for lokalt bruk. Denne løsningen henter kalender- og kontakt-informasjon fra en intern sql-server. Høgskolen ønsker derfor å tilpasse UNINETT's løsning, slik at den er kompatibel med Høgskolens nye kommunikasjonssystem, noe som innebærer å hente kalender- og kontakt-informasjon fra en ekstern server.
Ved å bytte fra den gamle PBX løsningen til den nye infrastrukturen vil Høgskolen kunne samle alle sine abonenter på en løsning, som igjen vil eliminere store kostnader knyttet til den gamle telefonsentralen. Den nye VoIP løsningen gir også muligheter for utvidet funksjonalitet, som f.eks. videokonferanser/video-telefoni.

1.2 Prosjektmål

1.2.1 Effektmål

Med dette prosjektet ønsker Høgskolen i Gjøvik å kunne fullstendig erstatte den gamle PBX løsningen til fordel for en mer åpen VoIP løsning. Som årsak av dette vil Høgskolen spare tilnærmet 200 000,- i årlig, da disse faste utgiftene utgår.

1.2.2 Resultatmål

Hovetmålet med prosjektet er å lage en åpen, fungerende switchboard applikasjon som inneholder all nødvendig funksjonalitet, men er åpen for utvidelser og/eller modifikasjoner. Dette innebærer at applikasjonen bruker standarder, og at den er enkel i bruk.

1.2.3 Læringsmål

Ved å jobbe med dette prosjektet vil vi få kunskap i java, hvordan switchboard applikasjoner fungerer, ip-telefoni med fokus på hvordan *Asterisk* og *Kamailio* fungerer i praksis, samt hvordan vi henter data fra en ekstern server med LDAP og CalDav. Vi vil også lære oss hvordan man går frem for å videreutvikle en ferdig applikasjon.

1.3 Rammer

Prosjektet har fastsatt tidfrist for innlevering av prosjektrapport den 15. mai.
Utviklingen av UNINETT's applikasjon har foregått i NetBeans. Derfor vil oppdragsgiver at vi skal fortsette å bruke samme utviklingsverktøy. Programmeringsspråket brukt var hovedsakelig Java inkludert API for CalDav og LDAP, med noe *SQL*/*Postgres*/*JDBC*. Vi kan derfor regne med at vi ikke trenger noen språk utenom disse.
For å kjøre test-miljø stiller oppdragsgiver med virtuell pc, som kjører linux(debian) og standard installasjon av Asterisk. De stiller også med telefoner for testing.

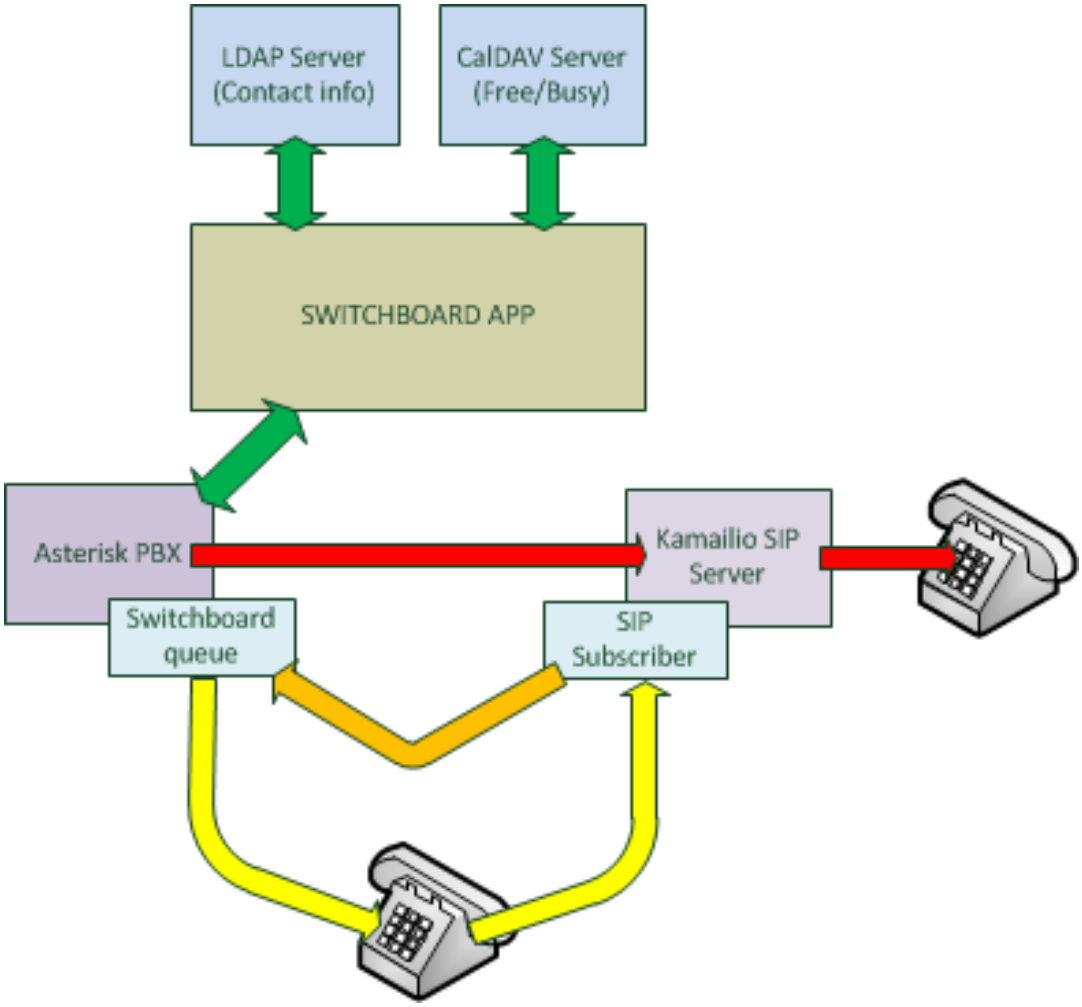
2 Omfang

2.1 Oppgavebeskrivelse

Hensikten med prosjektet er å gjøre endringer i en allerede eksisterende switchboard applikasjon, utviklet av UNINETT AS. Endringene som skal foretas skal gjøre applikasjonen mer åpen og generell enn UNINETT's løsning, slik at sluttløsningen er en åpen applikasjon som fungerer for Høgskolens nye infrastruktur og andre eventuelle brukere av SIP telefoni. Hovedoppgaven vil være å bytte ut UNINETT's løsning for å hente kalender- og kontakt-informasjon via postgressql+mysql, med en løsning som benytter seg av CalDav for kalenderinformasjon og LDAP for kontaktinformasjon. Både CalDav og LDAP brukes for å søke opp informasjon på en ekstern server, og returnere denne informasjonen til klienten. I tillegg skal applikasjonen utvides når det gjelder snarveier for brukerne, der det er behov for dette. Det skal også legges til et «varsel» for brukerne, slik at de får beskjed når det kommer et anrop, selv om de har minimert applikasjonen på pcen.
For å kunne gjennomføre endringene i oppgaven vil det være nødvendig å sette seg inn i og forstå kildekoden fra UNINETT. Dette vil være essensielt for våre endringer, slik at de utvikles etter samme struktur som de resterende delene av koden.

2.1.1 Systemkomponenter

Figuern nedenfor gir et overblikk over hele systemet, og hvordan de forskjellige komponentene henger sammen. Alle endringene vi skal gjennomføre iløpet av prosjektet ligger på «Switchboard app» siden i figuren.



Figur 1: Bildet er hentet fra Jon Langseth's introduksjon til prosjektoppgaven.

2.2 Avgrensning

Oppdragsgiver ønsker at alle protokoller og all programvare som brukes i utviklingen av prosjektet skal være uten *GPL*.
Protokollene for å hente kalender- og kontaktinformasjon er allerede bestemt av oppdragsgiver. For kalenderinformasjon skal det brukes CalDav, mens for kontaktinformasjon skal det brukes LDAP.
I og med at oppgaven går ut på å endre en eksisterende applikasjon, er det viktig at vi setter oss godt inn i kildekoden, og følger samme kodestruktur i vårt arbeid.

3 Prosjektorganisering

Prosjektgruppen består av tre dataingeniør-studenter ved Høgskolen i Gjøvik. Oppdragsgiver er Jon Langseth ved IT-avdelingen til Høgskolen i Gjøvik. Veileder er Tom Røise, Høgskolelektor ved Høgskolen i Gjøvik.

3.1 Ansvarsforhold og roller

Tron Løvås: Prosjektleder, ansvarlig for kontakt med relevante personer angående prosjektet og sammenstilling av dokumentasjon.
Anders Flisvang Nyen: Ansvarlig for fordeling og koordinering av arbeidsoppgaver.
Stig Rune Aasen: Ansvarlig for webområdet, og for å ta notater under møter.
Alle har individuelt ansvar for å notere timeforbruk, vi har felles ansvar for å fullføre møtereferat etter hvert møte. Hvert gruppemedlem er også selv ansvarlig for å sette seg inn i protokoller og rammeverk nødvendige for å løse oppgaven.
Videre ansvarsfordeling vil skje løpende under prosjektarbeidet, ettersom dette blir nødvendig.

3.2 Rutiner og regler i gruppa

Vi har følgende rutiner og regler

- Ukentlig arbeid på 25-30 timer. Dette vil forhåndsvis være mandag-fredag 10:00 til 16:00. (Med hensyn til vanlig undervisning)
- Mandager vil vi hovedsakelig ha et internt statusmøte.
- I oppstartsfasen skal vi ha møter med oppdragsgiver annenhver uke. Etter dette går vi over til å møtes en gang hver tredje eller fjerde uke etter behov.
- Vi vil ha ukentlige møte med veilder, hver torsdag kl. 10.00.
- Skrive logg etter hver arbeidsdag.
- Ved uenighet i gruppa angående beslutninger vil vi først rådføre oss med veileder. Hvis det fortsatt er uenighet i gruppa har prosjektleder myndighet til å ta den endelige avgjørelsen.

4 Planlegging, oppfølging og rapportering

4.1 Hovedinndeling av prosjektet

4.2.1 Systemutviklingsmodell

Prosjektgruppen har vurdert flere utviklingsmodeller som arbeidsmetode for vårt prosjekt, og kommet frem til følgende aktuelle modeller.

Fossefallsmodellen:

Fossefallsmodellen er en utviklingsmodell som baserer seg på en sekvensiell design prosess, hvor resultatet av hver fase gir grunnlag for arbeidet i neste fase. I og med at fremdriften skal være lineær, planlegges det ikke at man skal gå tilbake til en fase og endre på arbeidet som er gjort. Dette gir liten fleksibilitet, og forutsetter at arbeidet som blir gjort er funksjonelt. Ved bruk av fossefallsmodellen blir testing av systemet utsatt til systemet er ferdig.
Fossefallsmodellen's sekvensielle fremdrift passer bra for vårt prosjekt, bortsett fra at de enkelte fasene ikke gir grunnlag for den neste. Det passer derimot veldig dårlig å utsette testingen til systemet er ferdig, da vårt prosjekt krever at vi må kontinuerlig teste modulene ved endring. Det er også ugunstig for vårt prosjekt å ha liten fleksibilitet, da det kan komme inn ekstra funksjonalitet eller snarveier som ønskes i systemet.

Inkrementell utviklingsmodell:

For vårt prosjekt er vi ute etter en variant av fossefallsmetoden som har flere «fossefall» på grunn av at vi har to store funksjoner som vi skal redigere på i en allerede eksisterende løsning. Men for at vi skal kunne få begge til å fungere, må vi ha på plass den ene først. For den andre fungerer ikke før den første er på plass. Og nettopp dette gir den inkrementelle utviklingsmodellen oss en enkel arbeidsløsning på. Den har et fossefall pr. komponent, som igjen gjør at den ene komponenten er ferdigstilt før man begynner på den neste. Vi har fått oppgitt alle hovedkomponentene som oppdragsgiver ønsker seg i vår løsning, som igjen gjør det lett for oss å planlegge arbeidet, men vi må ta høyde for eventuelle ekstra funksjonaliteter fra oppdragsgiver. Den inkrementelle modellen gjør planleggingsarbeidet lett, samtidig som den er fleksibel nok til å takle eventuelle ekstra funksjonaliteter. Vi har fått informasjon av oppdragsgiver at hvis vi får tid, så har oppdragsgiver to funksjonaliteter som han ønsker seg.

V-modellen:

V-model er en utvidelse av fossefallsmodellen der hovedforskjellen er at etter hver utviklingsfase så har du en korosponderende fase der du har testing. Dette har ført til at modellen har blitt kalt verifikasjon og validerings model. De viktigste fordelene med v-model er at det er enkelt å bruke. Denne modellen passer vårt prosjekt fordi den er best på små prosjekt der kravene er lett forstått. Men med tanke på at vi skal ha muligheten for å legge inn ekstra funksjoner vil dette føre med store ulemper, da modellen er lite fleksibel.

Konklusjon:

Etter å ha diskutert fordeler og ulemper ved de forskjellige utviklingsmodellene, valgte vi å arbeide etter den inkrementelle modellen. Både fossefallsmodellen og V-modellen vil gi for lite fleksibilitet i forhold til hva vi ønsker for vårt prosjekt. Den inkrementelle modellen gir nok fleksibilitet, og arbeidsstrukturen, som baserer seg på å ha ett fossefall pr. modul passer godt for vår oppgave, ettersom vi har få, tidkrevende og uavhengige moduler.

4.2 Plan for statusmøter

Dette er møter vi kommer til å ha med både veileder og oppdragsgiver. Oppdragsgiver ønsket å ha muligheten til å si nei til å komme, noe vi har gitt han lov til. Dette er møter som kommer til å inneholde hvordan vi ligger an i prosjektet og eventuelle tips veileder og oppdragsgiver har til hvordan vi fortsetter arbeidet. Vi ønsker å ha 2 statusmøter i løpet av prosjektet og vi ønsker å holde de i uke 9 og i uke 14.

5 Organisering av kvalitetssikring

5.1 Dokumentasjon

Alle prosjektdokumenter utenom kildekoden skal lagres i docx-format, for å gjøre sammenstilling av dokumenter lettere.
Etter alle møter med veileder og oppdragsgiver skal det utarbeides et møtereferat. Møtereferatene og notater tilhørende møtene skal sikre at prosjektarbeidet fortsetter i riktig retning, ut i fra oppdragsgivers ønsker.
Timeforbruk skal dokumenteres induviduelt. Her skal det føres opp når, hvilken modul det ble arbeidet på, og hvor lenge arbeidet foregikk(avrundet til halve timer). Hvert gruppemedlem har et eget dokument hvor dette føres.

5.2 Kildekode

For å gjøre eventuelle senere utvidelser av applikasjonen lettere, skal vi hele tiden kommentere kildekoden vi legger til. Dette gjelder spesielt for ikke-triviell kode, og avanserte algoritmer. Det er også vesentlig at den nye kildekoden følger samme kodestruktur som den eksisterende koden.
Ved utvikling av ny kode skal alle variable som opprettes navngis med forhåndsbestemt id og understrek, etterfulgt av selve variabelnavnet (eks. t_antall). Dette tiltaket skal sikre at det ikke finnes variable med like navn i forskjellige deler av kildekoden.
All utvikling av kode skal foregå i NetBeans IDE 7.2.1 /7.3 beta 2.

5.3 Konfigurasjonsstyring

Som versjonskontroll for kildekode har vi valgt å benytte git. Git er et gratis versjonskontroll verktøy, med åpen kildekode. Vi valgte å bruke git istedenfor andre alternativer som subversion, fordi vi ble fortalt at git var et mer fremtidsrettet verktøy, som vi ville ha større verdi av i senere situasjoner.
Vi har valgt å kun benytte versjonskontroll for kildekode-dokumenter. Alt individuelt arbeid skal lagres i lokale filer, og skal sammenstilles når hele gruppa er samlet, hovedsakelig under de interne møtene på mandager.

5.4 Riskoanalyse

Prio	Beskrivelse	Konsekvens	Sannsynlighet	Risikonivå	Akseptabelt?	Ev. Tiltak
1	Finne API som er CalDaVog LDAP og samtidig ikke GPL	Kritisk	Lite sannsynlig	Middels	Nei	Fortsette letingen og samtidig finne en alternativ løsning
2	Ikke få testmiljøet opp	Kritisk	Lite sannsynlig	Høy	Nei	Ta kontakt med Gurvinder Singh
3	Kode for tett knyttet til nåværende løsning på kontakter og kalender	Høy	Sannsynlig	Høy	Ja	Bryte opp mer kode
4	Riktig tid og dato kan være vanskelig å få rett i CalDav	Lav	Sannsynlig	Middels	Nei	Ta kontakt med en person med ekspertise på dette område
5	Tap av data	Høy	Lite sannsynlig	Høy	Nei	Ha gode backup rutiner, ev. gjennopprette det tapte
6	Lite effektivitet i arbeidet	Middels	Sannsynlig	Middels	Til en viss grad	Motivere hverandre til å yte mer effektivt!
7	Lite kompetanse	Middels	Stor sannsynlig	Middels	Ja	
8	Fravær	Liten	Sannsynlig	Lav	Ja	
9	Tidsfrister	Liten	Lite sannsynlig	Lav	Nei	God dialog med oppdragsgiver

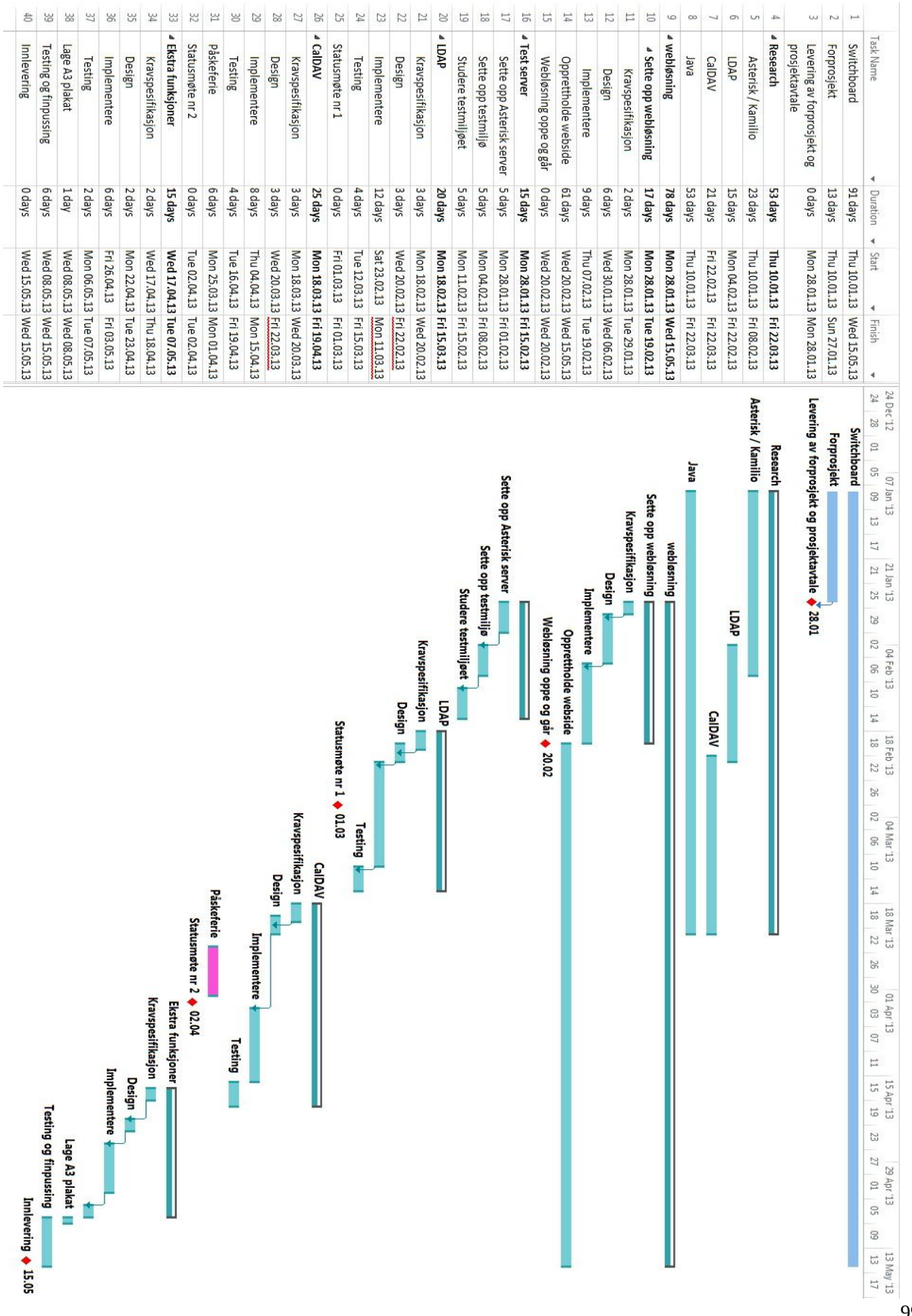
5.4.1 Forebyggende tiltak

Det største problemet vi kan møte på angående prosjektet og dets mål vil være å ikke finne en API som støtter CalDaV og LDAP og samtidig ikke avhenger av GPL. Sannsynligheten for at dette blir realitet er svært liten, men det er en risiko vi må ta med. Tiltak som vi kan ta i bruk er å fortsette letingen og samtidig vurdere siste utveien med å lage en selv.
Et annen kritisk risiko vi kan møte på er at vi ikke får opp et testmiljø, men sannsynligheten her også er svært liten ettersom det er et program som allerede er i bruk. Jon Langseth skal være med oss å sette opp dette testmiljøet, og hvis det skulle bli komplikasjoner, har vi muligheten til å ta kontakt med en fagperson med navn Gurvinder Singh.
Den siste risikoen som vi anser som ikke akseptabel og som er verdt å nevne er om tid og dato ikke blir i rett format og rett tid / dato i CalDaV. I tidligere prosjekter med CalDaV har dette kommet frem, og har vært litt vanskelig å ha med å gjøre. Det vi kan gjøre for å unngå dette er å søke opp informasjon om problemet og se om vi finner en vei til å unngå det. Eller vi kan høre med en ved IT-tjenesten på HiG som har erfaring med akkurat dette.

6 Plan for gjennomføring

6.1 Gantt-skjema

	Task Name	Duration	Start	Finish
1	Switchboard	91 days	Thu 10.01.13	Wed 15.05.13
2	Forprosjekt	13 days	Thu 10.01.13	Sun 27.01.13
3	Levering av forprosjekt og prosjektavtale	0 days	Mon 28.01.13	Mon 28.01.13
4	**Research**	**53 days**	**Thu 10.01.13**	**Fri 22.03.13**
5	Asterisk / Kamilio	23 days	Thu 10.01.13	Fri 08.02.13
6	LDAP	15 days	Mon 04.02.13	Fri 22.02.13
7	CalDAV	21 days	Fri 22.02.13	Fri 22.03.13
8	Java	53 days	Thu 10.01.13	Fri 22.03.13
9	**webløsning**	**78 days**	**Mon 28.01.13**	**Wed 15.05.13**
10	**Sette opp webløsning**	**17 days**	**Mon 28.01.13**	**Tue 19.02.13**
11	Kravspesifikasjon	2 days	Mon 28.01.13	Tue 29.01.13
12	Design	6 days	Wed 30.01.13	Wed 06.02.13
13	Implementere	9 days	Thu 07.02.13	Tue 19.02.13
14	Opprettholde webside	61 days	Wed 20.02.13	Wed 15.05.13
15	Webløsning oppe og går	0 days	Wed 20.02.13	Wed 20.02.13
16	**Test server**	**15 days**	**Mon 28.01.13**	**Fri 15.02.13**
17	Sette opp Asterisk server	5 days	Mon 28.01.13	Fri 01.02.13
18	Sette opp testmiljø	5 days	Mon 04.02.13	Fri 08.02.13
19	Studere testmiljøet	5 days	Mon 11.02.13	Fri 15.02.13
20	**LDAP**	**20 days**	**Mon 18.02.13**	**Fri 15.03.13**
21	Kravspesifikasjon	3 days	Mon 18.02.13	Wed 20.02.13
22	Design	3 days	Wed 20.02.13	Fri 22.02.13
23	Implementere	12 days	Sat 23.02.13	Mon 11.03.13
24	Testing	4 days	Tue 12.03.13	Fri 15.03.13
25	Statusmøte nr 1	0 days	Fri 01.03.13	Fri 01.03.13
26	**CalDAV**	**25 days**	**Mon 18.03.13**	**Fri 19.04.13**
27	Kravspesifikasjon	3 days	Mon 18.03.13	Wed 20.03.13
28	Design	3 days	Wed 20.03.13	Fri 22.03.13
29	Implementere	8 days	Thu 04.04.13	Mon 15.04.13
30	Testing	4 days	Tue 16.04.13	Fri 19.04.13
31	Påskeferie	6 days	Mon 25.03.13	Mon 01.04.13
32	Statusmøte nr 2	0 days	Tue 02.04.13	Tue 02.04.13
33	**Ekstra funksjoner**	**15 days**	**Wed 17.04.13**	**Tue 07.05.13**
34	Kravspesifikasjon	2 days	Wed 17.04.13	Thu 18.04.13
35	Design	2 days	Mon 22.04.13	Tue 23.04.13
36	Implementere	6 days	Fri 26.04.13	Fri 03.05.13
37	Testing	2 days	Mon 06.05.13	Tue 07.05.13
38	Lage A3 plakat	1 day	Wed 08.05.13	Wed 08.05.13
39	Testing og finpussing	6 days	Wed 08.05.13	Wed 15.05.13
40	Innlevering	0 days	Wed 15.05.13	Wed 15.05.13



6.1.1 Kommentarer til Gantt-skjema

Vi har valgt å satt opp gannt-skjema med de aktivitetene som foregår over lengre tid som igjen ikke har mange underpunker øverst for å få en bedre oversikt, som for eks research og webløsningen.

Forprosjekt:

I løpet av forprosjektet vil vi så klart jobbe med forprosjekt, men samtidig få en god oversikt over hvordan vi vil planlegge prosjektet. Samt få til kontinuelig møter med veileder, og møtene vi skal ha med oppdragsgiver.

Research:

Research er noe vi vil drive med for det meste første halvdel av bachelor oppgaven. Dette er noe vi vil jobbe med samtidig som de andre oppgavene vi skal jobbe med. Det vi kommer til å bruke meste av tiden til researchen på er Asterisk, Kalimio, CalDaV, LDAP og Java

Webløsningen:

Webløsningen er en nettside vi skal lage og opprettholde i løpet av hele bachelor oppgaven. Den kommer til å inneholde fremgangen vi har, møtereferater, hvordan vi skal gå videre med prosjektet. Litt generelt om hvordan prosjektet foregår. Ikke for mye i detalj, men en liten oversikt for allmennheten.

Ekstra funksjoner:

Hvis alt går som det skal, så vil vi få litt ekstra tid til å legge til flere funksjoner som oppdragsgiver kunne ønske seg, men som egentlig ikke var en del av prosjektet.

Vedlegg F Bibliotekliste

Biblioteker som ble brukt i prosjektet

- Antlr-2.7.7.jar
- Api-all-1.0.0-M16.jar
- Commons-collections-3.2.1.jar
- Commens-io-2.4.jar
- Commons-lang-2.6.jar
- Commons-pool-1.6.jar
- Dom4j-1.6.1.jar
- Log4j-1.2.17.jar
- Mina-core-2.0.7.jar
- Slf4j-api-1.7.2.jar
- Slf4j-log4j12-1.7.2.jar
- Xml-apis-2.0.2.jar
- Xpp3-1.1.4c.jar
- Appframework-1.0.3.jar
- Asterisk-java.jar
- Commons-configuration-1.9.jar
- Commons-logging-1.1.1.jar
- Eclipselink.jar
- Javax.persistence_2.0.4.v201112161009.jar
- Jcalendar-1.4.jar
- Log4j-1.2.9.jar
- Mysql-connector-java-5.1.23-bin.jar
- Postgresql-9.2-1002.jdbc3.jar
- Swing-layout-1.0.3.jar
- Swing-worker-1.1.jar

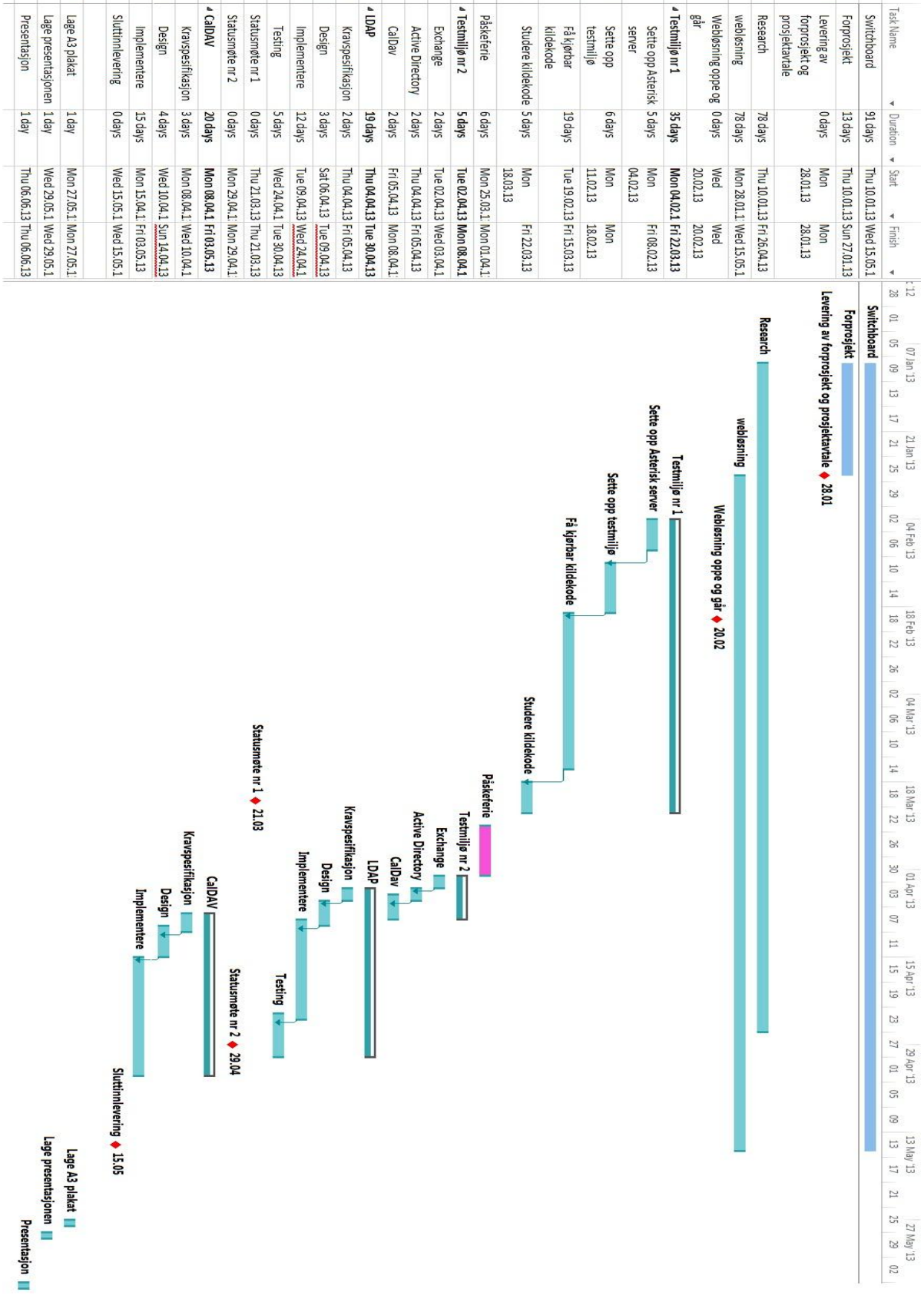
Vedlegg G Tidligere bachelorrapporter

Bachelorrapporter ved HiG som ble hentet inspirasjon fra

- Autoklav, år 2006
- Fredrik R Hørtvedt, Fredrik Jacobsen Kvitvik, Jørn André Myrland, år 2011
- Hevnj, år 2002
- KjøreRute, år 2010
- Stopmotion, år 2005
- Thagehaugen_GGOlimb_MKUrrang, år 2012

Vedlegg H Ganntskjema

Gannt-skjema, slik fremgangen ble



	Task Name	Duration	Start	Finish
1	Switchboard	91 days	Thu 10.01.13	Wed 15.05.1
2	Forprosjekt	13 days	Thu 10.01.13	Sun 27.01.13
3	Levering av forprosjekt og prosjektavtale	0 days	Mon 28.01.13	Mon 28.01.13
4	Research	78 days	Thu 10.01.13	Fri 26.04.13
5	webløsning	78 days	Mon 28.01.1	Wed 15.05.1
6	Webløsning oppe og går	0 days	Wed 20.02.13	Wed 20.02.13
7	Testmiljø nr 1	35 days	Mon 04.02.1	Fri 22.03.13
8	Sette opp Asterisk server	5 days	Mon 04.02.13	Fri 08.02.13
9	Sette opp testmiljø	6 days	Mon 11.02.13	Mon 18.02.13
10	Få kjørbar kildekode	19 days	Tue 19.02.13	Fri 15.03.13
11	Studere kildekode	5 days	Mon 18.03.13	Fri 22.03.13
12	Påskeferie	6 days	Mon 25.03.1	Mon 01.04.1
13	Testmiljø nr 2	5 days	Tue 02.04.13	Mon 08.04.1
14	Exchange	2 days	Tue 02.04.13	Wed 03.04.1
15	Active Directory	2 days	Thu 04.04.13	Fri 05.04.13
16	CalDav	2 days	Fri 05.04.13	Mon 08.04.1
17	LDAP	19 days	Thu 04.04.13	Tue 30.04.13
18	Kravspesifikasjon	2 days	Thu 04.04.13	Fri 05.04.13
19	Design	3 days	Sat 06.04.13	Tue 09.04.13
20	Implementere	12 days	Tue 09.04.13	Wed 24.04.1
21	Testing	5 days	Wed 24.04.1	Tue 30.04.13
22	Statusmøte nr 1	0 days	Thu 21.03.13	Thu 21.03.13
23	Statusmøte nr 2	0 days	Mon 29.04.1	Mon 29.04.1
24	CalDAV	20 days	Mon 08.04.1	Fri 03.05.13
25	Kravspesifikasjon	3 days	Mon 08.04.1	Wed 10.04.1
26	Design	4 days	Wed 10.04.1	Sun 14.04.13
27	Implementere	15 days	Mon 15.04.1	Fri 03.05.13
28	Sluttinnlevering	0 days	Wed 15.05.1	Wed 15.05.1
29				
30	Lage A3 plakat	1 day	Mon 27.05.1	Mon 27.05.1
31	Lage presentasjonen	1 day	Wed 29.05.1	Wed 29.05.1
32	Presentasjon	1 day	Thu 06.06.13	Thu 06.06.13

Vedlegg I Ldapkonfigurasjonsfil

```
# LDAP parameters
ldap.host = ldap://128.39.142.36:389
ldap.factories.initctx = com.sun.jndi.ldap.LdapCtxFactory
ldap.factories.control = com.sun.jndi.ldap.ControlFactory
ldap.searchbase = dc=not-hig,dc=test
ldap.user = 128.39.142.36\\CN=Anders Tester,CN=Users,DC=not-
hig,DC=test
ldap.userBase= 128.39.142.36\\
ldap.password = 123abcLOL
```

Vedlegg J Liste over filer som er koblet sammen

Overskriften sier filnavnet til filen som blir brukt i fil-listen den har under seg.

UsrPreferences.java

- EmpTableModel.java
- OpenserJpaController.java
- UsrPreferences.java
- Persistence.xml

SwitchBoardView.java

- EmpSelectionListener.java
- EmpTableModel.java
- PollStateVariable.java
- SBAsteriskManager.java
- SwitchBoardApp.java
- SwitchBoardView.java

SwitchBoardApp.java

- CalendarJpaController.java
- CalendarSelectionListener.java
- CalendarTabelModel.java
- CallSelectionListener.java
- CallTableRowRenderer.java
- CdrBox.java
- CdrTableModel.java
- CheckLock.java
- Clock.java
- EmpSelectionListener.java
- EmpTableModel.java
- LogTableModel.java
- MakeCall.java
- OpenserJpaController.java
- PollstateVariable.java

- SBAsteriskManager.java
- SBTableRowRenderer.java
- SwitchBoardAboutBox.java
- SwitchBoardApp.java
- SwitchBoardView.java

SwitchBoardAboutBox

- SwitchBoardAboutBox.java
- SwitchBoardView.java

ShowMessageDialouge

- EmpTableModel.java
- SwitchBoardView.java

SBTableRowRenderer

- CalendarTableModel.java
- EmpTableModel.java
- SBTableColRenderer.java

SBTableColRenderer

- CalendarTabelModel.java
- EmpTableModel.java
- SBTableColRenderer.java

SBAsteriskManager

- CallSelectionListener.java
- MakeCall.java
- PollStateVariable.java
- SwitchBoardView.java

PollStateVariable

- SwitchBoardView.java

OpenserJpaController

- CdrBox.java
- EmpTableModel.java
- SBAsteriskManager.java
- SwitchBoardView.java

MembershipPK

- Membership.java
- MembershipPK.java

Membership

- CalendarJpaController.java
- Membership.java
- Persistence.xml

MakeCall

- SBAsteriskManager.java

LogTableRowRenderer

- CdrTableModel.java
- LogTableModel.java

LogTableModel

- SBAsteriskManager.java
- SwitchBoardView.java

Location

- Location.java

- Persistence.xml

Ldap

- SwitchBoardView.java

GuestVisitors

- CalendarJpaController.java
- CalendarTableModel.java
- GuestVisitors.java
- Persistence.xml

GuestVisitations

- CalendarJpaController.java
- CalendarTabelModel.java
- GuestVisitations.java
- Persistence.xml

Grouplist

- CalendarJpaController.java
- Grouplist.java
- Persistence.xml

Event

- CalendarJpaController.java
- CalendarTabelModel.java
- EmpTableModel.java
- Event.java
- OpenserJpaController.java
- Persistence.xml

Employee

- CalendarJpaController.java

- EmpTableModel.java
- Ldap.java
- Persistence.xml

EmpTableModel

- CalendarTabelModel.java
- CdrBox.java
- CdrTableModel.java
- EmpSelectionListener.java
- LogTableModel.java
- SBAsteriskManager.java
- SwitchBoardView.java

EmpSelectionListener

- SwitchBoardView.java

Clock

- SBAsteriskManager.java

CheckLock

- SwitchBoardApp.java

CdrTableModel

- CdrBox.java

CdrBox

- CdrBox.java
- SwitchBoardView.java

Categorylist

- CalendarJpaController.java
- CalendarTabelModel.java
- Categorylist.java
- Persistence.xml

CallTableRowRenderer

- CallTableModel.java

CallTableModel

- Clock.java
- SBAsteriskManager.java
- SwitchBoardView.java

CallSelectionListener

- SwitchBoardView.java

CalendarTabelModel

- CalendarSelectionListener.java
- EmpTableModel.java
- SwitchBoardView.java

CalendarSelectinListener

- SwitchBoardView.java

CalendarJpaController

- CalendarTabelModel.java
- EmpTableModel.java
- SwitchBoardView.java

Acc

- Acc.java
- CdrTableModel.java
- LogTableModel.java
- OpenserJpaController.java
- Persistence.xml